Information-Centric Communication in Mobile and Wireless Networks

Inauguraldissertation
der Philosophisch-naturwissenschaftlichen Fakultät
der Universität Bern

vorgelegt von

Carlos Anastasiades

von Zürich, ZH

Leiter der Arbeit:
Professor Dr. Torsten Braun
Institut für Informatik

Information-Centric Communication in Mobile and Wireless Networks

Inauguraldissertation
der Philosophisch-naturwissenschaftlichen Fakultät
der Universität Bern

vorgelegt von

Carlos Anastasiades

von Zürich, ZH

Leiter der Arbeit:
Professor Dr. Torsten Braun
Institut für Informatik

Von der Philosophisch-naturwissenschaftlichen Fakultät angenommen.

Bern, 01.06.2016

Der Dekan:
Prof. Dr. Gilberto Colangelo

Abstract

Information-centric networking (ICN) is a new communication paradigm that has been proposed to cope with drawbacks of host-based communication protocols, namely scalability and security. In this thesis, we base our work on Named Data Networking (NDN), which is a popular ICN architecture, and investigate NDN in the context of wireless and mobile ad hoc networks.

In a first part, we focus on NDN efficiency (and potential improvements) in wireless environments by investigating NDN in wireless one-hop communication, i.e., without any routing protocols. A basic requirement to initiate information-centric communication is the knowledge of existing and available content names. Therefore, we develop three opportunistic content discovery algorithms and evaluate them in diverse scenarios for different node densities and content distributions. After content names are known, requesters can retrieve content opportunistically from any neighbor node that provides the content. However, in case of short contact times to content sources, content retrieval may be disrupted. Therefore, we develop a requester application that keeps meta information of disrupted content retrievals and enables resume operations when a new content source has been found. Besides message efficiency, we also evaluate power consumption of information-centric broadcast and unicast communication. Based on our findings, we develop two mechanisms to increase efficiency of information-centric wireless one-hop communication. The first approach called Dynamic Unicast (DU) avoids broadcast communication whenever possible since broadcast transmissions result in more duplicate Data transmissions, lower data rates and higher energy consumption on mobile nodes, which are not interested in overheard Data, compared to unicast communication. Hence, DU uses broadcast communication only until a content source has been found and then retrieves content directly via unicast from the same source. The second approach called RC-NDN targets efficiency of wireless broadcast communication by reducing the number of duplicate Data transmissions. In particular, RC-NDN is a Data encoding scheme for content sources that increases diversity in wireless broadcast transmissions such that multiple concurrent requesters can profit from each others' (overheard) message transmissions.

If requesters and content sources are not in one-hop distance to each other, requests need to be forwarded via multi-hop routing. Therefore, in a second part of this thesis, we investigate information-centric wireless multi-hop communication. First, we consider multi-hop broadcast communication in the context of rather static community networks. We introduce the concept of preferred forwarders, which relay Interest messages slightly faster than non-preferred forwarders to reduce redundant duplicate message transmissions. While this approach works well in static networks, the performance may degrade in mobile networks if preferred forwarders may regularly move away. Thus, to enable routing in mobile ad hoc networks, we extend DU for multi-hop communication. Compared to one-hop

communication, multi-hop DU requires efficient path update mechanisms (since multi-hop paths may expire quickly) and new forwarding strategies to maintain NDN benefits (request aggregation and caching) such that only a few messages need to be transmitted over the entire end-to-end path even in case of multiple concurrent requesters. To perform quick retransmission in case of collisions or other transmission errors, we implement and evaluate retransmission timers from related work and compare them to CCNTimer, which is a new algorithm that enables shorter content retrieval times in information-centric wireless multi-hop communication. Yet, in case of intermittent connectivity between requesters and content sources, multi-hop routing protocols may not work because they require continuous end-to-end paths. Therefore, we present agent-based content retrieval (ACR) for delay-tolerant networks. In ACR, requester nodes can delegate content retrieval to mobile agent nodes, which move closer to content sources, can retrieve content and return it to requesters. Thus, ACR exploits the mobility of agent nodes to retrieve content from remote locations. To enable delay-tolerant communication via agents, retrieved content needs to be stored persistently such that requesters can verify its authenticity via original publisher signatures. To achieve this, we develop a persistent caching concept that maintains received popular content in repositories and deletes unpopular content if free space is required. Since our persistent caching concept can complement regular short-term caching in the content store, it can also be used for network caching to store popular delay-tolerant content at edge routers (to reduce network traffic and improve network performance) while real-time traffic can still be maintained and served from the content store.

Contents

List of Figures

List of Tables

Preface

The work presented in this thesis has been performed during my employment as research assistant and PhD student at the Institute of Computer Science (INF) of the University of Bern. This work has been funded mainly by the State Secretariat for Education, Research and Innovation (SBFI) under project number C10.0139.

I would like to thank all people that supported me during the course of this thesis. First, I want to express my gratitude to my supervisor, Prof. Torsten Braun, for his trust and support throughout the years. He allowed me to work independently on projects but continuously challenged my ideas, encouraged me to publish them in reputable journals and conferences, and enabled me to travel to scientific conferences, European COST meetings as well as project proposal meetings. During my time in Bern, I could grow and develop many skills in a way I have never expected before. I would also like to thank Prof. Vassilis Tsaoussidis, the second examiner of this thesis, for his valuable comments that helped me improving the thesis, and for his great hospitality during my STSM stay at the SPICE center in Xanthi. Furthermore, I would like to thank Prof. Matthias Zwicker, who was willing to be a co-examiner of this thesis.

Many thanks go to my colleagues and former office mates at the University of Bern for many interesting discussions, good cooperations and fun events. In particular, I would like to thank Imad Aad, Markus Anwander, Eirina Bourtsoulatze, Marc Brogle, Benjamin Nyffenegger, Desislava Dimitrova, Mikael Gasparyan, André Gomes, Philip Hurni, Almerima Jamakovic-Kapic, Zan Li, Dima Mansour, Ali Marandi, Dragan Milic, Andreea Picu-Hossmann, Denis Rosario, Jonnahtan Saltarin, Eryk Schiller, Thomas Staub, Marcel Stolz, Nikolaos Thomos, Gerald Wagenknecht and Zhongliang Zhao. I would also like to thank my students Tobias Schmid, Jürg Weber, René Gadow, Lukas von Rotz, Arun Sittampalam, Alexander Striffeler, Wafaa El Maudni El Alami, Arian Uruqi, Nina Mujkanovic and Christoph Knecht for their efforts and excellent works. It was a pleasure working with such motivated and talented students.

Special thanks go also to Ruth Bestgen and Daniela Schroth, the former and current secretary of the CDS group, for their kind support of administrative tasks and for many personal advices and suggestions. They have put much efforts into improving our working atmosphere and making our daily lives in the group more enjoyable. I would also like to thank Peppo Brambilla, the system administrator of the INF institute, for his excellent and prompt support and for joining our PhD cof-

fee and lunch breaks. It was very nice to work with such an effective and friendly system administrator. Furthermore, I am also grateful to Michael Rolli and the Ubelix team for their great support with the Linux cluster providing me and my students a fantastic tool for our evaluations. Without Ubelix, many evaluations would have not been performed in time.

This PhD thesis was performed in the context of COST Action IC0906 "Wireless Networking for Moving Objects (WiNeMo)". Therefore, I could attend European COST meetings and go on research exchanges. During the course of the Action, I have met many nice people and could participate in interesting discussions. In particular, I would like to thank Yevgeni Koucheryavy, Marilia Curado, Vasilios Siris, Edmundo Monteiro, Geert Heijenk, Andreas Kassler, Ivan Ganchev and Paulo Mendes for all their efforts and support. Many thanks go also to Ioannis Komnios, Nikolaos Bezirgiannidis, Diego Borsetti, Sotirios Diamantopoulos and Sotiris-Angelos Lenas for the great hospitality during my STSM stay in Xanthi, making me feel at home.

Finally, I would like to thank my parents Silvia and Tasso Anastasiades and my brother Bryan Anastasiades. They always stood by me and supported me during all periods of disbelief and despair whenever I needed to recharge my mind. These moments of peace, when I visited them and could enjoy an exquisite meal with a glass of good wine, were my source of inspiration that guided me through the past years.

Part I

Introduction, Related Work and Evaluation Methodology

In Chapter 1, we motivate our work, identify open challenges and list the contributions of this thesis. Chapter 2 gives an overview of related work for all topics addressed in this thesis. Then, Chapter 3 presents our evaluation methodology by listing the evaluation tools and evaluated parameters.

Chapter 1

Introduction

Communication systems have evolved around a host-centric networking model to enable communication between well-defined locations. The first electric telegraphs at around 1838 made it possible to transmit encoded messages (Morse code) over dedicated cables between entities several kilometers apart. Similarly, the first electric telephones at around 1876 used dedicated cables to enable direct transmission of voice. Since dedicated cables turned out to be impractical with an increasing number of users, electric circuit switching has been developed enabling circuits between callers and callees to be connected at a central switching system.

The first computer networks that appeared in the early 1960s enabled users to remotely use (and share) scarce and expensive devices such as card readers, supercomputers or printers. At that time, the number of connected computers and the transferred amount of data was very small. Similar to the telephone network, the communication model included the conversation between exactly two machines: one machine to use the resource and one machine to provide access to it. However, different from telephone networks, it was not required to establish permanent circuits but packets were switched based on their intended destination. To unify communication among heterogeneous networks (as basis for the Internet), the Transmission Control Protocol and IP Protocol (TCP/IP) was specified in mid 1970s and has been widely developed in early 1980s ensuring end-to-end message transfer independent of the underlying network, including error control, packet segmentation, flow and congestion control as well as application addressing (port numbers).

Since the introduction of the world wide web (WWW) in the mid 1990s, access to information in the Internet has been facilitated for the general public and the Internet has moved from a host-centric (resource sharing) communication model to a universal tool to access content as illustrated in Figure 1.1. Today, the Internet has become our primary source of information, e.g., via news sites, interactive discussion forums, blogs, social networking, video-on-demand (VoD) or online shopping. In general, content location does not matter anymore as long as the content is authentic. However, communication is still based on the network architectures designed in the 70s.

Network traffic projections [79, 88] estimate a massive traffic increase for the

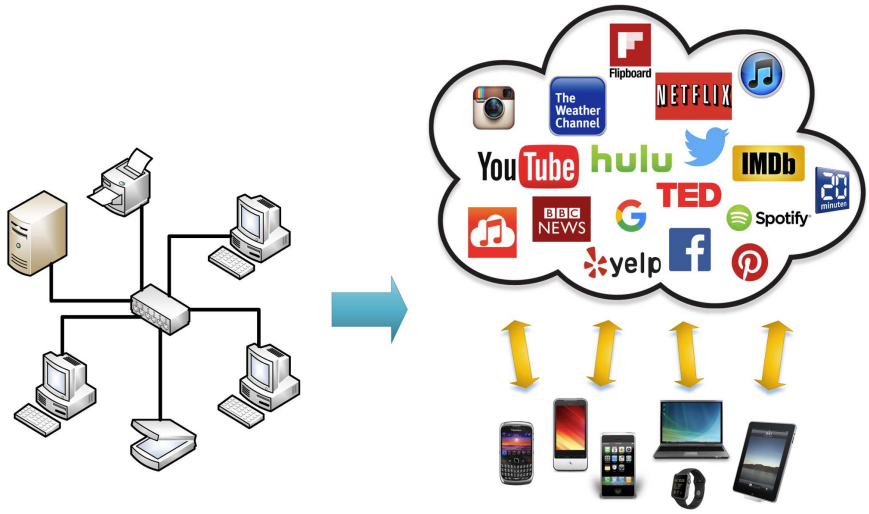


Figure 1.1: Shift from Resource Sharing to Content Delivery Networks.

next five years, which is mainly caused by two reasons. First, we are entering a connected world, where microprocessors are embedded in nearly all electronic devices because they have become cheap, powerful and more energy efficient, enabling those devices to collect and exchange data at any time. The number of mobile devices has already increased drastically in recent years. Since 2014 there are more mobile devices than people on earth [79], and the number of embedded devices is expected to increase even more, e.g., by connected smart homes and the Internet of Things (IoT). Second, the average data traffic per device is expected to increase due to higher (cellular and fixed line) data rates enabling the exchange of more dynamic content, e.g., personal photos and videos, via social networking as well as exchanging larger content sizes, e.g., through high definition video streaming. This increase in data traffic calls for new solutions and network architectures to address communication more efficiently in Future Internet communication. There have been many projects and activities addressing different aspects of the Future Internet, e.g., as listed by the European Future Internet Assembly (FIA) [10], the Asia Future Internet Forum (AsiaFI) [2] or the NSF Future Internet Architecture (NSF-FIA) [18]. Among all these approaches, Information-Centric Networking (ICN) addresses the particular challenges that the current Internet architecture is facing for increased data traffic and user mobility.

1.1 Overview and Benefits of Information-Centric Networking

Information-centric networking (ICN) proposes a paradigm change by routing messages based on names instead of endpoint identifiers as in host-based approaches. By this, ICN aims to better deal with the expected huge amount of traffic of the Future Internet. In particular, ICN can address the following aspects.

- **Scalability:** Because every exchanged message has a name, similar requests can be identified and aggregated. Thus, in case of multiple identical requests, e.g., for a live video broadcast, only one request needs to be forwarded to a content source avoiding unintentional Denial-of-Service (DoS) attacks by many legitimate users on a server. Furthermore, content can be cached automatically in every node to serve future requests reducing traffic load at core servers. The caching mechanisms can consider popularity of content at the network level, which may be measured by requests in the same content. Therefore, ICN can react to popularity dynamics and does not require pre-planning or application-specific distribution mechanisms like Content Distribution Networks (CDNs).

- **Mobility Support:** Because messages are not routed between endpoints but based on names, requests can be naturally satisfied by the closest content sources supporting seamless consumer mobility. Furthermore, ICN can also support data mobility. For example, when trying to access a bookmark in the browser after years, it may be unsuccessful (broken link) because the content moved to another server and got a new URL. When requesting content via name, the content can still be found assuming the forwarding tables have been updated in the meantime (but this is a required mechanism in ICN).

- **Security:** Every ICN message is signed and (optionally) encrypted. By this, the content is secured rather than the channel between endpoints, which has several benefits. For example, content can be retrieved from any cache from any neighbor because authenticity can be assured via signature. Thus, mobility does not require any support by establishing new secure channels to new locations. Furthermore, most spam messages may be prevented because users can authenticate content and need to request (or subscribe to) content before receiving it. Thus, content exchange is only performed if a user is interested.

- **Context Awareness:** ICN does not follow a classical client-server communication model such that requests can be satisfied by local content sources, reducing global Internet traffic and increasing the relevance (and accuracy) of retrieved information. In particular, ICN can increase information granularity by disseminating information of local importance that may never be widely accessible through the Internet. Thus, ICN enables the development of novel applications that do not rely on central servers in the Internet but enable users to create and dynamically share content with friends and neighbors based on their context. By that, ICN can support opportunistic communication in remote areas or disaster scenarios where central communication infrastructures may be not available.

- **Communication Technologies:** ICN can support cellular technologies, WiFi, mobile ad-hoc networks and others at the same time because it focuses on

content. Changes in the underlying channel may be handled by routing strategies and forwarding abstractions based on connectivity and availability of the corresponding technology. Loop-free forwarding (through caching and request aggregation) may avoid redundant traffic.

- **Efficient Wireless Communication:** ICN can exploit the inherent broadcast capability of the wireless medium. By transmitting broadcast requests, a user can perform implicit content discovery (get Data if a content source is available and nothing otherwise). Automatic caching in wireless nodes is also useful in case of collisions in multi-hop communication because content may then be retrieved from intermediate caches (where the collision occurred) and may not need to be retransmitted over the entire end-to-end path. Thus, in case of frequent collisions (where retransmissions may collide as well), some content may still be received by requesters. Furthermore, for time critical applications, e.g., real-time video streams, retransmitted content from intermediate caches may still be received in time (because they do not have to be transmitted over the entire path) increasing the number of received frames and, thus, improving the perceived quality of a video stream.

Several ICN architectures have been proposed (see [31, 226] for an overview and comparison). In this work, we focus on the Named Data Networking (NDN) architecture [119], which has previously also been known as Content-Centric Networking (CCN, see Section 2.1). In particular, we investigate NDN in wireless and mobile networks.

1.2 Challenges and Problem Statements

Within this thesis, we investigated different topics and challenges in the context of wireless and mobile (multi-hop) environments. When doing this, we faced the following problems and challenges.

- **Evaluation Methodology, Tools and Software:** NDN removes the idea of an end-to-end communication channel by using content names as fundamental communication units. Thus, it does not matter from where content comes from and it can be cached everywhere in the network. What seems to be the main benefit of NDN, is also one of the biggest challenges, i.e., subsequent content downloads may not be independent of each other due to caching. Therefore, existing performance metrics such as throughput or bandwidth may not always be appropriate. For example, a short round-trip-time (RTT) may be caused by an uncongested high performing link or by a closely cached content copy (if another requester has retrieved the same content shortly before). There is not much consensus in the research community how to evaluate NDN, e.g., what scenarios and parameters to compare. In

addition, during the course of this thesis, the development of NDN (in particular the CCNx implementation) was a very dynamic process with new code updates almost every 1-2 months (see Section 2.1) and there was barely any documentation except from the source code.

- **Content Discovery:** NDN is based on a pull-based (request-response) communication scheme. Thus, before content can be retrieved, knowledge of available content names is required. In distributed environments without continuous connections to the Internet, the number of available content objects may be limited such that specific content (preferred content names) may not be available. Then, it is crucial to decide whether a user request can also be satisfied with one of the available content objects. In particular, the design of content namespaces to enable efficient content discovery is a fairly unexplored area.

- **Efficiency of NDN Broadcast Communication:** In wireless networks, the connectivity (neighbor nodes) of a mobile node may change frequently. Since NDN messages do not contain endpoint identifiers, a mobile node can retrieve content from any neighbor node via broadcast (given that a neighbor node holds desired content). However, the efficiency of NDN broadcast communication has not been studied. In particular, wireless MAC protocols are optimized for wireless unicast communication (e.g., rate adaptation mechanisms, reliability via acknowledgments, lower power consumption due to duty cycles) and provide only reduced data rates during wireless broadcast communication. Moreover, broadcast requests address multiple nodes at the same time, which may increase the collision probability in case of multiple content sources.

- **Forwarding in Wireless Multi-hop Environments:** If a requester can not meet a content source directly, wireless multi-hop routing protocols are required to retrieve content. To enable multi-hop communication, forwarding entries need to be configured in intermediate nodes. The question is how to configure forwarding entries and define forwarding mechanisms in an efficient way. For example, content advertisements may expire quickly in case of mobility and regular updates may increase the control overhead, which may not be justified for unpopular content that is never requested. Flooding mechanisms (multi-hop broadcast communication) may be more robust to topology changes, but it may also result in an increased message overhead due to redundant (unnecessary) message transmissions.

- **Delay-tolerant NDN Communication:** NDN is based on symmetric Interest-Data forwarding paths. In delay-tolerant networks, there may be considerable delays between the transmission of Interests and the reception of corresponding Data messages. For example, if there is no continuous end-to-end path between requester and content source, Interests may expire and may

never reach a content source. Furthermore, due to node mobility, nodes that have forwarded Interests may have moved away and can not forward returned Data objects anymore, i.e., breaking the symmetry in NDN forwarding paths.

- **Caching:** NDN caches are considered as short-term storage to avoid retransmissions over the entire path to a content source in case of collisions or to synchronize multiple concurrent requesters of the same content. In the latter case, caches can consolidate even slightly time shifted requests to reduce network traffic. Since NDN caches need to support line-speed, they are implemented in main memory, which has a limited (small) size and is cleared in case of power outages. Therefore, short-term caching may not meet the requirements for delay-tolerant networking. Furthermore, small caches may result in frequent cache replacements, i.e., resulting in lower cache hit rates, in case of frequent transmissions of large content objects.

1.3 Contributions

1.3.1 Overview

Our main contributions are presented in Part II and III of this thesis. In Part II, we investigate opportunistic information-centric one-hop communication. In this part, requester nodes explore their direct neighborhood for desired content without requiring explicit device discovery. A requester needs to come into a content source's transmission range to retrieve the content. Thus, not every content object is available at all times and there are disruption periods where requesters may not be in range of the content source. Our contributions in Part II are as follows.

- **Content Discovery:** We describe three content discovery algorithms to explore available content objects in opportunistic environments without central name directories (cf. Subsection 1.3.2).

- **Opportunistic Content Retrieval:** We develop an opportunistic content retrieval application that persistently stores partially received content and enables requesters to resume disrupted content retrievals in case of short contact times to content sources (cf. Subsection 1.3.3). Furthermore, to understand and potentially improve wireless NDN communication, we study the efficiency of NDN broadcast communication.

- **Dynamic Unicast:** We develop a mechanism called Dynamic Unicast to increase efficiency of content retrieval from one-hop neighbors. The approach is based on implicit content discovery via broadcast requests and registers direct forwarding entries to discovered content sources for efficient content retrieval (cf. Subsection 1.3.4).

- **Raptor Codes Enabled Named Data Networking:** We develop a Data encoding scheme based on Raptor codes to reduce duplicate transmissions during NDN broadcast communications by increasing diversity of wireless Data transmissions (cf. Subsection 1.3.5).

Part III describes mechanisms to enable information-centric multi-hop communication. This is required if requesters may never see a content source directly and could, therefore, not retrieve content via one-hop communication even if they waited infinitely long. In Part III we present the following contributions.

- **Multi-hop Communication via Preferred Forwarders:** We describe a broadcast forwarding protocol that configures forwarding entries based on overheard Data prefixes and identifies preferred forwarders among intermediate nodes to reduce redundant message transmissions (cf. Subsection 1.3.6).

- **Dynamic Unicast Routing for Wireless Multi-hop Networks:** We describe a wireless multi-hop extension for Dynamic Unicast and present different forwarding strategies for reliable (mobile) multi-hop content retrieval (cf. Subsection 1.3.7).

- **Dynamic Retransmission Timers:** We analyze existing adaptive retransmission timers and describe new retransmission timers to increase throughput of information-centric wireless multi-hop communication in case of collisions (cf. Subsection 1.3.8).

- **Agent-based Content Retrieval for Delay-tolerant Networks:** We describe agent-based content retrieval to enable multi-hop content retrieval in case of intermittent connectivity (cf. Subsection 1.3.9). Because the approach does not require any modifications to ICN messages, it can be combined with wireless multi-hop routing enabling operation in arbitrary environments.

- **Persistent Caching:** We present a persistent caching extension that can supplement NDN short-term caching at each node to increase the availability of popular delay-tolerant content near requesters (cf. Subsection 1.3.10).

We gained insights how to evaluate NDN networks by using several evaluation platforms and investigating different performance parameters (cf. Chapter 3). In particular, we have implemented an ICN framework for OMNeT++ [205] that allowed us to thoroughly analyze NDN in mobile wireless networks. In addition, we extended the direct code execution framework (NS3-DCE [16]) of the network simulator NS3 [17] such that we can evaluate real NDN implementations based on CCNx [27] with different parameters in parallel on multiple computing nodes of a Linux Cluster [200].

1.3.2 Content Discovery in Wireless Information-Centric Networks

If a central communication infrastructure is not available because it is broken, e.g., natural disasters such as floodings or earthquakes, or it was not available in the first place, e.g., in remote areas, requesters need to learn what content is available before they can request content. Information-centric opportunistic communication does not require device discovery (because the location of content is irrelevant) but can exploit implicit content discovery via broadcast requests to learn whether a certain content object is available or not. If a content request is not answered, the content object is not available, which means that no neighbor devices can provide the content.

In Chapter 4, we describe three different content discovery algorithms, namely Regular Interest Discovery (RID), Enumeration Request Discovery (ERD) and Leaves First Discovery (LFD). RID exploits implicit content discovery by always requesting the first segment of a content object before looking for other content. ERD is similar to the service discovery protocol for DNS (DNS-SD): it uses enumeration requests, which request next-level name components for a certain prefix without requesting the actual content. Thus, individual ERD Data messages are smaller than for RID. LFD is a combination of RID and ERD using RID to reach the leaves of a name tree and then ERD at the leaf level. We evaluate all three algorithms in various scenarios with different content and node densities as well as with different namespace structures, i.e., a flat, hierarchical and mixed namespaces. Basic evaluations of RID and ERD have been published in [52, 201] and more detailed evaluations including LFD have been published in [50, 182]. Evaluations show that algorithms designed for flat namespaces (ERD) do not perform well in hierarchical namespaces. In particular, if the namespace structure is unknown, LFD should be preferred because it performs overall best in all scenarios.

1.3.3 Opportunistic Content Retrieval with Resume Capability

If contact durations to content sources are short, only partial content may be retrieved at once. In case of long intercontact times, already received content objects may not be found in caches anymore due to their limited space. In this case, content downloads need to be restarted from the beginning resulting in redundant Data transmissions.

In Chapter 5, we describe a requester application for opportunistic networks that persistently stores meta information in case of disrupted content retrievals such that incomplete content retrievals can be resumed if connectivity is regained. We deploy our source code on wireless mesh nodes and show that processing and data overhead is negligible. Furthermore, we evaluate wireless throughput as well as power consumption during broadcast and unicast communication. We find that throughput with unicast communication is significantly larger than with broadcast communication. Furthermore, listener nodes that are not interested in the content have an increased power consumption of more than 20% when they receive un-

desired broadcast messages. These properties (and the fact that unicast forwarding entries can not be configured statically in opportunistic networks) build the motivation of Dynamic Unicast (see Subsection 1.3.4 or Chapter 6). In addition, we have observed that broadcast data rates can be significantly increased with shorter Interest lifetimes because requesters can react quicker to collisions. This observation forms the basic motivation of our adaptive Interest lifetime algorithms in Chapter 10. Contributions of Chapter 5 have been published in [48, 176, 220].

1.3.4 Dynamic Transmission Modes to Support Opportunistic and Information-Centric One-hop Communication

Information-centric networking enables users to retrieve content from the closest content source and it is not necessary to keep connectivity to (or find) a specific host. By this, ICN can support context-specific content downloads, e.g., retrieving temperature from a close-by sensor instead of a server several kilometers away. Since NDN messages do not maintain source or destination addresses, most wireless NDN works use broadcast communication to make communication resilient to individual node mobility and enable ubiquitous caching in every node. However, broadcast requests may trigger transmission from multiple nodes increasing the probability for duplicate Data transmissions and collisions. With ubiquitous caching, every node that overhears Data transmissions automatically becomes a content source increasing content density and collision probability even more.

Chapter 6 shows by simulations that broadcast Data transmissions need to be delayed to reduce collisions and duplicate transmissions whereas unicast communication does not require such delays. Then, we describe Dynamic Unicast, published in [42], where requests are only transmitted via broadcast until a content source has been found among one-hop neighbors. Afterwards, a direct (unicast) forwarding face is configured to the same content source until it becomes unavailable. We observe that for only a few requesters, ubiquitous caching is often not required because content may not be requested anymore. However, for multiple requesters, content density is also high with Dynamic Unicast since each requester still caches retrieved content. Simulations confirm that broadcast is not as efficient as expected in case of many requesters and Dynamic Unicast results even in faster transmissions including fewer or only slightly more message transmissions.

1.3.5 RC-NDN: Raptor Codes Enabled Named Data Networking

Information-centric networking is not optimized for wireless broadcast communication because the same copies of content are (re-)transmitted along the same communication paths such that shared resources on the wireless medium are not efficiently used. There have been some research efforts [147, 225, 140, 212, 60, 172] that introduce network coding in ICN architectures to address inefficiencies in ICN data delivery. However, these approaches do not consider the implications of the proposed modifications as well as the limitations of information-centric network-

ing (see Subsection 2.7.2). In particular, if packets are re-encoded at intermediate nodes, new (encoded) content would be created, requiring new signatures and consequently new trust models.

In Chapter 7, we describe RC-NDN, an approach that encodes Data (with Raptor Codes) only at content sources. Although potential gains may be smaller than when re-encoding messages at intermediate nodes, extensive simulations, which have been published in [51, 190], show that RC-NDN significantly outperforms original NDN in terms of content retrieval times and number of transmitted Interest and Data messages. Interestingly, the performance of RC-NDN increases with the number of requesters. Thus, RC-NDN is particularly appropriate for dense urban environments such as train stations or sport stadiums, where multiple requesters can benefit from each other when requesting popular content.

1.3.6 Information-Centric Wireless Multi-hop Communication via Preferred Forwarders

If content sources can not be found in one-hop distances, requests need to be forwarded towards content sources multiple hops away. In static wireless networks, prefixes can be configured with the help of routing protocols [216, 112] and redundant random searches [75]. However, in wireless community networks, node connectivity and content availability may change over time. Proactive periodic exchange and configuration of all possible prefixes may overload the network.

In Chapter 8, we describe a broadcast multi-hop scheme that configures overheard content prefixes in forwarding tables. If Interests are flooded, they may be forwarded by multiple nodes resulting in redundant transmissions. Our forwarding concept is based on preferred forwarders, which are responsible for Interest forwarding while non-preferred forwarders delay Interest forwarding and transmit Interests only later if they do not overhear the Data transmission. A simulation study, published in [43], shows that this approach is much more efficient in terms of duplicate transmissions and throughput than original NDN, where only Data messages are delayed. However, the approach may be susceptible to random mobility (resulting in larger forwarding delays) when preferred forwarders are regularly moving.

1.3.7 Dynamic Unicast for Wireless and Mobile Multi-hop Networks

Mobile ad-hoc networks (MANETs) have been investigated for more than two decades and diverse proactive and reactive routing protocols have been proposed [81, 122, 159, 158]. However, most existing MANET routing protocols establish paths between endpoints, which do not work well in case of frequent node failures or high mobility. Multi-path routing protocols [192] can increase robustness of end-to-end communication but they also increase communication overhead due to redundant paths (and individual paths are still prone to link failures).

In Chapter 9, we extend Dynamic Unicast for multi-hop communication and describe two Interest forwarding strategies. We have implemented the approach in CCNx and evaluate it with NS3-DCE. Extensive evaluations in mobile scenarios with multiple requesters and multiple content sources show that Dynamic Unicast can still support all NDN benefits, namely implicit content discovery, caching and scalability of wireless multi-hop communication such that only a fraction of requests needs to be forwarded to content sources. Furthermore, Dynamic Unicast results in shorter content retrieval times and fewer Data transmissions than broadcast for high and low content densities. We also implement and evaluate a Content Request Tracker, which keeps track of unicast transmissions and reverts to broadcast communication if a certain number of unicast requests from different neighbors is detected. Multi-hop Dynamic Unicast and Content Request Tracker have been published in [54, 221].

1.3.8 Adaptive Interest Lifetimes to Support Information-Centric Wireless Multi-hop Communication

Wireless multi-hop communication is error-prone because received and forwarded messages can collide. If collisions occur near destinations, retransmissions during host-based communication need to be performed over the entire path between requester and content source, hence, they experience a similar collision probability. In case of frequent collisions, retransmissions may consume most of the bandwidth and may drastically reduce throughput in the network. ICN can mitigate this problem because content is automatically cached in every node, thus, retransmissions need only to be performed over the last hop where the collision occurred. Interest retransmissions can be performed after their lifetimes have expired.

In Chapter 10, we evaluate existing retransmission timers from literature for information-centric wireless multi-hop communication and introduce two modified algorithms. Our evaluation study, published in [53, 210], shows that adaptive Interest lifetimes perform significantly better than fixed lifetimes. Our proposed algorithms perform better than existing algorithms in low traffic scenarios and perform similar in high traffic scenarios. In particular, we have observed that it is not required to double Interest lifetimes in case of timeouts because most timeouts are caused by high RTT variations or collisions. In both cases content may be found in caches of intermediate nodes such that slightly increased Interest lifetimes can often retrieve the subsequent Data messages without timeouts.

1.3.9 Agent-based Content Retrieval for Delay-tolerant Information-Centric Networks

In case of intermittent connectivity between requesters and content sources, symmetric Interest - Data message forwarding may not work because the network topology may have changed on the return path. There are approaches to enable delay-tolerant NDN [87, 146] but they modify ICN messages formats and message

processing procedures, which make them incompatible with multi-hop forwarding in connected networks. However, most communication environments may not be strictly classified into disrupted (delay-tolerant) and connected networks since connectivity may change based on location (urban vs. rural), daytime (rush hour vs. late at night) or season (summer vs. winter). Network protocols should, therefore, operate in isolated islands but should also benefit from communication infrastructures if they are available.

In Chapter 11 we describe agent-based content retrieval (ACR) where content retrieval can be delegated to agent nodes which retrieve content on behalf of requesters and deliver it back. We implement ACR in CCNx and compare it to Dynamic Unicast in various mobile scenarios with different node densities, node velocities and content sizes. While Dynamic Unicast is superior to ACR for high node densities, our evaluations show that ACR performs better than Dynamic Unicast for low and intermediate node densities where multi-hop forwarding may result in frequent disruptions. Furthermore, ACR is beneficial in case of large content objects and works well even under high mobility. Contributions presented in Chapter 11 have been published in [45, 34, 49, 177, 221].

1.3.10 Persistent Caching for Information-Centric Networks

One of the main benefits in NDN is caching because it decreases path lengths and reduces Data transmissions by storing content closer to requesters. Caches in NDN need to support line-speed, and are usually implemented in volatile memory, which is expensive, power hungry and only available in small capacities. Thus, NDN caches can not support delay-tolerant networking because content may be replaced quickly or may be deleted if the system is temporarily turned off. In addition, related traffic studies of social networking services [223, 175, 169] have shown that users in the same geographic area tend to access similar content because their social environment is similar. Thus, storing locally popular content persistently at edge nodes can significantly reduce network traffic.

In Chapter 12, we describe a persistent caching extension based on the repository implementation in CCNx. This enables popular delay-tolerant content to be stored persistently for a longer time, while real-time traffic is still only served from volatile regular caches (which support line-speed). Persistent caching supports agent-based content retrieval, but it can also be applied to network caches at edge routers, where it improves perceived performance of end users (due to lower content access latencies) and reduces operational costs for the network infrastructure (due to less backhaul traffic). We evaluate the validity of persistent caching via extensive experiments using YouTube and web server traffic models. The performance study has been published in [47, 46, 95].

1.4 Thesis Outline

Chapter 2 gives an overview over the most important topics and related works that are addressed in this thesis. We describe our evaluation methodology in Chapter 3 including the tools and parameters that were used in this thesis to evaluate NDN. In NDN, knowledge of available content names is required to efficiently retrieve content. Chapter 4 presents content discovery algorithms for opportunistic environments without central naming repositories. Then, Chapter 5 explores opportunistic content retrieval with resume operations in case of disruptions during ongoing content retrievals. To increase efficiency of opportunistic one-hop content retrieval, we present Dynamic Unicast in Chapter 6 and a Raptor Encoded Named Data Networking (RC-NDN) framework in Chapter 7. However, if content sources are not in one-hop distance to a requester, requests need to be forwarded via multi-hop communication. Chapter 8 describes a forwarding approach for multi-hop broadcast communication based on overhearing and preferred forwarders. As alternative to broadcast communication, Chapter 9 presents a Dynamic Unicast extension and forwarding strategies for mobile multi-hop communication. Then, Chapter 10 analyzes adaptive Interest lifetime algorithms for information-centric wireless multi-hop routing to reduce timeout periods in case of collisions or other transmission errors. Since multi-hop routing is not possible in case of intermittent connectivity, Chapter 11 presents agent-based content retrieval for delay-tolerant networks, which can be combined with multi-hop routing in case of high node densities. Finally, Chapter 12 presents a persistent caching concept for popular delay-tolerant content, which can be combined with regular short-term NDN caching, to reduce network traffic and enable delay-tolerant networking via agents. We conclude our work by highlighting our major findings in Chapter 13 and describe challenges that can be addressed as future work in Chapter 14.

Chapter 2

Related Work

2.1 Information-Centric Networking

Several Information-Centric Networking architectures have been proposed in the last decade (see [31, 226] for an overview). Among those, the Named Data Networking (NDN) and Content-Centric Networking (CCN) architectures have become very popular due to their open source reference implementations. In Subsection 2.1.1, we describe basic concepts and data structures of CCN/NDN. As we show in Subsection 2.1.2, NDN and CCN are closely related to each other. Hence, for the sake of simplicity (see Section 3.1 for more information), we use the term *NDN* in remainder of this thesis.

2.1.1 Named Data Networking (NDN)

In this subsection, we give an overview of NDN node roles, data structures and message processing procedures as well as namespace structures and message headers that are relevant in the context of this thesis.

NDN Node Roles

Routing in NDN is based on names and not endpoint identifiers. In general, there are three different node roles in NDN.

1. *Consumers* or *Requesters* send Interests for content.
2. *Producers*, *Publishers* or *Content Sources* have created content and can reply Interests with Data messages.
3. *Routers* or *Forwarders* receive messages and may forward them towards a content source. In some works, we also refer to them as *Listener nodes* because they may overhear and forward broadcast messages from other nodes but they do not request content themselves, i.e., no messages originate from them. Listener nodes can also reply Data messages directly without forwarding Interests if they have matching content in their caches.

Content is organized in segments (also called chunks). Every segment, which is included in a Data message, has a unique name and is signed by the publisher who created the content. NDN is a pull-based ICN architecture: requesters need to transmit Interest messages for all segments of a content object to retrieve the complete content object. Interests may be answered by content sources directly (if they are in direct transmission range) or they are forwarded via forwarders towards content sources. Then, Data is returned on the reverse path. Since NDN does not require name resolution procedures, it can be applied to mobile ad-hoc networks, where connectivity to central infrastructures may not always be available.

Data Structures for NDN Message Processing

The CCNx [27] project provides an implementation of the CCN/NDN concepts (see Subsection 2.1.2). In CCNx, all message processing is performed by the CCN daemon (CCND). Figure 2.1 gives an overview of the CCND, which can be found in every NDN node, i.e., requesters, content sources and forwarders. Links from the CCND to applications or other hosts are called *faces*. A CCND has the following three data structures:

1. The Forwarding Information Base (FIB) contains forwarding entries with *Data names* (or prefixes thereof) to direct Interests via *face out* towards content sources. To avoid loops, Interests are never forwarded on the same face from where they were received.

2. The Pending Interest Table (PIT) stores unsatisfied forwarded Interests together with the face on which they were received (*face in*) and forwarded (*face out*). PIT entries form a multicast tree such that each Interest is only forwarded once at a time (Interest lifetime) over a certain face.

3. The Content Store (CS) is used as cache in a NDN router storing received Data messages temporarily. It is considered as short-term storage to avoid retransmissions over the entire path to a content source and to synchronize multiple concurrent requests for the same content even if they are slightly time shifted.

NDN Message Processing

We describe NDN message processing with the help of Figure 2.1. When receiving an Interest, it is first checked whether the requested Data can be found in the CS (step 1). If this is the case, the Data message can be returned (step 2) without further propagating the Interest. If Data can not be found in the CS, the PIT is consulted (step 3). Existing PIT entries prevent forwarding of similar Interests over the same faces because the requests are already pending (Interest aggregation). A PIT entry is removed if matching Data comes back or if the Interest expires (based

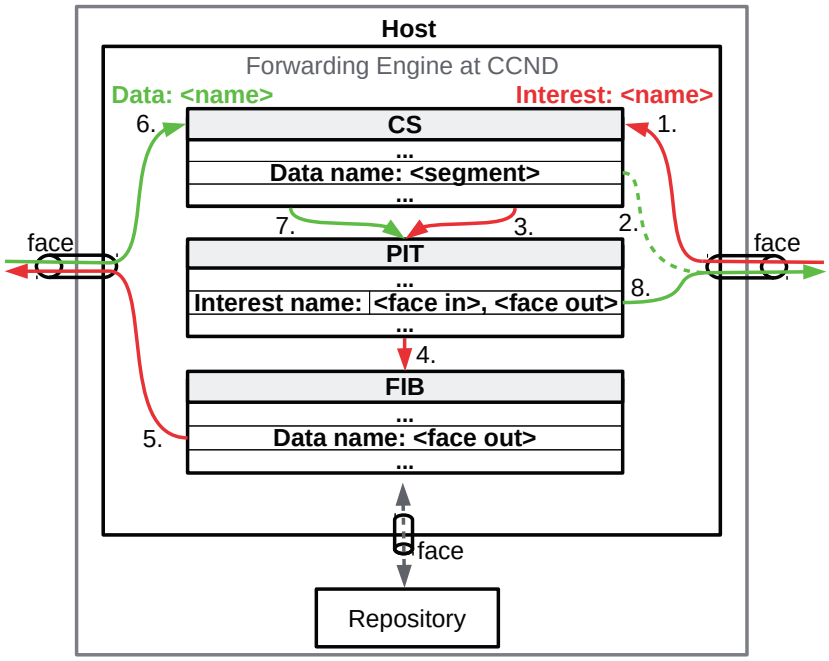

Host
Forwarding Engine at CCND
Data: <name>
Interest: <name>
6.
1.
CS
...
Data name:
...
7.
3.
2.
face
face
PIT
...
Interest name: <face in>, <face out>
...
8.
4.
FIB
...
5.
Data name: <face out>
...
face
Repository

Figure 2.1: NDN Message Processing.

on the Interest lifetime defined in the Interest header). If there is no PIT entry (step 4), the FIB defines over which faces Interests can be forwarded towards a content source (step 5). The matching of Interest names to FIB entries is based on longest-prefix matching similar to IP networks, i.e., FIB entries may contain complete Data names or only prefixes thereof. If there are no matching FIB entries, Interests are dropped. The number of Interests that can be concurrently transmitted by an application (before at least one Data message needs to be received) is defined by the *pipeline size*. After receiving a Data message in return to an Interest, it is stored in the CS (step 6). Based on recorded PIT information (step 7), Data messages can be forwarded on the reverse path towards requesters (step 8).

Prior to each Data transmission, a Data message is included and scheduled for transmission in a queue. To avoid duplicate Data transmissions, a broadcast delay, which is randomly selected within the interval $[DP, 3DP]$, is applied when scheduling broadcast transmissions. The data pause DP is a configurable parameter, which is set by default to 10ms. If a node overhears the transmission of the same Data message from another node, it can cancel a scheduled Data transmission (duplicate suppression).

Content sources can publish and persistently store content in repositories. Repositories are implemented as local applications, i.e., they receive and transmit messages via internal face from and to the CCND as illustrated in Figure 2.1. Repositories store all Data messages including headers and signatures in one file, i.e., the *repofile*. For fast access to content in the repofile, the repository keeps references in a B-tree.

Content Names

In NDN, *content names* have a hierarchical structure composed of multiple (arbitrary) name components $c_0, \cdots, c_n$, a file name $fname$, a $version$ number that labels different versions of the same content and a segment number s_n to mark each segment individually. The name structure looks as follows:

$$/c_0/\cdots/c_n/fname/version/s_n$$

To ensure globally unique names and support routing, content names may be aggregated by publisher specific prefixes similar to DNS names. There are no restrictions on content names and they may be selected arbitrarily. In addition, hierarchical names may not indicate the location of content objects, i.e., content from the same publisher may be downloaded and provided by different mobile hosts.

Header Fields in NDN Messages

Interest messages contain several header fields [5], which are considered in the forwarding and matching process. The *Interest Lifetime* determines how long an Interest stays in the PIT. Since no Interests are forwarded for existing PIT entries, the Interest lifetime has a direct impact on the time when Interests can be retransmitted. The *Scope* (if present) limits forwarding to the local host or neighboring hosts. The *AnswerOriginKind* determines whether Data may be retrieved from caches or needs to be retrieved from persistent storage at a repository. Furthermore, *Exclude filters* can be used when requesting content with a *name_prefix*, i.e., longest-prefix matching of names in Interest and Data messages, to indicate already known (or unwanted) next-level name components */name_prefix/components/*.

Every Data message [4] contains a signature from the publisher. Furthermore, the *freshnessSeconds* value in the Data header specifies the lifetime of a segment in the CS after its reception (before it gets stale). A final bit in the Data header marks the last segment of a content object.

2.1.2 Evolution of CCN and NDN projects

Figure 2.2 gives an overview of CCN and NDN evolution. CCN has been presented at CoNEXT in 2009 [119] and at the same time PARC has released the open source reference implementation CCNx [27]. The NDN project [13] is an NSF-funded Future Internet project that has started one year later and was based on PARC's CCNx source code. At the end of the three-years project in August 2013, NDN and CCN have separated because of different research agendas. While PARC geared towards a closed source commercial solution, NDN wanted to continue the project as open source research project.

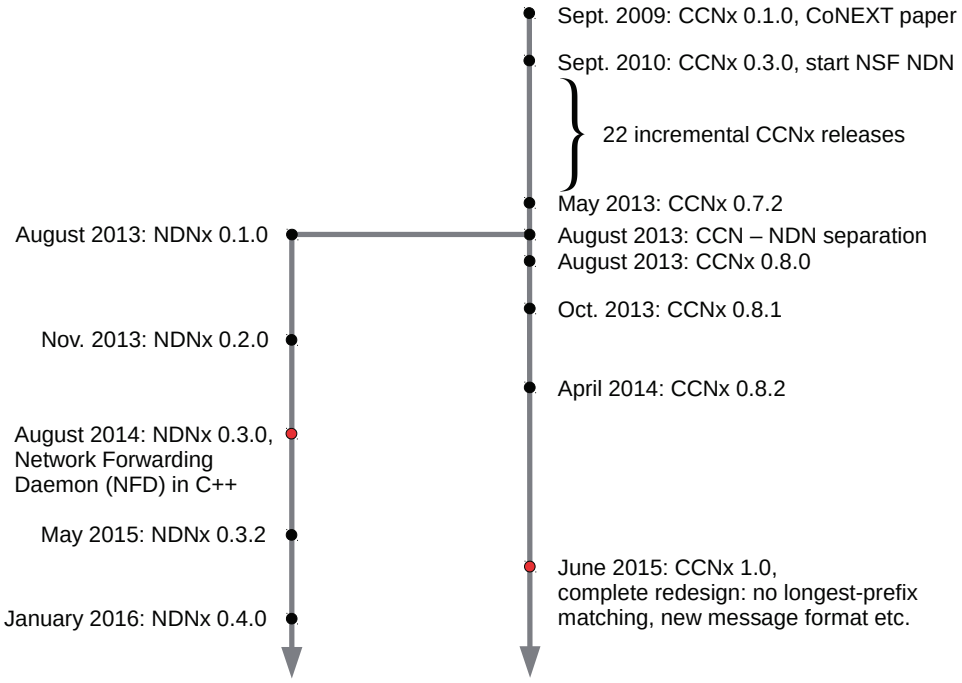


Figure 2.2: Evolution of CCN and NDN Projects.

CCNx

The CCNx [27] project provided the first implementations of Content-Centric Networking (CCN) [119] and was developed by PARC. The CCNx source code featured a C-implementation of the CCN daemon (CCND), which is responsible for message processing and forwarding, as well as application libraries implemented in Java and C. Until version 0.8.2, CCNx was developed as open source project under the GNU GPL license, which enabled the growth of a large scientific community in Content-Centric Networking.

In June 2015, the source code of CCNx 1.0 has been made available under a free evaluation license for educational institutions and a commercial evaluation license for commercial developers. However, since January 2016, the source code is again available for everyone under simplified licensing terms [19]. CCNx 1.0 has been completely rewritten featuring a different forwarding engine and different message formats. For example, longest-prefix matching of Interest and Data messages (Interests can retrieve Data messages from caches or repositories even if Interest names are only prefixes of Data names) as well as Interest selectors, e.g., Exclude filters (see Subsection 2.1.1), are no longer supported. Instead, names in Interest and Data messages need to match exactly [150].

NDNx

NDN and CCN were closely related to each other when NDN was based on CCNx. In August 2013, CCN and NDN separated and since then, NDNx [15] refers to the codebase of the NDN project. However, the first NDNx 0.1 source code was mainly a fork of CCNx 0.7.2 (with renamed namespace from ccnx to ndnx) and NDNx 0.2 was a continuation of CCNx including features developed within the NDN project. In August 2014 NDNx 0.3 has been published featuring a new Network Forwarding Daemon (NFD) implemented in C++ (equivalent to CCND in CCNx). With NDNx 0.3, the last ties to CCNx have been removed. The current NDNx 0.4.0 release is published under the GNU GPL and LGPL license. In NDNx 0.4.0, longest-prefix matching of Interest and Data messages and Exclude filters are still supported [14].

2.2 Service and Content Discovery

2.2.1 Device and Service Discovery

Service discovery enables users to detect devices that run specific services in local networks by service name (or service type) without knowing the host names of the devices. Service discovery supports zero-configuration networking, i.e., automatic configuration without manual operator intervention, by using multicast requests to discover available services. In most service discovery protocols such as the Service Location Protocol (SLP) [103], the Simple Service Discovery Protocol (SSDP) component of Universal Plug and Play (UPnP) [86], or the Web Services Dynamic Discovery (WS-Discovery) [25], devices answer service replies via unicast including a minimalistic service description and a pointer, e.g., an URL, to more detailed information. In large wired networks, devices can additionally register their services at centralized directories, e.g., directory agents [103], control points [86] or discovery proxies [25], to reduce multicast service requests. In contrast, DNS-based Service Discovery (DNS-SD) [74] for the multicast Domain Name System (mDNS) [73] uses not only multicast discovery but also multicast replies such that other hosts on the network can see the responses and keep their caches up to date.

Several service discovery algorithms have also been proposed for mobile ad hoc networks (see [207] for a survey). Due to changing connectivity patterns between mobile nodes, centralized directories (as proposed for wired networks) are not practical and represent a single point of failure. Most protocols are, therefore, either based on distributed directory-based service discovery (where directory nodes are organized in a backbone or cluster) or based on directory-less service discovery. In the latter case, service providers broadcast service advertisements (proactive mode) or requesters broadcast service requests (reactive mode). To lower the traffic load, the scope of broadcast can be limited to only a few number of hops or broadcasting can be avoided, e.g., via probabilistic forwarding or selective forwarding for known service providers. The literature study [207] concludes that cross-layer protocols that integrate service discovery with routing are more effi-

cient (in terms of message and energy overhead) than service discovery protocols running on top of routing protocols (see Section 2.3 for an overview of commonly used host-based routing protocols). Hence, this conclusion indicates a promising potential for ICN in mobile ad hoc communication, i.e., since messages are forwarded based on (content or service) names and not endpoint identifiers, service discovery could be integrated even more efficiently with information-centric than host-based routing protocols.

2.2.2 Relation between Content Discovery and ICN Routing Protocols

Routing in ICN is equivalent to finding a content source for a given content name (or a service provider for a given service name). Consequently, there is no clear differentiation between routing and content discovery protocols in related ICN literature. Due to loop-free forwarding in NDN, Interests can be flooded or forwarded over multiple faces to find content sources [76, 75, 227]. An overview of current ICN routing protocols can be found in Section 2.1. However, routing protocols are essentially used to configure forwarding entries in the FIB in a transparent way, i.e., without users noticing it. In addition due to name aggregation in NDN, routing protocols may only configure content prefixes, which are enough for routing, but they do not provide information about content published under these prefixes. Some ICN content discovery protocols use Bloom Filters to exchange content availability information of local caches among neighbors [219, 133, 134]. However, since caches are only short-term storage and may contain only partial files, these approaches may result in a huge overhead. Furthermore, it is not possible to describe arbitrary content names with Bloom Filters but only content objects or segments of known content collections. Yet, in opportunistic scenarios with limited content availability, content names may not be known a-priori and users may want to know what content can be locally retrieved.

In Chapter 4, we investigate opportunistic content discovery in NDN namespaces based on one-hop broadcast requests. Broadcast requests enable *implicit content discovery* in wireless networks since Data is returned in response to Interest messages only if it is available. In combination with longest-prefix matching of Interest and Data messages, implicit content discovery enables users to request content even before it is created or to find different resources, e.g., different temperature sensors or base stations, for the same name prefixes without knowing the complete name of the resources [213, 217, 52, 45]. Since the same content objects, i.e., copies, may be downloaded and stored at repositories and caches of multiple nodes, broadcast discovery requests are answered via broadcast replies to enable duplicate suppression (if another node transmits an identical content object).

2.3 Host-based Wireless Ad-hoc Networking

Mobile ad-hoc networks (MANETs) are composed of mobile nodes that create ad-hoc connections among themselves to enable communication in areas without communication infrastructures. MANETs have been an active research area for more than two decades and diverse proactive and reactive routing protocols have been proposed [81, 122, 159, 158]. Yet, most existing MANET routing protocols establish paths between endpoints and, therefore, do not work well in case of frequent node failures or high mobility. Although multi-path routing [192] has been proposed to increase robustness of end-to-end communication, it also increases communication overhead. Furthermore, individual paths are still prone to link failures. In this subsection, we give a brief overview of the most relevant host-based wireless routing protocols, which are still used in wireless networks and have influenced information-centric wireless routing protocols (see next subsection).

2.3.1 Proactive Routing

Optimized Link State Routing (OLSR) [121] and its successor OLSRv2 [80] are proactive routing protocols. Each node stores a routing table with all available destinations and the number of hops to reach them. To reduce broadcast transmissions, multipoint relays (MPR) are selected among one-hop neighbors. To do that, each node periodically transmits hello messages with neighbor information to its one-hop neighbors. MPR selectors are selected such that two-hop neighbors can be reached in the most efficient way. Because the MPR set is re-calculated in case of topology changes, OLSR/OLSRv2 target rather static and low mobility networks.

2.3.2 Reactive Routing

Dynamic Source Routing (DSR) [123] is a reactive source routing protocol, where senders include the complete forwarding node sequence in packet headers. To know which nodes to select, senders need to perform route discovery via broadcast. Forwarding nodes include their node IDs and updated hop distances in the packet before re-broadcasting it. When reply packets return on the reverse path via unicast, senders know the complete path to the destination and can remember it. Route maintenance mechanisms are applied to detect path breaks before discovering new routes to the same destinations.

Ad-hoc On-Demand Distance Vector Routing (AODV) [159] and its successor AODVv2 [158], which has been formerly named Dynamic MANET On-demand Routing Protocol (DYMO), are other reactive routing protocols. Similar to DSR, path discovery is performed via broadcast until a node has been found which knows the destination. Then, reply packets return on the reverse path via unicast. However, different from DSR, no information is included into the packets but soft states (source node, destination node, broadcast ID, sequence number) are established in all forwarding nodes. Forwarding paths are deleted automatically if they are not

used for a specific time. If links break, route maintenance mechanisms are applied to discover new routes to the same destinations.

2.3.3 Geographic Routing

Geographic routing protocols forward messages based on location information, e.g., GPS or other localization techniques, towards the geographic location of a destination. Examples of such protocols are the Greedy Perimeter Stateless Routing (GPSR) [124], where packets are forwarded to the closest neighbor towards a destination, the Distance Routing Effect Algorithm for Mobility (DREAM) [57], where packets are forwarded in the direction of an expected region (where the destination is assumed), or Terminodes routing [59], which uses greedy geographic routing to cover long distances towards a destination region and proactive routing (based on local routing tables) within the destination region. Most routing protocols require periodic beacons that include a node's geographic location but there are also approaches without beaconing, e.g., BLR [104].

Geographic routing protocols are in general robust to topology changes, but they require additional information, i.e., each node needs to know its own location and the source needs to know the location of the destination. In general, obtaining accurate location information is not a trivial task, e.g., collected information may expire quickly in case of high mobility and reliable localization mechanisms are required for indoor and outdoor environments. Furthermore, position-based forwarding may result in an increased processing overhead because forwarding decisions (compare coordinates in each packet with own location) need to be performed at every node for every packet.

2.3.4 Data-Centric Routing

Routing in Wireless Sensor Networks (WSNs, see [33] for a survey) has similarities to CCN because routing is data-centric, i.e., data is requested based on names while node IDs of responding nodes are not important. A popular data-centric WSN routing protocol is Directed Diffusion (DD) [116]. In DD, a sink broadcasts interests for names to find sensors with matching data. The interests establish soft states in forwarding nodes such that data can travel at a low rate via unicast on the reverse path. In contrast to CCN, these soft states are not deleted after receiving a data packet but maintained to receive data at a specific rate until they expire. A sink can reinforce soft states positively (higher data rate) or negatively (lower data rate). Despite similarities to CCN, data-centric WSN routing protocols can not be used as general purpose MANET routing protocols. Since sensor networks are built for a specific purpose, e.g., monitoring the temperature, sensors can understand data values and aggregate them. In CCN, data aggregation is not possible by intermediate nodes because they i) may not understand the data and ii) would need to create and add new signatures when modifying data. Furthermore, WSNs target typically rather static scenarios with a sink node that collects rather redundant data

from multiple sensors, while MANETs can be highly mobile comprising multiple requesters and diverse content sources.

2.4 Information-Centric Routing Protocols

Information-centric networking (ICN) has been proposed to address security and scalability issues of current host-based communication. In contrast to host-based communication, requests are routed based on names towards content sources and are not based on endpoint identifiers. If the connectivity to a content source breaks, e.g., because the distance becomes too long, it is not required to establish a new path to the same source or perform handovers to connect to a new source. Instead, requesters can implicitly find content at sources or caches nearby. In this section, we give an overview of existing wired and wireless ICN routing protocols.

2.4.1 Wired Routing Protocols

Routing Protocols for Static Networks

OSPFN [216] and NLSR [112] are link-state routing protocols, which have been proposed for wired NDN networks. These protocols disseminate routing updates (with information of physical links between named routers, together with content prefixes) through the entire network. Each router can then build a complete network topology, i.e., containing all routers with associated content prefixes, to calculate multiple paths (faces) for each name prefix in the FIB. Since Interest aggregation and caching prevents routing loops, FIB entries can contain more than one face. Recently, a more efficient link state routing protocol (smaller communication and storage complexity than OSPFN and NLSR) called Link State Content Routing (LSCR) [106] has been proposed. Similar to OSPFN and NLSR, LSCR learns the complete network topology of named routers. However, instead of flooding publisher information of all publishers (as OSPFN and NLSR), LSCR propagates publisher information only from preferred publishers (king anchors), which can be selected in a distributed way at each router. Yet, OSPFN, NLSR and LSCR target rather static networks without frequent changes in topology and content availability to avoid many path updates.

Mobility Support

NDN supports consumer mobility because Interests are routed based on configured FIB entries from a requester's position to a content source [199, 44]. In contrast, source mobility is more complex because forwarding entries (location of the content source) may change. To address global source mobility, naming structures using proxy-based resolution (indirection) [131, 107] and *location/identity* splits [168, 108, 125] have been proposed. A locator is a prefix that defines the location

(*Point_of_Attachment*), where the device is attached. For example, by using temporal name components in the form /<*Point_of_Attachment*>/<*Device_Id*>/<*prefix*>/ content that is generated and located at a specific device can be moved together with the device to another location (*Point_of_Attachment*).

Optimized Routing to Off-Path Caches

Routes to original content sources can be configured via routing protocols for static networks as described above. To detect routes to alternative content sources closer to the requester, which may not be on a configured path to the original content source, optimizations are required. Bloom filters [133, 219] have been proposed to exchange caching information among neighbors. However, nodes would need to guarantee minimal cache validity times in order to be useful.

In the context of NDN routing, the concept of adaptive Interest forwarding [228] has been investigated. Since FIB entries may contain multiple forwarding faces for the same content prefix (without risking routing loops), requirements on individual forwarding faces can be relaxed, e.g., forwarding faces may only indicate locations where the content may be found with high probability (but without certainty). Based on successful Data retrievals, forwarding nodes can select which FIB face to use in the next transmission. Further work [227] has indicated that routing protocols are mainly used to bootstrap message forwarding in communication networks and probing (adaptive Interest forwarding) can be used to select the best performing configured face.

Following this concept, another approach [76] floods Interest for the most popular content (which may be stored in many nodes) to find new off-path content sources nearby and uses controlled forwarding based on configured FIB entries for less popular content. Due to flooding, the number of required FIB entries can be reduced at the expense of slightly more Interest transmissions. Further work [75] uses an exploration phase where Interests are forwarded over the shortest configured path to a permanent content copy and at the same time additionally over a randomly selected face to discover temporal content copies in caches. In the following exploitation phase, Interests are forwarded only over the best face (learned from exploration phase) until the performance over the selected face degrades. In this case, a new exploration phase is started.

Another approach [111] proposes a tracker-based caching scheme, where network nodes can register cached content at a tracker server. The network is divided into sub-systems and each sub-system has a separate tracker server. Routing is based on an indirection approach where Interests are first forwarded to the local tracker server to check if another node in the same sub-system has a cached Data copy. Only if no cached copy can be found, the Interest is further forwarded towards the original content source.

2.4.2 Wireless Routing Protocols

Different from wired networks, the topology of mobile ad-hoc networks may change dynamically. Analytical work [206] has shown that the cost of maintaining routing information in mobile networks may overwhelm the benefits of structured solutions. Thus, flooding may be beneficial in networks with high host churn.

Broadcast Protocols with Node Identifiers

Listen-First-Broadcast-Later (LFBL) [143] is the first on-demand MANET routing protocol for named data networking. Similar to DSR, LFBL adds additional information to all messages but in contrast to DSR only source/destination endpoints (node IDs) and hop counters are added instead of a complete node sequence. Similar to AODVv2, nodes that overhear transmissions can store distance information to endpoints locally. However, in contrast to existing MANET routing protocols, all messages are broadcasted. Forwarders delay re-broadcasting of messages based on the distance from previous senders, i.e., the longer the distance the shorter is the delay. E-CHANET [41] is a similar approach for pedestrian mobility. Similar to LFBL, all messages contain a source node ID of the requester, a destination node ID of the content provider and a hop count for the distance. The same mechanisms have also been used for Content-Centric Vehicular Networking (CCVN) [37].

However, when node IDs are added to both Interest and Data messages, the communication is not strictly information-centric anymore since node IDs of providers matter and provider handovers [41] are required. Furthermore, Interest aggregation for multiple requesters becomes more complex because of different node IDs and (potentially) different hop distances. In addition, the hop distance to content providers may be inaccurate in case of mobility and cached Data. If hop distances and node IDs need to be updated at every hop, they cannot be protected by the publisher's signature making them susceptible to attacks.

Broadcast Protocols without Node Identifiers

Wireless ICN communication without endpoint identifiers has been mainly investigated in the context of vehicular networks. It has been shown that name prefixes can be used to quickly find resources that provide certain content [213]. This is beneficial for high-speed mobility, when connectivity to a content source is short and topology discovery would take too much time. In particular, it has been shown [213] that NDN outperforms MobileIP for vehicular communication with high speeds because NDN requests can seamlessly retrieve content from any vehicle or base station while MobileIP requires handovers (delays until re-gaining connectivity) when moving from one base station to another. Since computers carried by vehicles may have significantly more storage and energy than mobile handheld devices, all NDN communication can be performed via broadcast and all overheard traffic can be cached, even unsolicited content for which no Interests have been transmitted (no matching PIT entries). Further work [217] has proposed a naming

design for mobile Data collection (without evaluations). All Interests are transmitted via broadcast such that they can be satisfied by neighboring nodes. Data mules may overhear and cache broadcast communication for later redistribution to consumers. Several mechanisms have been proposed to enable efficient multi-hop broadcast communication [214]. A collision avoidance timer delays message forwarding randomly to avoid collisions and duplicate transmissions. A pushing timer delays messages based on a node's distance from the content source enabling nodes farther away to re-broadcast packets earlier to make more progress. However, if multiple vehicles have a similar distance from a content source such as in dense environments, pushing timers, i.e., intervals of potential forwarding delay values, may need to be either large (to reduce the probability that multiple vehicles send a packet at the same time) or they may result in many duplicate transmissions. Therefore, we present a different approach for multi-hop broadcast communication based on preferred forwarders in Chapter 8. Intermediate nodes that overhear a Data transmission from other nodes slightly before their own transmission conclude that they are not required for the transmission (non-preferred forwarders) and delay subsequent Interest transmissions more, i.e., they increase an Interest Forwarding Delay (IFD), while other nodes assume that they are preferred and slightly decrease their IFDs. This simple strategy enables intermediate nodes to use initially large IFDs to prevent duplicate transmissions but the throughput increases with time when preferred and non-preferred forwarders have been identified, i.e., IFDs of preferred forwarders tend to zero. Preferred forwarders are particularly efficient in rather static networks, but efficiency may decrease in highly mobile networks when preferred forwarders may regularly move away.

Geographical Routing Protocols

To route messages quickly over large distances, geographic routing protocols are beneficial. While location information encoded in names [214, 217] may enable traffic information to be quickly forwarded further away from where it was originated, it would also require forwarding strategies to understand name semantics. To avoid complex forwarding strategies based on name semantics, a Link Adaptation Layer (LAL) [100] has been proposed to enable broadcast support and location information for raw 802.11 frames. LAL is a layer 2.5 protocol that appends location information as additional header to all transmitted MAC messages. By that, each node can compute the distance between the sender and itself and trigger forwarding timers based on it. Besides a GPS receiver, LAL requires every node to have a digital map to locate nodes on the map. Messages are forwarded with a delay based on the distance to the previous node: the shorter is the distance, the longer is the wait. If the same transmission is overheard from another node, LAL locates the forwarder on the digital map. If the other forwarder is closer to the destination, forwarding can be suppressed.

Navigo [101] is an extension for geographic routing based on LAL. The main idea is to bind content names to a producer's geographic area and guide Interests

towards these areas, which are $200m \times 200m$ squares on a digital map. In a discovery phase, requesters send Interests in all directions to find a content source. Responders attach their geographic area in returned data packets such that future requests can be directed towards the same area. LAL has been extended to map geographic areas to faces (Geo Faces). The FIB can then store name mappings to Geo Faces.

Navigo provides similar advantages (robustness) and disadvantages (accurate localization, processing overhead) as regular geographic routing protocols (see Subsection 2.3.3). In addition, Navigo supports quick content retrieval from the nearest content source due to ICN broadcast communication. However, since broadcast requests address multiple nodes at the same time, broadcast delays are required to enable overhearing of Data transmissions and suppress duplicates (see Subsection 2.1.1). With ubiquitous caching, content density increases drastically and even more nodes may be able to return Data in response to broadcast Interests. As we show in Chapter 6, a high content density may require large broadcast delays to limit duplicate Data transmissions. However, large broadcast delays decrease wireless throughput such that in case of short contacts to content sources (no continuous connectivity) content downloads may not be completed in time.

Broadcast Discovery - Unicast Content Retrieval

While broadcast delays are necessary for broadcast communication, they are not required during unicast transmissions. Therefore, it may be beneficial to avoid broadcast communication whenever unicast communication is possible. Reactive Optimistic Name-based Routing (RONR) [55] is a routing protocol that has been developed for static sensor networks. RONR is based on flooding to find a content source and dynamically configuring unicast faces in the FIB on the reverse path. After a discovery phase (broadcast), subsequent Data can be retrieved from the content source via unicast. In static IoT scenarios, RONR can reduce the number of radio transmissions by 50% compared to broadcast and, thus, save energy on resource constrained devices. However, RONR has not been designed for and evaluated in mobile ad-hoc networks with changing connectivity patterns that require dynamic Interest forwarding strategies and FIB update mechanisms.

In Chapter 6, we present Dynamic Unicast (DU), which has been developed independently at the same time as RONR. Similar to RONR, DU creates dynamic unicast faces to content sources after a broadcast discovery, but different from RONR, connectivity to neighbor nodes can change dynamically with DU. We show that DU is not only beneficial in static networks but also for opportunistic one-hop content retrieval because it increases throughput during (short) connectivity periods to content sources, i.e., no broadcast delays, and reduces the number of duplicate Data transmissions. In Chapter 9, we show that DU can be extended to work efficiently for mobile multi-hop communication. In particular, we show that DU can still aggregate Interests and cache Data on multi-hop paths such that only a fraction of all messages reaches content sources. Furthermore, retransmissions do not need

to be performed over the entire path but can be satisfied from the nearest cache. The described mechanisms do not require localization mechanisms, such as GPS coordinates, and can be used without modifications of NDN Interest and Data messages. For multi-hop DU, we do not limit Interest flooding to a certain number of hops (to find a content source) because hop counters were not available in NDN Interest messages [5, 14], but are available only in CCNx 1.0 [149]. Yet, a recent study [215] shows that scoped flooding, e.g., limiting flooding to only 3 hops, may be more efficient than unlimited flooding (gain vs. message overhead). This observation is orthogonal to our developments, i.e., by introducing hop counters in NDN Interest messages, scoped flooding can be easily integrated with multi-hop DU to further increase message efficiency.

2.5 Retransmission Timers for NDN

Interest lifetimes can be considered as retransmission timers because they define when Interests can be retransmitted, i.e., only after previous Interests have expired and have been removed from the PIT (due to Interest aggregation). The default Interest lifetime in CCNx is 4 seconds, but there are approaches to apply adaptive Interest lifetimes for congestion control in ICN. Early works [154] use an Interest lifetime based on TCP's retransmission timeout (RTO) [36]. TCP's RTO uses an exponential moving average ($sRTT$) of current and past round trip times (RTTs) and the variance of RTT values (rttVAR) as follows.

$$\begin{aligned} sRTT &= 0.8 \times sRTT + 0.2 \times RTT \\ \text{rttVAR} &= 0.75 \times \text{rttVAR} + 0.25 \times abs(sRTT - RTT) \\ RTO &= sRTT + 4 \times \text{rttVAR} \end{aligned}$$

The RTO has a lower bound of 1s and in case of a timeout, it is doubled up to a maximum value.

The *TimeoutEstimator* [62, 61] is similar to TCP [117] but it uses a smaller gain for the RTT variance, i.e., 0.125 instead of 0.25. In addition to TCP, the retransmission timer is multiplied with a factor 2 to ensure that the $requestTO$, i.e., the Interest lifetime, is larger than RTT values.

$$\begin{aligned} err &= RTT - sRTT \\ sRTT &= sRTT + 0.125 \times err \\ \text{rttVAR} &= \text{rttVAR} + 0.125 \times (abs(err) - \text{rttVAR}) \\ RTO &= sRTT + 4 \times rttVAR \\ requestTO &= 2 \times RTO \end{aligned}$$

ICP [65] uses another approach based on the history of RTT measurements. The retransmission timer is based on the minimum value and the midrange of minimum and maximum RTT samples over the last 20 measurements as follows.

$$RTO = RTT_{min} + 0.5 \times (RTT_{max} - RTT_{min})$$

Exponential moving averages without variance considerations have been applied in CHoPCoP [231]. There, the variance of RTT samples is neglected but instead $sRTT$ is multiplied with "6" to ensure that Interest lifetimes are not a limiting factor. Then, routers need to explicitly signal congestion back to requesters if they are overloaded.

Recent works argue that RTT calculations need to consider the IDs of content sources that served the content due to caching [171] and multi-homing [66, 62] which may result in varying path delays. Similarly, provider IDs have been proposed [40] for multi-homing in wireless information-centric networks. However, individual Interest lifetimes in [40] are still based on TCP's RTO.

In Chapter 10, we evaluate existing retransmission timers in wireless multi-hop scenarios with one content source and multiple requesters. Furthermore, we describe two algorithms for information-centric wireless multi-hop communication, i.e., one that considers RTT variability and one that does not, and compare it to existing algorithms (described in this section). While most existing works are based on simplified implementations and simulations, we have implemented all algorithms in CCNx and evaluated them by emulations not only in terms of content retrieval times but also message overhead.

2.6 Delay-Tolerant Communication

Delay-tolerant Networking (DTN) addresses communication in challenged networks with intermittent connectivity (large intercontact times between nodes and lack of instantaneous end-to-end paths). In some DTNs, e.g., for interplanetary communication, connectivity patterns are known and communication can be planned, while in extreme terrestrial environments, connectivity patterns are unpredictable and unknown. The latter case is often also denoted as opportunistic networking, a subset of delay-tolerant networking. The main goal of opportunistic networking is to exploit contact opportunities between users to support best-effort content and service interactions where fixed network infrastructure may not be available or only at a high cost.

2.6.1 Delay-Tolerant Architecture and the Bundle Protocol

The Delay-Tolerant Networking Architecture [68, 90] evolved from the Interplanetary Network (IPN) architecture [69], which proposed a store-and-forward overlay network to support Internet-like services across long interplanetary distances. The Bundle protocol (BP) [179] describes a delay-tolerant protocol stack to exchange bundle messages and support intermittent connectivity. A bundle is the core protocol data unit, which contains multiple blocks with meta data and application data. Bundles are large in size but they can be fragmented similar to IP fragmentation. Every node that runs the bundle protocol has a bundle protocol agent (BPA), one or more convergence layer adapters (CLA) and an application agent (AA). The BPA is

the core component of the bundle protocol: it offers the BP services, e.g., registrations in bundle endpoints (see below) or transmissions of bundles, and executes the procedures of the bundle protocol. CLAs send and receive bundles on behalf of the BPA. Several convergence layers (CLs) have been proposed including TCP-based protocol adaptations for DTNs [83] or datagram based protocols [127]. The AA uses the Bundle protocol to request or accept the delivery of application specific content. Bundle nodes can register in bundle endpoints, which are described by Uniform Resource Identifiers (URIs). Based on bundle endpoint registrations, the BPA sends and receives bundles and delivers it to the AA. Bundles are transmitted in bursts and are stored locally until the next forwarding opportunity arises. To reduce the burden of caching content, nodes can delegate responsibility for storage and retransmission to custodians on the path. Although the bundle protocol is push-based, there have been some efforts, e.g., the Bundle Protocol Query Extension Block [92], to enable applications to query storages of nodes along a path.

Delay-tolerant networking (DTN) and information-centric networking (ICN) have many similarities [91, 197] such in-network storage, late bindings of content to locations, data longevity and flexible routing. There are even structural similarities between the CCN/NDN and the BP architecture: both have a core component for message registrations and transmissions, i.e., called CCND in CCNx or BPA in BP, and both can run applications that communicate via the core component, i.e., applications in CCNx or application-specific elements of the AA in BP. However, the main difference between both concepts is that ICN focuses on named data in rather well-connected networks while DTN focuses on network contacts and named endpoints in disruptive environments with high delays.

2.6.2 Pocket Switched Networks (PSNs)

Pocket Switched Networking (PSN) [115, 173] addresses delay-tolerant networking on mobile devices, e.g., smart phones, to exploit storage capabilities and short-range data transfer protocols for content exchange. While the DTN architecture [68] is based on host-centric addressing using endpoint identifiers, PSNs use data-centric communication where mobile devices exchange content exploiting social characteristics of the humans carrying the devices. Haggle [191] describes a popular data-centric network architecture for opportunistic networks. Data is described by keywords and users have profiles, i.e., node descriptions, which are disseminated in the network. When the profile matches data on another user's device, the device pushes matching data to the owner of the profile. Data discovery and dissemination is triggered when a device meets a new neighbor or via periodic updates for existing neighbors. There is no multi-hop communication and routing is replaced by the mobility of the nodes. PodNet [136, 105] is a similar approach, where users transmit periodic hello beacons to detect neighbors before connecting to them. Content is then exchanged pairwise by a solicitation protocol. The successor project of Haggle, called SCAMPI [161], has developed a service-oriented platform for mobile and pervasive networks. It contains a communication subsys-

tem, which is responsible for the detection of neighboring peers and the exchange of messages. Direct peer sensing mechanisms are applied to discover peers and services within communication range based on IP multicast or static IP discovery. To discover nodes further away, the platform defines transitive peer discovery, where nodes exchange information about other nodes they have discovered.

The ongoing Umobile [24] project develops a universal mobile-centric and opportunistic communication architecture that combines Delay-Tolerant Networking (DTN) and Information-Centric Networking (ICN) principles in a common framework to meet the requirements of the Future Internet. This includes several aspects such as content discovery [215], service support in challenged networks [174], infrastructure support to detect and predict mobility patterns [142] or pervasive data sharing [148].

2.6.3 DTN Routing Protocols

Routing in DTNs [120] has been studied for more than a decade. Many DTN routing protocols [139] have been proposed such as Epidemic Routing [203], where a node copies its messages to all nodes that it encounters, Spray-and-Wait [186], which limits the number of forwarded copies, or prediction-based forwarding based on the history of past encounters [138]. In recent years, increasingly more DTN routing protocols rely on social characteristics [233] to improve message delivery. Based on neighbor discovery [153], nodes can create a contact graph to detect communities as well as extract centrality, similarity and friendship characteristics for more efficient forwarding. Nodes that regularly see each other are reliable message carriers because they belong to the same community while nodes that have long-lasting and regular contacts may be considered as friends, who may even share common interests [233]. However, mapping contacts to social relations is complex and DTN routing performance heavily depends on how the mapping (contact aggregation) is performed [114].

2.6.4 Delay-Tolerant Information-Centric Networking

By targeting named data rather than node endpoints, ICN is promising for DTN communication because requesters can quickly find and retrieve desired content if available at any neighboring device. CEDO [87] is a first approach to extend CCNx with DTN functionality. Interests stay in the PIT until they are satisfied. Whenever a contact is detected (through periodic hello beacons), a message that summarizes all pending Interests is transmitted. A receiver of such a message sends back all Data messages that it has in the cache. Regular Interest forwarding via FIB is disabled. Another approach [146] introduces the concept of logical faces for communication in disaster scenarios. It assumes that communities (location where content can be retrieved and from where Interests have originated) are static and mobile mules forward messages between communities. Logical faces are used to map communities to physical interfaces. Whenever mules reach the corresponding

communities, the logical faces become active and Data can be retrieved based on information in the PIT.

Both approaches [87, 146] keep Interest messages in the PIT until nodes encounter the desired content source or community. However, NDN content may contain multiple segments, which need to be requested individually and, typically, the requester does not know the content size until receiving the final segment. Thus, a requester does not know how many Interests are required to retrieve the content and there is no immediate feedback when content transfer is finished (delay-tolerant content retrieval via data mules). If too few Interests are forwarded by a requester, a data mule may not retrieve the complete content and may need to travel several times between communities. Too many Interests, however, result in inefficient PIT memory management. Since both approaches [87, 146] modify NDN message processing and message structures significantly, they are incompatible to regular (multi-hop) NDN protocols for well-connected networks.

In Chapter 11, we describe agent-based content retrieval (ACR). ACR is an information-centric delay-tolerant networking approach that does not require any modifications to regular NDN message processing because DTN support is provided by application modules above NDN layers. Requesters can delegate content retrieval to agent applications on neighbor devices, which move closer to content sources, can retrieve content and return it to requesters. Hence, ACR can exploit human mobility [142] or public transportation systems [126] to enable delay-tolerant (best-effort) communication. There are DTN studies [56] showing that by offloading delay-tolerant data onto cars of daily commuting traffic (e.g., to migrate large data portions between data centers), high cumulative bandwidths can be achieved, e.g., a cumulative bandwidth of 10 Gbps if 20% of all vehicles participate with 1TB of storage. In particular, if content retrieval does not have any strict time constraints, delay-tolerant communication may enable energy-efficient communication [204] even if multi-hop routing may be possible. ACR may support efficient opportunistic networking because it can exploit implicit content discovery via NDN broadcast requests and does not require periodic neighbor (device) discovery. In fact, from an information-centric perspective, it may be questionable whether device discovery is required at all because the existence of neighbor devices does not reveal any information about available content or the neighbor's ability and willingness to perform certain tasks. Instead, nodes can transmit request messages when they are interested in content or services, and neighbor nodes would only answer (become a contact) if they can satisfy the request.

ACR can be combined with regular NDN multi-hop communication because NDN message processing is not modified. This is important in many situations where DTN communication may be dynamically used only for a certain time, e.g., if central communication infrastructures are temporarily not available, or at a certain place, e.g., to cover remote areas. However, after a short time people may be using the infrastructure again, e.g., when arriving at home or at work. Then, they may want to resume incomplete downloads or upload collected information without converting and re-signing it.

2.7 Network Coding in Information-Centric Networks

2.7.1 Existing Coding Approaches

The potential of network coding in Information-centric networks has been first discussed in [147]. Random linear network coding has been used to encode multiple packets (chunks) together. Hence, instead of sending the original (uncombined) packets, senders transmit network coded packets over multiple paths. Network coding decreases the probability to send duplicates of the same packet and increases packet diversity in the network. Therefore, received packets have higher probability to be useful for decoding. Network coding requires additional header fields in both Interest and Data messages: Interest messages contain a flag indicating whether the response should be network coded and Data messages require an additional header with the coefficients used for encoding.

Motivated by [147], CodingCache [225], a caching extension with network coding, has been proposed. Simulations show that network coding can increase cache hit rate due to cache diversity. This is due to the re-encoding process (network coding), which is performed in intermediate nodes to improve packet diversity. Furthermore, Interests are forwarded uniformly at random to maximize the overall cache hit rate. An analytical study on network coding in NDN [140] proposes offline solutions that increase network efficiency by dynamically exploiting network-coded caching and multicasting. The solutions are examined in a butterfly network and the evaluations show that random linear coded caching and multicasting is sufficient for achieving minimum cost caching-aided multicast. Linear network coding has also been employed in a software-defined networking (SDN) based controller framework [212] for efficient cache management and content routing. Furthermore, a content aware delivery scheme for scalable video transmissions has been proposed [60]. The approach uses prioritized random linear network coding to account for different video layers that have unequal importance. Then, a rate allocation optimization problem is formulated and the authors show that their approach achieves a close to optimal performance. More recent work [172] has integrated a network coding framework into CCNx and re-designed the way how Interest and Data messages are processed. Evaluations in butterfly and selected PlanetLab topologies show notably shorter content retrieval times compared to original CCNx.

2.7.2 Limitations of Existing Coding Approaches

Although existing works [147, 225, 140, 212, 60, 172] seem to be very promising, they do not consider the implications of the proposed modifications on (and limitations of) Information-centric networks. All the approaches use additional headers for network coding and process the messages similar to regular network traffic. However, since the foundation of Information-centric networks are names, i.e., content objects should be uniquely identified and processed by their name, net-

work coding should also be reflected in the content name and not only by message flags. Otherwise, it might be possible that two content objects have the same name (with and without message flags), which would violate the "uniqueness" of names.

The matching process in NDN is deterministic: if the same Interest is retransmitted, the same segment will be retrieved. However, existing network coding approaches assume that during the matching process a random combination of chunks is encoded and returned. If matching is turned into a random process at the CCND [60, 172], retrieving missing pieces of information becomes more complex. Below we discuss three approaches to retrieve innovative (non-duplicate) Data with Interests containing only a prefix of the Data name.

1. The first approach is based on a deterministic CCND, i.e., an unmodified CCNx implementation. To avoid the retrieval of duplicate Data, additional information about already received chunks need to be included in Interest messages. Exclude filters have been proposed [5] for this purpose, but they have disadvantages. Specifically, the sizes of Interest messages would increase with the number of excluded packets and this would introduce additional delays for Interest transmission and matching (to potential Data). Furthermore, it would not be possible to transmit multiple Interests with the same name prefix simultaneously via the same face because the PIT would prevent it (Interest aggregation), i.e., Exclude filters are not considered for forwarding but only in the matching process. If multiple Interests with Exclude filters could still be transmitted, only one Data message that is not included in the Exclude filters could satisfy all Interests. Thus, Interests with Exclude filters need to follow a stop-and-wait strategy, i.e., Interest transmission after Data reception, and cannot use pipelining.

2. The second approach [60] modifies NDN message processing at the CCND and integrates Bloom Filters in Interest and Data messages indicating the client IDs that are interested in the content. Thus, the design is not strictly information-centric anymore and the scheme may not be scalable for an increasing number of clients (to be included in Bloom Filters). In addition, the comparison of client IDs in Bloom Filters of Interest and Data messages is significantly more complex than prefix matching in original NDN. Since Interests from the same clients would look the same (same Bloom Filters), Interest processing in the PIT is modified to enable the forwarding of identical Interests from the same clients (like pipelining, but there is no upper limit) and prevent forwarding of identical Interests from different clients. Then, a returned Data message does only consume one of the Interests and the content source is responsible to transmit sufficiently innovative Data for the same clients (same Bloom Filter).

3. The third approach, called NetCodCCN [172], also modifies NDN message processing at the CCND. Instead of adding additional information in Interests, routers remember in the content store whether they have already

forwarded a Data message or not. By this, a router can avoid sending the same Data twice over the same face. However, such an approach has limitations. First, the amount of state information in each router would increase significantly. Currently, only the PIT keeps state information for the duration of an Interest lifetime (at maximum), but still several concerns [209, 98] have been raised that this may overload a router's memory. Additional information for network coding would need to be stored for an even longer time than an Interest lifetime. In particular, the decision how long information should be kept is not straightforward. Second, the content store has a limited size and content may be replaced frequently. However, after the removal of a Data message, also all state information would be removed such that duplicate Data transmissions can not be avoided. Hence, the content store would need to guarantee a minimum lifetime for each network coded Data message (which may not always be possible due to the content store's limited size). Third, if all knowledge is stored in routers, dynamic networking and consumer mobility can not be supported. For example, if consumers move to a new location, information at the new router may not be accurate and information from the old router cannot be used anymore. NetCodCCN enables the transmission of multiple concurrent Interests but there is no upper limit (maximum pipeline size) and the decision whether to forward an Interest is more complex than in original NDN, i.e., it is based on the expected number of innovative Data messages to be received within the Interest lifetime and the number of already forwarded Interest messages.

All three approaches result in an increased message processing overhead to i) match Interests with innovative Data, ii) transmit Interests and iii) re-encode Data at intermediate nodes. Furthermore, all network coding approaches [147, 225, 140, 212, 60, 172] do not consider security aspects of NDN. Since a content publisher signs every Data message, re-encoding by means of network coding is equivalent to generating new content with a new signature. Thus, re-encoding can not be performed transparently at intermediate nodes without requesters noticing it. If the generated network coded packets are not signed, malicious nodes could interfere with the communication by introducing single bogus messages. If nodes that apply network coding would sign messages, transitive trust models would be required.

2.7.3 Raptor Codes

Raptor codes [181] belong (like network codes) to the family of fountain codes [64]. However, unlike other fountain codes, Raptor codes have linear encoding and decoding times, hence, they are appealing for real-time data delivery in mobile and wireless networks that consist of low cost devices. Raptor codes are endowed with the rateless property that permits the generation of a potentially unlimited number of encoded packets from a given set of source packets K. Raptor codes perform similarly to Reed-Solomon codes and are characterized by a very small

communication overhead ϵ that is typically in the range $[3\%, 5\%]$. Hence, $K \cdot (1+\epsilon)$ packets should have been received before content can be successfully decoded. Obviously, Raptor codes have only probabilistic guarantees of decoding, i.e., when the number of received packets N exceeds K, Raptor codes can be decoded with a probability that increases with N. When Raptor codes are used in networks with diversity, e.g., overlay networks, the communication systems take advantage of enhanced packet diversity to cope with loss without requiring Automatic Repeat-Request (ARQ) messages. Although Raptor codes are traditionally applied end-to-end, they can also be combined with network codes [193] to further enhance the resilience of communication systems.

Raptor encoding has two phases. First, packets are encoded by a pre-coder, e.g., a LDPC [97] encoder, and then the resulting (pre-coded) packets are fed into a LT encoder that combines them by means of XOR coding to generate the Raptor encoded packets. The number (degree) of combined packets depends on a carefully designed degree distribution function. Raptor encoded packets are augmented with a header that can be the seed of a pseudorandom generator used for deciding which packets have been combined to generate an encoded packet. This header is required, as Raptor codes have an implicit coding structure, i.e., neither the number of encoded packets is known a priori nor the packets that have been encoded together.

The application of network coding in wireless communication systems [94] brings many benefits such as resilience to network dynamics, increased throughput and decreased content delivery times. In Chapter 7, we describe RC-NDN, which is an encoding scheme based on Raptor codes that reduces duplicate Data transmissions by increasing Data diversity during wireless NDN broadcast communication. RC-NDN does not require modifications to the CCND because basic NDN message processing is not modified. To avoid Exclude filters and, hence, enable pipelining (see Subsection 2.7.2), Interest messages contain IDs of encoded packets. Furthermore, in contrast to approaches 2 and 3 in Subsection 2.7.2, RC-NDN supports consumer mobility and does not require additional state information for forwarding at the CCND. However, to limit the processing overhead and avoid new signatures in intermediate nodes (in contrast to all network coding approaches [147, 225, 140, 212, 60, 172]), RC-NDN encodes packets only at content sources and does not perform re-encoding at intermediate nodes. Yet, RC-NDN shows significant improvements (shorter content retrieval times and fewer Data transmissions) over regular NDN in opportunistic one-hop broadcast communication.

2.8 Caching

Caching in information-centric networking has been subject to extensive research in recent years. In NDN, content is cached everywhere along the downloading path resulting in high cache redundancy. It has been shown that popular content tends to be cached at the leaves of the network [167] and, therefore, allocating more storage

resources to edge routers than core routers is beneficial in terms of performance [166] and energy consumption [135]. It has been shown [72] that caching less, e.g., caching content only in a subset of nodes along a path, can achieve more (better performance in terms of cache and server hit rates). To avoid redundant caching, several strategies have been proposed such as limiting the number of cached copies along the path to one [89], probabilistic caching based on distance from the content source [166], latency until a content object is retrieved [67], or applying network coding to ensure content diversity in caches [225, 212]. Other approaches are based on coordination for content deletion, e.g., pushing deleted content one-level upstream the caching hierarchy [137], adaptation of the number of cached chunks based on the content popularity [77] or redundancy elimination via fingerprints of content chunks [157].

Since the size of a content store (cache of the CCND in CCNx [27]) is very limited, cached content is only available for a rather short time. Hence, the content store can be considered as short-term cache to avoid retransmissions over the entire path to a content source in case of collisions or to synchronize multiple concurrent requesters of the same content. In the latter case, content stores can consolidate even slightly time shifted requests, depending on how long content is cached, to reduce network traffic. However, in some scenarios, short-term caching may not be enough and content needs to be stored persistently at the expense of slightly slower access times. In particular, if content should be available for a longer time, e.g., for delay-tolerant networking, custodian-based information sharing [118] or network caches to ensure high availability and performance similar to content distribution networks (CDNs), it should be stored persistently.

In Chapter 12, we investigate a persistent caching approach based on the repository implementation of CCNx 0.8.2. While every Data message in NDN needs to go through the content store (short-term cache), repositories can monitor the network traffic and store only a subset of content, e.g., non real-time traffic such as popular pictures or videos. Hence, persistent caching is orthogonal to existing ICN research on caching, because it can be combined with all (short-term) caching approaches. Our persistent caching approach is necessary for agent-based content retrieval (see Chapter 11) but it can also be used to store popular content in regional network caches. For example, YouTube traffic studies [235] indicate that there is no strong correlation between global and local popularity, i.e., videos of local interest may have higher local popularity. Furthermore, several studies of social networking services [223, 175, 169] report that user interests have significant homophily and locality characteristics. This means that people geographically close to each other may have similar trends in accessing content [218] because the users' social relationships and interests are often clustered around commonalities in their locations [78]. Hence, regional caches or servers that provide (regionally) popular content can significantly reduce network traffic because many requests can be satisfied in the same geographic region [235, 223, 175, 169].

Chapter 3

Evaluation Methodology

In this chapter, we first give an overview over the CCNx versions used within this thesis. Then, we describe evaluation tools that were used (and partly developed) within this thesis. Finally, we describe performance parameters for the evaluation.

3.1 Software used within this Thesis

The work in this thesis has been performed in the context of a SBFI funded project on Enhanced Mobile Ad hoc Communication with Content-Centric Networks (project no. C10.0139). We based our investigations on open source implementations of CCNx. During the course of the project between April 2011 and April 2014, 25 incremental CCNx releases have been published [8]. Therefore, evaluations in different sections of this thesis may be performed with different versions of CCNx. In Table 3.1, we give an overview of all CCNx versions used in this thesis, together with the most important changes for our work. A more complete list of changes can be found at `http://www.ccnx.org/releases/ccnx-0.8.2/NEWS`.

Version	Location	Information
CCNx 0.4.2	Section 4.3	first used version
CCNx 0.6.0	Chapter 5	**since CCNx 0.5.0**: Repository implementation in C, Repository Sync Protocol
CCNx 0.7.1	Section 11.4	**since CCNx 0.6.1**: new PIT data structure, improved handling of pending Interests, **CCNx 0.6.1 - 0.7.1:** improvements to Android CCNx services, improved Android stability.
CCNx 0.8.2	Sections 4.4, 11.5 and Chapters 9, 10, 12	**since CCNx 0.8.0:** improved content store implementation, AnswerOriginKind=2 for generated Interests, unsolicited content becomes stale immediately **since CCNx 0.8.2:** refactored strategy layer, forwarding strategies (default, parallel, load sharing), enforced deletion of unsolicited content

Table 3.1: CCNx Versions in Evaluations.

Our first work with CCNx 0.4.2 was using the CCNx Java libraries to perform discovery operations. However, the Java libraries were missing many essential features of the C libraries such as header modifications of CCNx Interest messages, e.g., Interest lifetimes could not be modified. Because all required missing features had to be implemented separately (e.g., see [201]), we used the C libraries for all subsequent CCNx implementations. Since CCNx 0.5.0, a new more efficient repository implementation in C has been made available (the previous implementation was in Java). This enabled us to evaluate CCNx 0.6.0 on wireless mesh nodes without requiring Java. Although the PIT structure has been improved later in CCNx 0.6.1, we expect that it would have only a minor impact on our evaluations (if any, the results would be slightly better), because we only considered one-hop broadcast and no multi-hop communication. For our evaluations on Android smart phones, we used CCNx 0.7.1, which incorporates already all important CCNx Android features and updates. All other CCNx evaluations were based on CCNx 0.8.2, which features an improved and cleaner content store implementation (compared to 0.7.1) including a refactored strategy layer with three different forwarding strategies. By that, custom forwarding strategies could be integrated easier than before. Furthermore, unsolicited content (no matching entry in the PIT) becomes stale immediately upon reception, i.e., $freshnessSeconds = 0$. Since CCNx 0.8.2, the deletion of unsolicited content in the content store is enforced, which has an impact on our content discovery algorithms (see Subsection 4.4.4).

For the sake of conformity (and because NDNx also experienced major modifications in recent years, see also Subsection 2.1.2), we decided not to switch to NDNx in August 2013 but continue with CCNx until 0.8.2. Until August 2014, NDNx was still based on the CCNx source code and even now, NDNx 0.4.0 and CCNx 0.8.2 are conceptually very similar.

Because CCNx 1.0 differs substantially from CCNx 0.8.2, some approaches described in this thesis, e.g., content discovery in Chapter 4 and agent delegation in Chapter 11 may not be possible anymore in CCNx 1.0 (due to lacking support for Exclude filters and longest-prefix matching for Interest and Data messages). Yet, these mechanisms can still be supported in NDNx 0.4.0. Thus, future developments for information-centric wireless networks should rather be based on NDNx.

3.2 Evaluation Methods

When we started our work on Named Data Networking in 2011, CCNx was new and no evaluation methodology had been established yet. During the course of this thesis, we have, therefore, performed evaluations in different ways, i.e., by measurements on real wireless devices, via simulations or emulations.

3.2.1 Simulations with OMNeT++

For repeatable and scalable evaluations in mobile and wireless networks, we have implemented a NDN framework in OMNeT++ [205]. Figure 3.1 gives an overview of the framework: it contains a CCNx Layer including all memory components such as the Content Store (CS), Pending Interest Table (PIT), Forwarding Information Base (FIB) as well as content queues with the same delays and flags as in CCNx [27]. Applications are implemented at an application layer and communicate with the CCNx layer via internal face. A basic component of the application layer is timeout management, i.e., reaction to Interest expirations, and pipelining. We are using our NDN framework over UDP/IP because this provides fragmentation support (for large NDN packets) without supporting reliability and congestion or flow control, i.e., retransmissions are only controlled by Interest lifetimes. There is a face adapter, which configures broadcast or unicast faces at startup via a CCNDC method from a text file (corresponding to CCNDC [63] in CCNx, which is used to manually configure the FIB at the CCND) or during runtime via Dynamic Unicast (only for the evaluations in Chapter 6). At the MAC layer, we use the IEEE 802.11g standard.

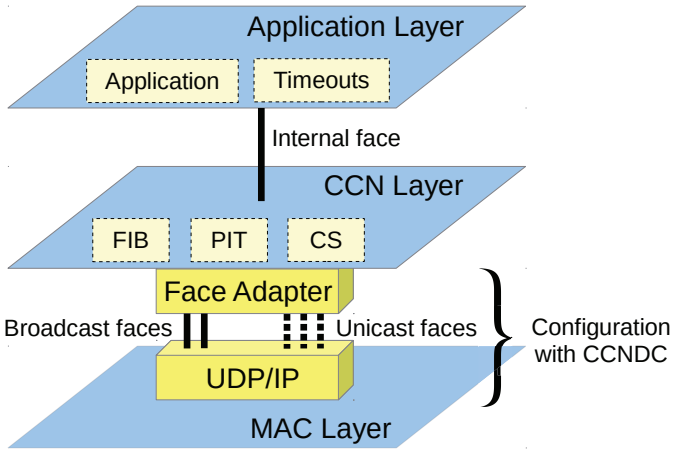


Figure 3.1: NDN Framework for OMNeT++.

The framework allows us to process and store information like CCNx except for signature calculations and verifications, which are omitted for simplicity. Thus, developed mechanisms in our OMNeT++ framework could be easily implemented in CCNx. We did not use ndnSIM [30], the Named Data Networking (NDN) simulator, for two reasons. First, ndnSIM was not yet available when we implemented our OMNeT++ framework. And even later, there were frequent and substantial modifications to ndnSIM making it difficult to track changes in both ndnSIM and CCNx. Second, ndnSIM used slightly different data structures and message processing procedures than CCNx, including new features that were developed within the NDN project (not available in CCNx). Hence, we used our OMNeT++ framework to stay close to CCNx in order to better understand CCNx

operation in wireless networks and facilitate subsequent CCNx implementations. The OMNeT++ framework is based on CCNx 0.4.2 and we performed updates until CCNx 0.6.0. We did not include modifications to the PIT (CCNx 0.6.1) and the content store (CCNx 0.8.0) since these modifications target performance gains in real systems, which would not be measurable in a discrete-event network simulator (where processing time is not considered). The simulation framework was used for evaluations in Chapters 6, 7 (Part II) and Chapter 8 (Part III).

3.2.2 Evaluations on Physical Devices in Experimental Testbeds

We have deployed CCNx on different physical devices to perform real-world measurements with CCNx. In this subsection, we list these devices.

PCEnginges Alix 3D2

We used PCEngines ALIX 3D2 [20] wireless mesh nodes, which have a 500MHz AMD Geode GPU, 256MB DDR DRAM and are equipped with two 802.11 miniPCI radio cards. The Alix boards were running on ADAM [188], a small-footprint embedded operating system. The used image was based on Linux kernel 3.2.18. Evaluations with wireless mesh nodes enabled us to evaluate the processing overhead of CCNx and its impact on achievable transmission times. Furthermore, we could evaluate the energy overhead of CCNx during unicast and broadcast communication with the Rigol DM3058 [22] digital multimeter (cf. Chapter 5).

Google Nexus 4

Smart phones may be a target platform for mobile NDN applications. Therefore, we have deployed and evaluated CCNx on LG Google Nexus 4 [12] smart phones running on Android 4.2.2. However, due to the costs of the devices, evaluations could only incorporate a few nodes. Furthermore, evaluations have only been performed in static scenarios because repeatable real-world evaluations with mobile nodes are very time-consuming and complex. Consequently, we used these smart phones only for basic evaluations in Section 11.4.

Physical Servers on Ubelix

Real-world evaluations need to be performed in real-time and evaluations can not be parallelized. Consequently, multiple evaluations (to obtain statistically relevant data) may take a lot of time. Therefore, evaluations that do not rely on wireless communication and require many resources (see Chapter 12) have been evaluated on physical servers of Ubelix [200], a Linux cluster at University of Bern. By that, we could run more than 300 evaluations in parallel on different nodes, which allowed us to evaluate many parameters in a short time.

3.2.3 Network Emulation

To enable repeatable wireless and mobile evaluations of CCNx (in contrast to simplified implementations in network simulators), we use network emulation tools that run CCNx on virtual hosts and simulate host mobility and wireless communication via network simulator. By that, external influences such as interferences can be avoided, enabling a more comprehensible protocol analysis. Furthermore, evaluations can be performed in simulated time, which is faster than real-time.

VirtualMesh

At first, we used VirtualMesh [187], which is a network emulation tool for wireless mesh nodes developed at University of Bern. VirtualMesh intercepts traffic generated by CCNx on virtualized hosts (XEN virtual machines) and redirects it to the network simulator OMNeT++ [205], which simulates wireless communication on the physical medium. To get statistically meaningful results (multiple evaluation runs per parameter configuration), we have implemented an evaluation framework for VirtualMesh [201] to automatically create XEN virtual machines, configure CCNx nodes and process evaluation results. However, since VirtualMesh does not scale well with increasing network size, we only used it for rather small networks. Consequently, we used VirtualMesh only for our first evaluations in Section 4.3.

NS3-DCE

Later, we used NS3-DCE 1.4 [16], which is a direct code execution framework for the NS3 network simulator [17]. In contrast to OMNeT++, NS3 represents packets as bit sequences in network byte order equivalent to real-world network packets. Therefore, evaluations with NS3-DCE do not require message translations (message encapsulations) into network simulator packets as with VirtualMesh, resulting in a lower processing (and wireless communication) overhead compared to VirtualMesh. While VirtualMesh emulates entire end-systems (including operating systems) accurately in full-fledged virtual machines, NS3-DCE considers only communication protocols, e.g., CCNx, by executing it for every node within virtual processes (which may contain multiple threads for separate tasks). Therefore, evaluations with NS3-DCE can be setup quicker and require fewer resources, i.e., they scale better, than evaluations with VirtualMesh.

The NS3-DCE framework uses absolute paths to load and store temporary files during an evaluation on secondary storage. If NS3-DCE runs only on one host (subsequent evaluation runs), this is not critical because evaluations do not interfere with each other. However, if evaluations need to run in parallel (with different seeds) on multiple computing nodes of a Linux cluster, modifications are required to load and store these files under different paths at computing nodes.

Figure 3.2 illustrates NS3-DCE evaluations on Ubelix [200] conceptually. Instead of subsequent evaluation runs on a single host, multiple evaluation runs can be started on the submit host to be executed in parallel on computing nodes. This

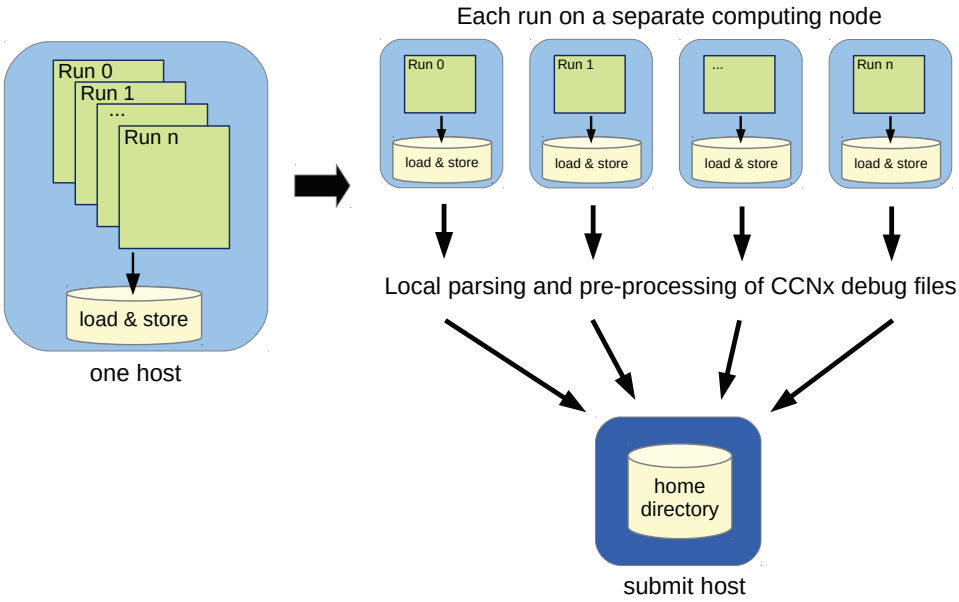


Figure 3.2: Parallelization of NS3-DCE Evaluations.

means that all CCNx instances of the same evaluation run (including all wireless nodes) are always executed on the same computing node such that no communication (or synchronization) between cluster nodes is required. Temporary evaluation files are stored at and loaded from secondary storage of a computing node. For each simulated node of an evaluation run, there is a debug output, which contains all received and transmitted messages, and a status file, which contains the execution time of an application, e.g., the time until a requester has completed the retrieval of a content object. To reduce read and write operations on the cluster's network storage, the output files (debug output and status files) of each evaluation run are parsed and pre-processed locally at every computing node after the run has finished. The pre-processed data (message and time statistics per run) is then compressed and copied to a user's home directory, where it can be further processed by the user. Evaluations with NS3-DCE on Ubelix have been performed in Sections 4.4 and 11.5 as well as Chapters 9 and 10.

3.3 Evaluation Parameters

The implications of ICN are more profound than one might think. Because content can be cached everywhere in the network, performance metrics such as network throughput, bandwidth or latency may not be meaningful in some scenarios because content is not necessarily transmitted between two end points. Since content is automatically cached in ICN, requests from different users are not independent of each other. For example, if a node has requested a specific content object, all of its neighbors do not need to retrieve the same content object from the original

content source anymore but can get it via one-hop communication from the nearest cache. Similarly, if requesters do only request partial content, e.g., only the beginning or the end of a file, other requesters may experience significant RTT variations because some Data messages may be retrieved from caches while others may need to be retrieved from the original content source.

No consensus has been found in related ICN literature how to evaluate information-centric networks, i.e., there is no common evaluation methodology and there are no common evaluation parameters. During the course of this thesis, the evaluation methodology and parameter selection has slightly changed and we use slightly different evaluation parameters in our evaluations (as described in the following subsections). We have obtained statistics for the node roles *requesters*, *listeners* and *content sources* (see Subsection 2.1.1) separately. All results have been obtained via multiple evaluation runs (in most cases: 100 different runs). If evaluations were performed on physical devices, the CCND of all nodes had been restarted between evaluation runs to clear the caches.

3.3.1 Time Parameters

We have used the following parameters to measure the performance of NDN in respect to time. Since only requesters retrieve content, statistics were only collected at requesters. We illustrate parameters with boxplots of minimum values, 25 quantiles, median, 75 quantiles and maximum values (except for cumulative retrieval/decoding times, which show average values).

- **Throughput:** It is calculated based on the size of a content object and the time to retrieve to it. We used this parameter only in our first evaluation works when there was only one requester and one content source, thus, only one ongoing Data transfer that influences the caches.
- **Content Retrieval/Decoding Time:** These parameters denote the time that a requester needs to retrieve a complete content object (or to retrieve a sufficient number of packets to decode a complete content object) after sending the first request. The parameters have been used if there was only one requester.
- **Cumulative Retrieval/Decoding Time:** These parameters denote the average time until a certain number of requesters have retrieved/decoded the complete content object. Hence, these parameters have been used when there were multiple (mobile) requesters in the network, illustrating at what time most of the requesters receive content.
- **Discovery Time:** This parameter defines the time until a node has discovered all content names in a network.

3.3.2 Message Parameters

Since network throughput or content retrieval times do not give a complete picture of the evaluated protocols (e.g., there may be cached content), we evaluated also different message parameters for all node roles. The message parameters have been obtained by, e.g., parsing the CCND debug output, which can be set with the CCND_DEBUG environment variable [163]. For all message parameters, we either show boxplots of minimum values, 25 quantiles, median, 75 quantiles and maximum values, or average values with standard deviations. In general, we evaluate the following message parameters in all scenarios.

- **Transmitted Interest Messages:** The number of Interest messages that are transmitted by a node over the wireless medium (not satisfied by local cache). We evaluate this parameter for requesters and listener nodes. For the latter, these messages are also called "Forwarded Interests" because listeners transmit Interests only during multi-hop communication.
- **Transmitted Data Messages:** The number of Data messages transmitted over the wireless medium by content sources, requesters and listener nodes.
- **Received Data Messages:** The number of Data messages received by requesters and listener nodes.
- **Received Duplicate Data:** The number of duplicate Data messages at requesters or listener nodes, i.e., content that is already available in the local cache.
- **Duplicate Interest messages:** The number of duplicate Interests at listener nodes or content sources. A duplicate Interest is blocked by the PIT because a similar Interest is already pending. We evaluate this parameter only for broadcast multi-hop communication in Chapter 8.
- **Transmitted Notifications:** The number of notifications sent by requester or agent nodes in Chapter 11. For pull-based notifications, the notifications include Interest messages transmitted by requesters and Data messages replied by agent nodes. For push-based notifications, the notifications correspond to Interest messages transmitted by agent nodes.
- **Number of Collisions:** The number of collisions that occur at a requester or content source. We evaluate this MAC layer parameter only in Chapters 6 and 8.
- **Cache Hits:** The number of Interest messages that can be satisfied from a local content store (cache).
- **Message Sizes:** We evaluate the sizes of transmitted Interest and Data messages (transmitted number of bytes) in Chapters 4 and 11.

- **Message Overhead:** For multi-hop communication in Chapters 9 and 11, we evaluate the (Interest and Data) message overhead of requesters, content sources and listener nodes individually as follows.

$$Overhead = \left(\frac{\sum_{i=1}^{N} m_i}{N} \right) \left(\frac{1}{S} \right), \tag{3.1}$$

  where N is the number of nodes in the network, m_i is the number of messages sent by node i and S is the content size (number of segments). The left component in Equation 3.1 denotes the average number of messages transmitted by a node. The average number is normalized by the number of segments (right component) to relate it to the number of required messages (segments). Because requests are not independent of each other, the overhead measures the network traffic, i.e., average values for requesters, content sources and listener nodes, in each run. The boxplots show then the values over 100 different evaluation runs.

3.3.3 Power Measurements

In Chapter 5, we evaluate CCNx on wireless mesh nodes. By this, we can perform energy measurements in idle mode (when nothing is received or transmitted), at requesters, content sources and listener nodes. In particular we evaluate the following parameters.

- **Power Consumption in Watt:** The power consumption at requesters, content sources or listener nodes in comparison to idle mode.
- **Energy Consumption in Joule:** The energy consumption at requesters and content sources during the transmission of different content sizes.

3.3.4 Cache Parameters

In Chapter 12, we evaluate our persistent Caching extension using the following parameters.

- **Hit and Miss Rates:** The number of requests that are satisfied (hit) or not satisfied (miss) by the persistent cache.
- **Number of Deletions:** The number of deletion operations that are required for cache maintenance.
- **Deletion Times:** The average duration of deletion operations, split into sorting, copying and cleanup times.

3.4 Evaluation Overview

Figure 3.3 gives an overview of all applied evaluation methods and specifies the chapters in which they were used. While the OMNeT++ NDN framework and NS3-DCE were used in multiple chapters, all other methods were only used in one chapter. Chapters 4 and 11 contain evaluations with two methods.

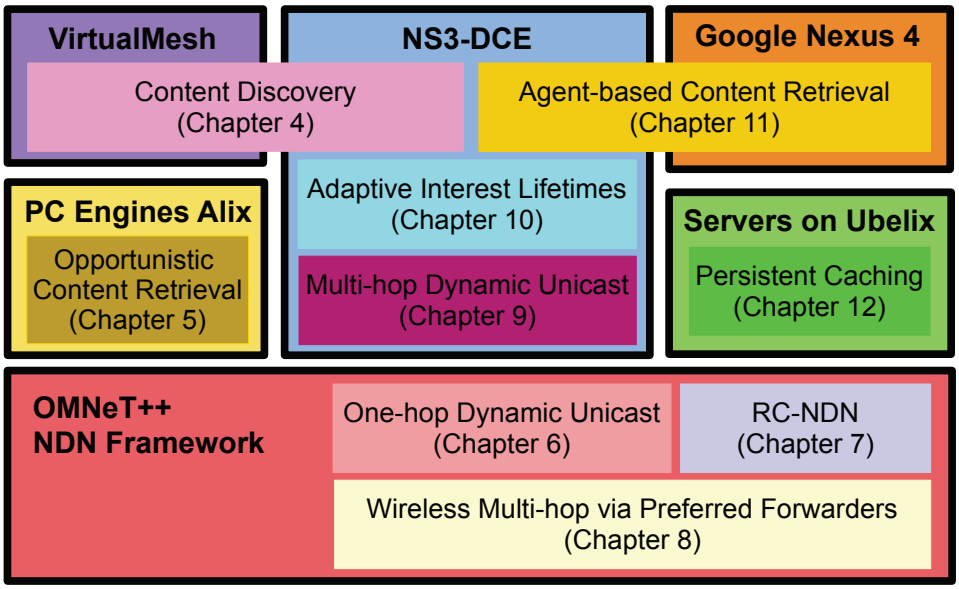


Figure 3.3: Chapter Overview of Evaluation Methods.

Part II

Opportunistic Information-Centric One-hop Communication

In Chapter 4, we describe three opportunistic content discovery algorithms to detect available content names in an opportunistic environment. After content names have been learned, requesters can retrieve desired content. Chapter 5 presents an information-centric resume application to enable opportunistic content retrieval in case of intermittent connectivity to a content source. To increase the efficiency of opportunistic content retrieval from neighbor nodes, we investigate two approaches. First, we present Dynamic Unicast in Chapter 6, which dynamically creates unicast links to neighbors if they hold desired content. Second, we describe RC-NDN in Chapter 7, which encodes Data at content sources to increase diversity in wireless broadcast communication (and reduce duplicate Data transmissions).

Chapter 4

Content Discovery in Opportunistic Information-Centric Networks

4.1 Introduction

Natural disasters, e.g., floodings, earthquakes or wars, can destroy communication infrastructures and, thus, prevent infrastructure-based communication. While most works have focused on redundancy and infrastructure resilience [189], not much work has been performed to investigate approaches for operation during or after disasters, when fixed infrastructures may be destroyed. Information-centric networking seems appealing for opportunistic communication because it enables communication whenever desired content is available independent of neighbor nodes. In fact, recent work [198] has shown that most communication patterns that take place during disasters are of information-centric nature, e.g., retrieval of disaster information or dissemination of warnings. Information-centric networking can work in isolated islands that are disconnected from central infrastructure to provide any kind of information. For example, first responders could carry storage units in their backpacks with world news [198] or cars could provide multi-media content [96] to remote cities that lack the required infrastructure. A node can broadcast a request for a content name to receive content if a content source is available. This is also known as *Implicit Content Discovery*. However, to retrieve content (enable implicit content discovery), knowledge of available content names is required. In a disaster scenario, requesters cannot get naming information from central repositories, and even if they could, it may not be meaningful if most content could not be retrieved due to broken links. In this case, if preferred content is not available because of limited reachability, users may also be satisfied with "second-best" alternatives. For example, if a video with the current weather forecast is not available, users may also be satisfied with a textual weather description.

In this chapter, we investigate algorithms to support opportunistic content discovery based on broadcast requests. We base our work on Named Data Networking (NDN), which is based on a hierarchical naming structure. To ensure globally unique names and support routing, content names may be aggregated by publisher

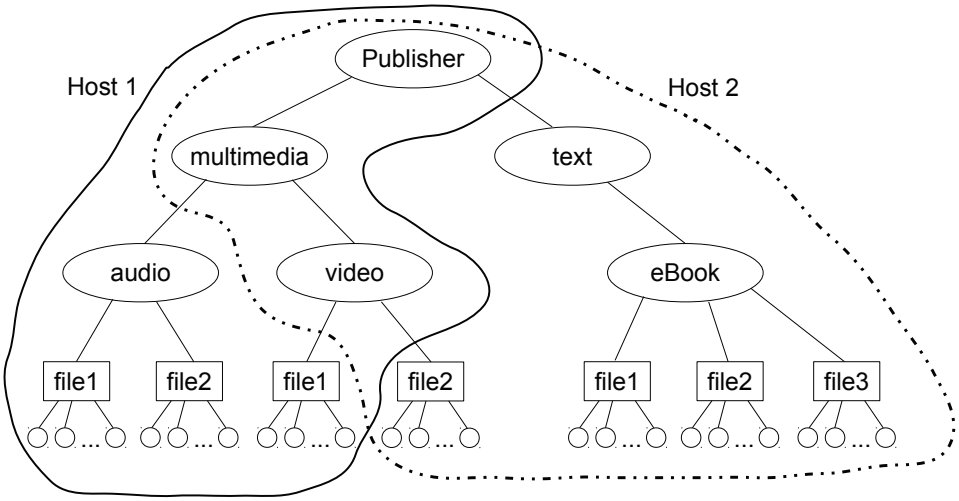


Figure 4.1: Hierarchical Name Structure: Files may be Stored on Different Hosts.

specific prefixes similar to DNS names as illustrated in Figure 4.1. The ellipses correspond to name components and the rectangles to files. Each file consists of one or several segments denoted by small circles. There are no restrictions on content names and they may be selected arbitrarily. Furthermore, hierarchical names may not indicate the location of content objects as Figure 4.1 shows. Content from the same publisher may be downloaded and provided by different mobile hosts. To find content from a specific publisher, an Interest in the general prefix */Publisher* is sufficient due to longest-prefix matching of Interest and Data messages. A node can define the scope of discovered content, e.g., requests with the prefix */Publisher* retrieve all content from a publisher while */Publisher/video* only retrieve videos.

In Section 4.2 we describe two basic and one enhanced content discovery algorithms. The basic algorithms are called Regular Interest Discovery (RID), which is based on implicit content discovery, and Enumeration Request Discovery (ERD), which is similar to DNS-SD [74] in flat namespaces. The enhanced algorithm is called Leaves First Discovery (LFD) and uses elements of both RID and ERD. Our algorithms are designed to find available content names in the absence of central naming repositories. However, once content names or content collections (multiple files with the same prefix) have been identified, implicit content discovery can be used to selectively retrieve desired content or synchronization protocols [9] can be applied to synchronize content collections among repositories. In Section 4.3, we evaluate both basic algorithms (RID and ERD) in a small static network. The conclusions from these evaluations [52, 201] have lead to optimizations of RID and ERD as well as the design of LFD. In Section 4.4, we compare LFD with RID and ERD. These evaluations [50, 182] are performed with mobile nodes in larger networks. Furthermore, all algorithms are analyzed with flat, hierarchical and mixed namespace structures.

4.2 Content Discovery Algorithms

In distributed environments, where connectivity to fixed infrastructures and central repositories may not exist, nodes need to perform content discovery to learn what content names are available. Content discovery algorithms based on Bloom filters [134] or the Sync protocol [9] work only for already known content collections. In addition, the Sync protocol triggers the retrieval of missing content automatically independent of whether users want the content or not. In this section, we describe three algorithms for content discovery in naming structures (name trees) such as depicted in Figure 4.1.

4.2.1 Notation and Parameters

We describe two basic discovery algorithms, namely Regular Interest Discovery (RID) and Enumeration Request Discovery (ERD), as well as an enhanced discovery algorithm called Leaves First Discovery (LFD), which is a combination of both. The common parameters for all algorithms are listed in Table 4.1.

Parameter	**description**
p	request prefix, initially $p = root$
IL	Interest lifetime: IL_{long}, IL_{short}
EF	Exclude filter in an Interest
to	# timeouts, initially $to = 0$ stop if to equals T
NT	name tree with name components
RD	request delay

Table 4.1: Parameter Definitions for Discovery Algorithms.

The request prefix p is initially set to a $root$ prefix, which defines the starting point for discovery, i.e., the root of the name (sub-)tree. Broadcast requests in content prefixes can result in multiple different content replies at the same time. The request delay RD denotes a waiting time between the reception of a Data message and the next Interest transmission; this waiting time has been added to all algorithms based on our evaluations in Section 4.3.3. We use two different values for the Interest lifetime IL, i.e., IL_{short} to retrieve content from the local cache and IL_{long} to get content from neighboring nodes. We do not set the scope in Interest messages with IL_{long}, i.e., unlimited scope, but set it to "only local host" with IL_{short}. Interests do not request specific content but contain a general prefix to discover content published under the prefix. To avoid duplicate transmissions and receptions, Interests have an Exclude filter EF. If an Interest times out, the Interest is retransmitted until the number of timeouts to reaches a certain limit T. All discovered names are stored in a name tree NT.

4.2.2 Basic Algorithm: Regular Interest Discovery (RID)

Algorithm 1 describes Regular Interest Discovery (RID) and we illustrate its operation with the help of the name tree in Figure 4.2. RID is based on recursive expression of Interest messages to browse a name tree via Depth-first traversal. Due to longest prefix matching, a requester can express an Interest in a root prefix $p = /Publisher$ to retrieve content under that prefix (implicit content discovery). To discover the subtree under a root prefix, only the first segment of a content object is requested. After a Data message c has been received (line 10), RID extracts the file name without segment number (line 12), e.g., */Publisher/multimedia/audio/file1* (step 1 in Figure 4.2), and stores it in the name tree NT (line 13). Then, the last component, i.e., *file1*, is removed from the request prefix (line 14) and included in the Exclude filter EF (line 15) to request other files such as file2 (step 2 in Figure 4.2). If only one Interest in the prefix p has been transmitted so far, i.e., *first_request*, and there were no timeouts to (line 16), the algorithm waits for RD before the next Interest transmission to reduce duplicate transmissions (line 17). After waiting for RD, the Interest lifetime is set to IL_{short} to retrieve content from the cache. In case of a timeout with IL_{short}, the Interest is retransmitted with IL_{long} (line 21). In case of a timeout with IL_{long}, the timeout counter to is increased before the Interest is retransmitted (line 23). After T timeouts with IL_{long}, RID assumes that all content has been discovered under that prefix (timeout event) and climbs one level up in the name tree, e.g., to */Publisher/multimedia*, excluding the sub-tree it came from (line 7), i.e., *audio*. RID always requests the first segment of a content object. Thus, it can quickly go down to a leaf of the next sub-tree, e.g., */Publisher/multimedia/video/file1* (step 3 in Figure 4.2). Following this strategy, RID can discover the content names of all 7 content objects in Figure 4.2.

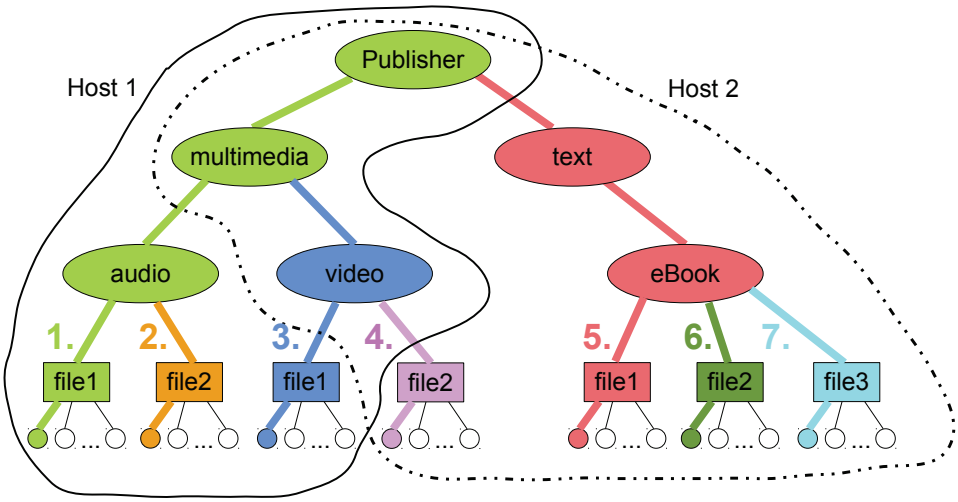


Figure 4.2: RID Request Procedure on Name Structure. The numbers correspond to received Data messages, which contain the first segment of a content object.

Algorithm 1 Regular Interest Discovery (RID)

```
 1: function RID_DISCOVER(p, EF)
 2:     p_new = RID_GET (p, EF, 0)
 3:     if root ∉ p_new then
 4:         SCHEDULE_NEXT_DISCOVERY (time)
 5:     else
 6:         p_next = remove_last_comp(p_new)
 7:         last_comp(p_new) → EF
 8:         RID_DISCOVER(p_next, EF)

 9: function RID_GET(p, EF, to)
10:     c = SEND_INTEREST (p, IL_long, EF)
11:     while (Data c has been received) do
12:         name = getName(c)
13:         name → NT
14:         p = remove_last_comp(name)
15:         last_comp(name) → EF
16:         if first_request and to == 0 then
17:             wait(RD)
18:         c = SEND_INTEREST (p, IL_short, EF)
19:     if to < T then
20:         if timeout with IL_short then
21:             RID_GET (p, EF, to)
22:         else
23:             RID_GET (p, EF, to + 1)
24:     else
25:         return p
```

A full discovery round is completed when RID has climbed up to the root of the name tree and experienced T timeouts, i.e., $root$ is not a prefix of p_{new} anymore (line 3). In the next discovery round (line 4), which starts after $time$, already known names can be excluded.

4.2.3 Basic Algorithm: Enumeration Request Discovery (ERD)

Enumeration Request Discovery (ERD) is based on the consecutive expression of enumeration requests and targets content at repositories. Enumeration requests [6] are Interest messages with a special marker in the name to requests all next level components of a prefix p at a certain repository. The enumeration response, i.e., list of next level components for p, is identified by the ID of the corresponding repository, which is based on its public key. ERD in flat namespaces is similar to DNS-SD [74] and mDNS [73].

Algorithm 2 describes ERD and we illustrate its operation with the help of the name tree in Figure 4.3. ERD starts by expressing an enumeration request in the root prefix, e.g., $p = /Publisher$ (line 13). This may trigger an enumeration response from repository 1 on host 1, i.e., */Publisher/ID1*, containing the next-level component *multimedia* (step 1 in Figure 4.3). The received name components, i.e., here only *multimedia*, are then added to the name tree NT (line 15) and the

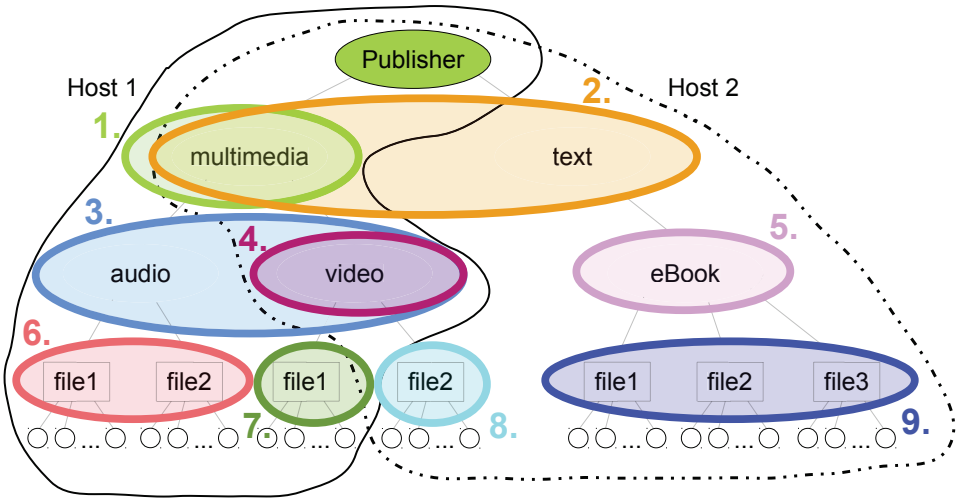


Figure 4.3: ERD Request Procedure on Name Structure. The numbers correspond to received Data messages, which contain the next name components for a prefix at a repository.

Algorithm 2 Enumeration Request Discovery (ERD)

```
1:  function ERD_DISCOVER(p, EF)
2:      ERD_GET (p, EF, 0)
3:      p_next = next_comp_on_lvl(p, NT)
4:      if p_next ≠ {} then
5:          ERD_DISCOVER (p_next, {})
6:      else
7:          p_next = next_lvl_comp(p_new, NT)
8:          if p_next ≠ {} then
9:              ERD_DISCOVER (p_next, {})
10:         else
11:             SCHEDULE_NEXT_DISCOVERY (time)

12: function ERD_GET(p, EF, to)
13:     L = SEND_ENUMERATION (p, IL_long, EF)
14:     while (Enumeration L has been received) do
15:         getNames(L) → NT
16:         getID(L) → EF
17:         if first_request and to == 0 then
18:             wait(RD)
19:         L = SEND_ENUMERATION (p, IL_short, EF)
20:     if to < T then
21:         if timeout with IL_short then
22:             ERD_GET (p, EF, to)
23:         else
24:             ERD_GET (p, EF, to + 1)
```

repository ID *ID1* is added to the Exclude filter (line 16). To reduce duplicates, ERD also waits for RD after the *first_request* (line 18). The next enumeration request uses IL_{short} and excludes *ID1* to retrieve enumeration responses of other

hosts from the cache (line 19). In this example, repository 2 on host 2 has replied with */Publisher/ID2* containing the components *multimedia* and *text* in the payload (step 2 in Figure 4.3). ERD continues with the same prefix (but more excluded IDs) until an enumeration request has timed out T times (timeout event). Then, *next_comp_on_lvl* (line 3) checks the name tree for other name components on the same level. If there are other components, discovery continues on the same level (line 5). If there are no more components, *next_lvl_comp* (line 7) continues with the first component on the next level in the name tree, e.g., */Publisher/multimedia* or */Publisher/text*. Since enumeration responses are identified by repository IDs, redundant information cannot be avoided. If multiple repositories store the same content, their enumeration responses appear to be different (different repository IDs), although they do not provide new information, e.g., information retrieved in step 3 and 4 in Figure 4.3 is partially redundant. Hence, to discover the content names of all 7 content objects in Figure 4.3, 9 enumeration responses are transmitted. Yet, ERD payloads (enumeration responses) contain only content names but no content segments and, thus, have smaller size than RID responses. A discovery run is finished at the leaves of the name tree, i.e., if there are no more name components. Then, the next discovery run starts after $time$ (line 11).

4.2.4 Enhanced Algorithm: Leaves First Discovery (LFD)

Leaves First Discovery (LFD), which uses elements from both RID and ERD, is presented in Algorithm 3 and we illustrate its operation with the name tree in Figure 4.4. Similar to RID, a regular Interest is transmitted first to quickly find content and reach the leaves of the name tree (line 2), e.g., */Publisher/multimedia/audio/file1* (step 1 in Figure 4.4). After the first Data message has been received, LFD has reached a leaf level and can perform ERD discovery to retrieve enumeration responses from all repositories (line 7), e.g., the enumeration response */Publisher/multimedia/audio/ID1* contains the name components *file1* and *file2* (step 2 in Figure 4.4). If all enumerations have been received at a level, i.e., T timeouts have been triggered in ERD_GET (line 7), and it is not the root level (line 8), the algorithm climbs one level up in the name tree by removing the last component (line 11), e.g., from */Publisher/multimedia/audio* to */Publisher/multimedia*, excluding the component from the last sub-tree (line 12), e.g., */audio*, and performs a RID request to quickly reach the leaves of the next sub-tree (line 13), e.g., */Publisher/multimedia/video/file1* (step 3 in Figure 4.4). If no leaves would be found at a level (no other sub-tree), RID requests would be retransmitted up to T times (line 15) before climbing one level up (lines 17-19). Following this strategy, LFD can discover all 7 content objects in Figure 4.4. Thus, LFD includes advantages of RID to quickly reach content leaves but does not require as much data overhead as RID because mostly content names are requested and fewer content segments. Discovery stops with T timeouts at the root of the tree and the next discovery starts after $time$ (line 9 and 21).

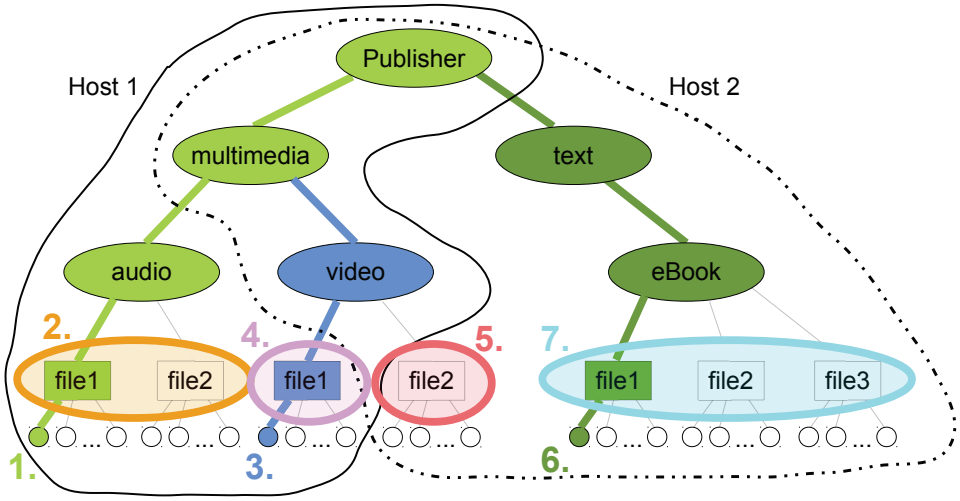


Figure 4.4: LFD Request Procedure on Name Structure. The numbers correspond to received Data messages, which contain either a content segment (to reach the leaves level) or the next name components for a prefix at a repository.

Algorithm 3 Leaves First Discovery (LFD)

```
1:  function LFD_DISCOVER(p, EF, to)
2:      c = SEND_INTEREST (p, IL_long, EF)
3:      if (Data c has been received) then
4:          name = getName(c)
5:          name → NT
6:          p_new = remove_last_comp(name)
7:          ERD_GET (p_new, {}, 0)
8:          if p_new == root then
9:              SCHEDULE_NEXT_DISCOVERY (time)
10:         else
11:             p_next = remove_last_comp(p_new)
12:             last_comp(p_new) → EF
13:             LFD_DISCOVER (p_next, EF, 0)
14:     else if to < T then
15:         LFD_DISCOVER (p, EF, to + 1)
16:     else if p ≠ root then
17:         p_next = remove_last_comp(p)
18:         last_comp(p) → EF
19:         LFD_DISCOVER (p_next, EF, 0)
20:     else
21:         SCHEDULE_NEXT_DISCOVERY (time)
```

4.2.5 Overview of Discovery Messages

In Table 4.2 we list the required discovery messages for each algorithm.

Algorithm	Messages
RID	Interest message to retrieve Data message with first segment of a content object. → Depth First Search (DFS) on name tree.
ERD	Enumeration request (Interest message) to retrieve enumeration response (Data message) with next-level name components from each repository. → Breadth First Search (BFS) on name tree.
LFD	Interest message to retrieve Data message with first segment of a content object to quickly reach leaf level. On leaf level: enumeration request to retrieve enumeration response with file names from each repository. → First DFS, then BFS.

Table 4.2: Interest and Data Messages transmitted for each Content Discovery Algorithm.

4.3 Evaluation of Basic Algorithms

4.3.1 Evaluation Tools

In this section, we evaluate implementations of Enumeration Request Discovery (ERD) and Regular Interest Discovery (RID) algorithms, which have been implemented in CCNx 0.4.2. The implementations are evaluated by emulations with VirtualMesh [187], which combines the real network stack (by running CCNx on virtualized hosts) with simulations of the wireless communication. The wireless communication is simulated by the OMNeT++ [205] network simulator using the INET framework [11] with the default IEEE 802.11b MAC layer implementation.

4.3.2 Evaluation Parameters and Scenarios

Figure 4.5 shows the evaluation topology and Table 4.3 lists the evaluation parameters. Four repository nodes, i.e., hosts running repositories containing different content objects, are placed in a grid with 50m distance to each other. A *discovering node* (black node in Figure 4.5) is placed in the middle at a distance of 35m to each repository node. Due to the short distance, all five nodes can directly communicate and overhear broadcast messages from each other. We only consider transmission errors due to collisions but no additional bit error models.

The *discovering node* performs a discovery operation on the four repository nodes. We differentiate between two basic content distribution scenarios in our evaluations:

1. **Common Scenario**: all repositories store exactly the same content objects.

2. **Distinct Scenario**: every content object is uniquely stored at only one of the repositories.

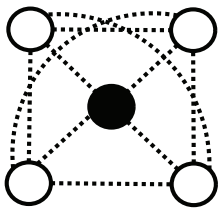


Figure 4.5: Network Topology with 1 Discovering Node (middle) and 4 Repository Nodes with Content Objects.

Parameter	**Value**
interface	1 × IEEE 802.11b, data rate: 11 Mbps
propagation loss model	free space path loss path loss: $\alpha = 2$
radio sensitivity	-85 mW
TX power	2.0 mW
Mobility	static
playground size	100m x 100m
repository nodes	4 nodes grid with 50m distance
Interest lifetime	0.5s retransmission after 0.6s
retransmission limit T	2

Table 4.3: Simulation Parameters for Content Discovery Evaluation with VirtualMesh.

All content objects *content_#* are published under the same name hierarchy */prefix/content_#* and the segment size is set to 4096 bytes. The discovery algorithms are implemented as applications forwarding Interests via the local face to the CCND. If the content is in the content store, it will be returned immediately without forwarding the Interest to the wireless medium, otherwise the Interest is forwarded to other nodes and temporarily included in the PIT. Both discovery algorithms (ERD and RID) express Interests in the general prefix '/prefix/' to discover the available content objects at all repositories. Based on the reception of a discovery response (Data message with first segment or enumeration response with name components), the algorithms express the next Interest excluding already received information.

In this section, we evaluate early RID and ERD implementations, which differ slightly from the descriptions in Section 4.2. First, we do not differentiate between between long and short Interest lifetimes IL_{long} and IL_{short}. The Interest lifetime is set to 0.5 seconds and we perform a retransmission of the same Interest after a retransmission delay of 0.6 seconds if no response has been received. The retransmission delay is slightly larger than the Interest lifetime to ensure that the existing PIT entry has expired on the discovering node such that a retransmitted Interest can be forwarded by the local CCND. Second, we do not use a request delay, i.e.,

$RD = 0$. In Subsection 4.3.3, we evaluate the impact of RD on duplicate transmissions. For this, the RD delay is applied after each Interest transmission and not only after the first request (different than described in Section 4.2).

If not stated otherwise, we use a retransmission limit of two retransmissions before a timeout is assumed (three transmissions in total). All cached Data messages remain valid for the entire duration of the discovery. Every configuration is evaluated in 20 different runs and before each run starts, all CCND caches are cleared.

4.3.3 Discovery Delays

Broadcast requests may trigger potentially many responders. In CCNx, broadcast content transmissions are scheduled randomly by a *broadcast delay* within the interval $[DP, 3DP]$, where DP denotes the *data pause*. Once scheduled, the content object stays in the sender's send queue until the broadcast delay is due; then it is forwarded to lower layers for transmission. A long DP may increase the discovery time but enables other hosts to detect identical responses. In this subsection, we evaluate different DP values and their impact on the discovery time as well as transmitted Interest and received duplicate Data messages.

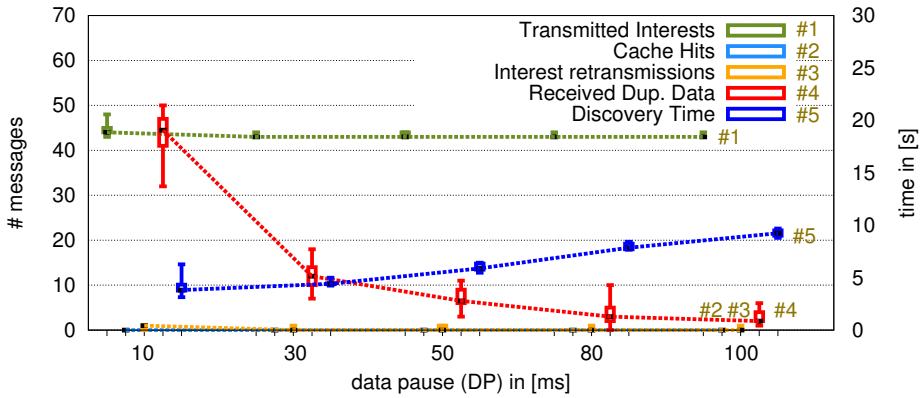


Figure 4.6: Discovery of 40 Content Objects with RID in the Common Scenario.

Figure 4.6 shows the performance of RID discovery if the network comprises 40 different content objects, which are all stored on all hosts, i.e. the Common Scenario. The x-axis denotes different DP values in milliseconds. The figure shows the transmitted Interest and received duplicate Data messages at the discovering node as well as the time to discover all content objects, i.e. the discovery time. As expected, the number of received duplicate Data messages is higher with short DP values and decreases significantly with higher values. For a DP of 10ms, the number of received duplicate Data messages is even higher than the number of transmitted Interests. Because of the small DP value, the repository nodes schedule their Data transmissions almost at the same time not leaving enough time to

detect and suppress duplicate Data transmissions. As soon as the Data messages have been forwarded from the send queue to the lower layers, no cancellation is possible anymore. The number of required Interest retransmissions is surprisingly low: for a DP of 10ms, every Interest is retransmitted at most once. For the discovery of 40 content objects, such retransmissions occurred at most five times when using a DP value of 10ms and at most once when using a DP value of 30ms or higher. Since all content objects are stored on all hosts (Common Scenario), every Interest triggers the transmission of the same Data message from all repository nodes. Therefore, to discover 40 content objects, the discovering node transmits at least 43 Interests: 40 Interests to discover the content objects and 3 additional Interests to detect a timeout using the retransmission limit of 2.

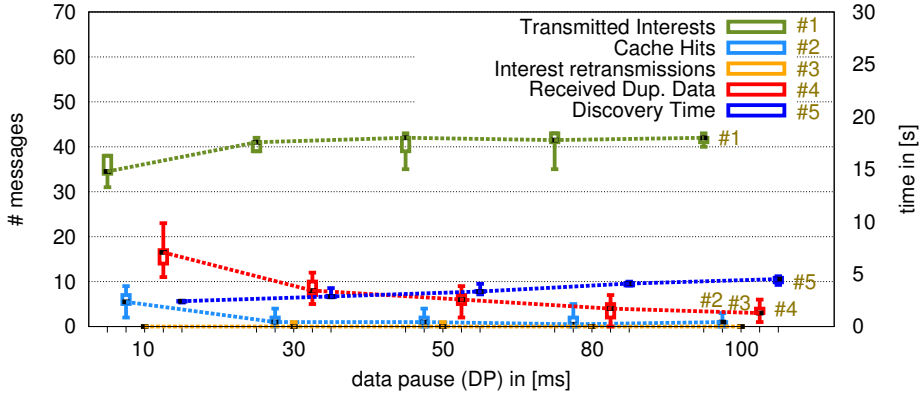


Figure 4.7: Discovery of 40 Content Objects with RID in the Distinct Scenario.

Figure 4.7 shows the results for the discovering node when using RID discovery for 40 content objects stored uniquely at different nodes (Distinct Scenario). Since every content object is only stored at one node, every Interest in the general */prefix/* will trigger different Data replies from the repositories (longest-prefix matching of Interest and available Data messages at each repository). Since transmitted Data messages are not the same, hosts do not cancel their scheduled Data transmissions if they overhear another Data transmission, although they reply Data to the same Interest. In fact, if Interests are expressed in content prefixes but not exact content names, a repository node has no means in determining whether an Interest has already been answered by another node or not (unless both nodes would answer with the same Data message, which could be identified as duplicate Data transmission). Therefore, the content store of a discovering node may receive multiple Data messages per Interest, i.e., *parallel Data transmissions*, but only one Data message per Interest is forwarded to the discovering application. Subsequent Interests may then be satisfied from the content store (cache) and may not be transmitted over the wireless medium anymore. The discovery time for RID in the Distinct Scenario is approximately halved compared to the Common Scenario since different repository nodes reply to the same Interests with different Data messages

resulting in a faster discovery. However, the number of cache hits is quite low as Figure 4.7 shows. Surprisingly, Figure 4.7 shows that although all content objects are uniquely stored at only one host, the discovering node receives duplicate Data messages for all DP values. This means that a repository node sends the same Data message multiple times. The reason for this is the fact that subsequent Interests are expressed immediately after the reception of a Data message resulting in duplicate Data transmissions in case of unsynchronized repositories (Distinct Scenario). We

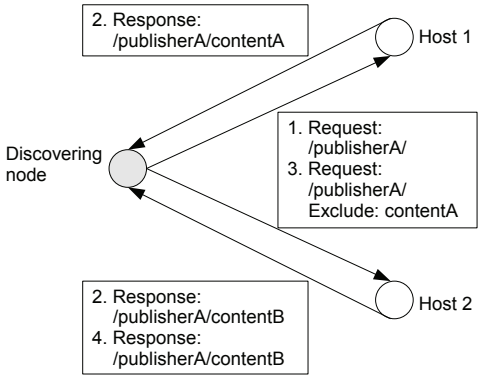
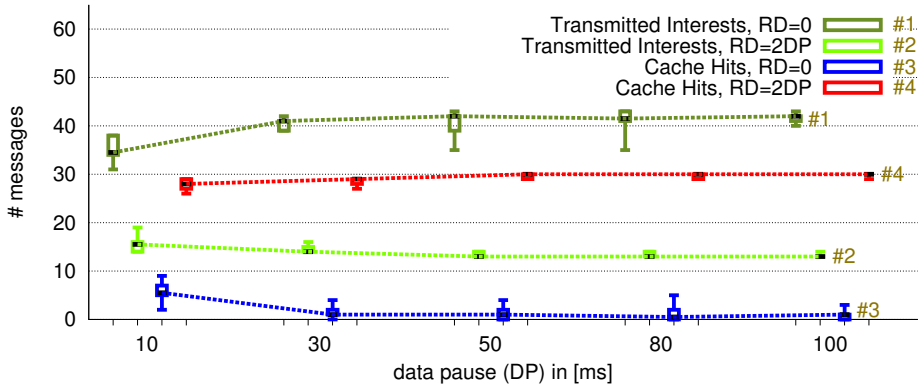


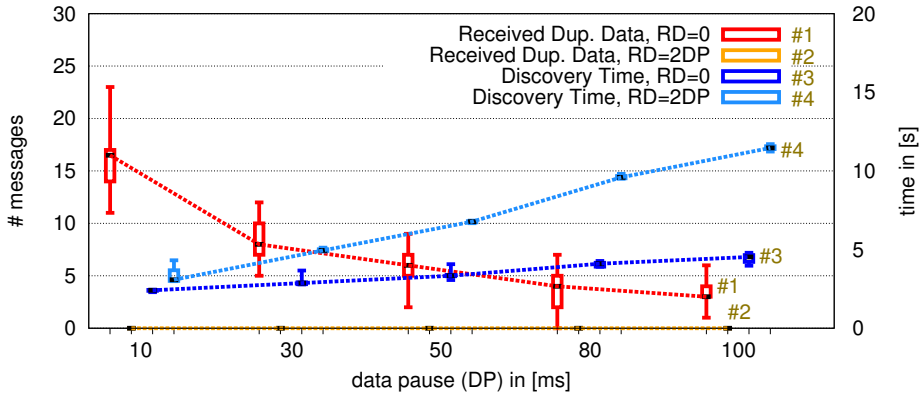
Figure 4.8: Duplicate Data Transmissions in case of Unsynchronized Repositories.

illustrate the problem with the help of Figure 4.8 where two hosts store different content objects. An Interest in the general prefix '/publisherA/' triggers different Data responses from both hosts. While host 1 answers with 'contentA', host 2 may schedule the transmission of 'contentB'. If the discovery node would transmit the next Interest immediately after receiving 'contentA' from host 1 but before receiving 'contentB' from host 2, the Interest would only exclude 'contentA'. If host 2 receives the new Interest but has already scheduled the transmission of 'contentB', i.e., removed it from the send queue and forwarded it to lower layers, it does not remember the previous transmission of 'contentB'. Consequently, since the Interest only excludes 'contentA', host 2 would transmit 'contentB' again (duplicate Data transmission). Therefore, whenever a discovering node receives a Data reply, we wait an additional Request Delay (RD) before the next Interest is transmitted. We set $RD = 3DP - 1DP = 2 \times DP$, i.e. the difference between maximum and minimum broadcast delay. This enables the reception of Data replies from other repository nodes before the next Interest is transmitted. If different Data messages have been received, the next Interest may be satisfied from the local cache. Otherwise, the Interest may be forwarded to the wireless medium.

Figure 4.9a shows transmitted Interest messages and cache hits when applying a Request Delay RD. For $RD = 2DP$, three times more Interest messages can be satisfied from the cache and, therefore, fewer Interests need to be transmitted over the wireless medium. Figure 4.9b compares received duplicate Data messages and discovery times in the same scenario. For $RD = 2DP$ we can avoid the reception of any duplicate Data relieving the wireless medium from unnecessary transmis-



(a) Transmitted Interests and Cache Hits.



(b) Received Duplicate Data and Discovery Time.

Figure 4.9: Discovery with RID for RD=2DP and RD=0 in the Distinct Scenario.

sions. However, the discovery time increases compared to $RD = 0$ because RD is applied after every Data reception, i.e., before each new Interest transmission.

In the following evaluations of this section, we set $DP = 50ms$ and $RD = 2DP$. This prevents duplicate Data transmissions in the Distinct Scenario and results in a low number of duplicate Data transmissions in the Common Scenario. It is not possible to avoid duplicate Data transmissions completely in the Common Scenario because two senders may always select the same broadcast delay with a certain probability depending on the DP value. We observed in our evaluations that DP values above 50ms do not significantly reduce the number of received duplicate Data messages but result in a much larger discovery time. Since in all our evaluations, retransmissions occurred very infrequently and at most once per Interest, we set the retransmission limit to 1 (two transmissions in total). Although higher network congestion levels might require higher retransmission limits to discover content, it would also result in more Interest retransmissions, which may increase

congestion even more. Thus, in highly congested networks, it might be better to consider the corresponding content objects as (temporarily) unavailable.

4.3.4 Enumeration Request Discovery vs. Regular Interest Discovery

In this section, we compare Enumeration Request Discovery (ERD) and Regular Interest Discovery (RID) with respect to discovery time and number of transmitted Interest messages. We use a data pause of $DP = 50ms$ and set $RD = 2DP$. The retransmission limit is set to 1 resulting in 2 unresponded Interest transmissions before a timeout is detected. We evaluate the efficiency of ERD and RID in the same 5-nodes scenario as illustrated in Figure 4.5 and consider different numbers of content objects and content distributions, i.e., Common and Distinct Scenario. All hosts either comprise 1, 4, 12, 20, or 40 content objects.

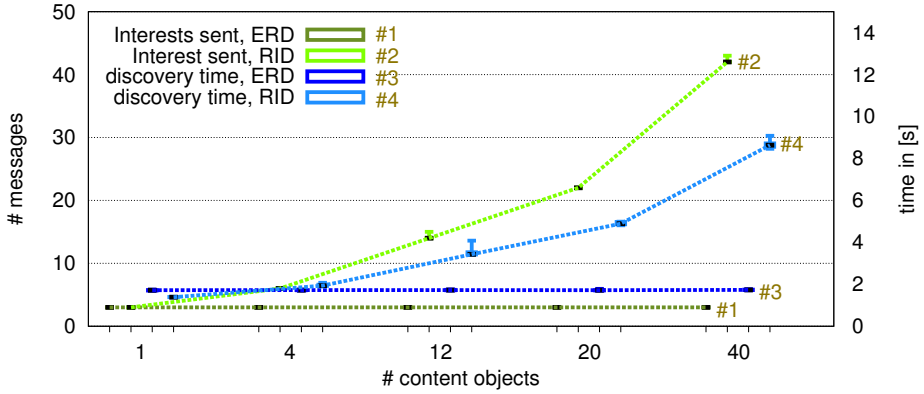


Figure 4.10: Comparison between ERD and RID in the Common Scenario.

Figure 4.10 illustrates the number of Interest transmissions and the discovery time for ERD and RID in the Common Scenario. The x-axis denotes the number of content objects to be discovered. If only one content object is available, RID is more efficient, since it cannot learn anything new after the first request and the discovery stops quickly. On the contrary, ERD requests an enumeration response (list of name components) from all hosts. Hence, only after checking the name components from all Enumeration Responses, the discovering node can be certain to have received everything. Although this would require multiple Enumeration requests, subsequent Enumeration responses may be retrieved from the cache. The number of Enumeration requests does not depend on the number of content objects on hosts but on the number of hosts in the vicinity. Therefore, the number of Enumeration requests and the discovery time is constant in our setting for all content configurations. On the contrary, the number of transmitted Interest messages and Data replies increase significantly for RID with increasing number of content objects. If all hosts store the same content objects, the requester has to express an

Interest for every single content object. Since RID requests ask for the first segment of a content object, the transmissions of the corresponding Data messages require considerably more time.

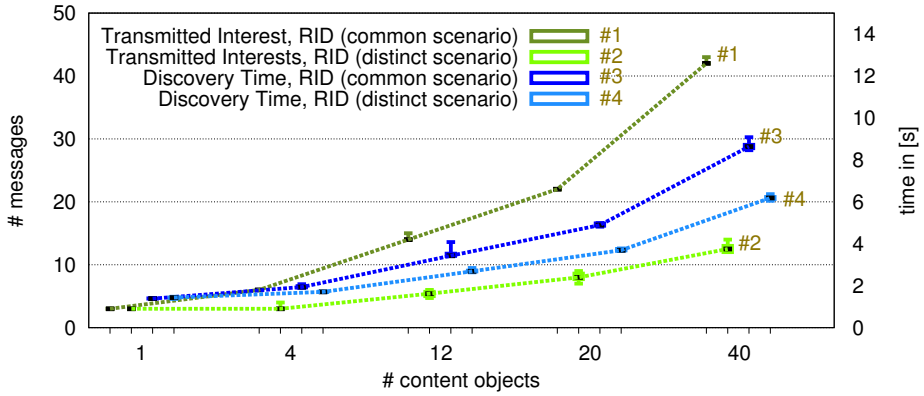


Figure 4.11: RID Performance in the Common and Distinct Scenario.

The comparison between ERD and RID looks similar for the Common and Distinct Scenario. If only one content object is stored at one node, only this node will respond to requests in the Distinct Scenario. In this case, ERD performs similar to RID: only three discovery requests are transmitted (one Interest to retrieve the content and two unresponded Interest transmissions to detect a timeout). However, RID performance degrades with increasing number of discovered content objects due to the increasing number of transmitted Data messages similar to the Common Scenario. While ERD performs nearly identical in the Distinct and Common Scenario, RID performs better in the Distinct Scenario. Figure 4.11 illustrates the number of Interest transmissions and the discovery time for RID in the Common and Distinct Scenario. Compared to the Common Scenario, the number of Interest transmissions can be reduced by up to 70% (for 40 content objects) in the Distinct Scenario. This reduction is due to parallel Data transmissions, i.e., one Interest may trigger Data replies from different hosts, as described in Subsection 4.3.3. Yet, despite significantly fewer Interest transmissions in the Distinct Scenario, the discovery time is reduced only by up to 30%. This is because we evaluate an early implementation of ERD and RID in this section, which applies RD after each Interest transmission. However, this may result in unnecessary waiting times, e.g., it may not be required to apply RD if Interests can be satisfied from the local cache and are not transmitted to the wireless medium. Due to the evaluations in this section, we have optimized our algorithms as described in Subsection 4.2. In particular, RD is only applied after the *first_request* (which uses a long Interest lifetime IL_{long} to retrieve content from other repositories) but RD is not applied when targeting Data from the cache (with shorter Interest lifetime IL_{short}).

4.3.5 Conclusions

Discovery of available content names is very important in mobile NDN to learn what content is available. Users require this information to retrieve content in subsequent content retrievals. We evaluated two methods for content discovery: ERD is based on name enumeration requests and RID is based on regular Interests. The discovery algorithms target the wireless broadcast environment. Since wireless broadcast communication is unreliable and no MAC layer acknowledgments are available, discovery mechanisms have to account for occasional collisions. Therefore, we included a retransmission counter that initiates a retransmission if no information is received within a timeout period. Evaluations have shown that a retransmission limit of 1 (in total 2 unresponded Interest transmissions) is enough to detect timeouts, i.e., unavailability of content objects, in our scenarios. Furthermore, evaluations have shown that delaying the transmission of content objects, i.e., increasing the data pause, helps reducing collisions and duplicate Data transmissions but this is not enough. In case of unsynchronized repositories with different content objects, a Request Delay is required between Data reception and next Interest transmission to avoid duplicate Data transmissions and reduce the number of transmitted Interests, i.e., benefit from parallel Data transmissions. However, to avoid unnecessary waiting periods when retrieving Data from the cache, the Request Delay should only be applied for the first request in a general prefix.

Enumeration Request Discovery (ERD) has shown good performance in our evaluations, because it is independent of the number of content objects. However, the approach depends on the number of repository nodes that store the requested content. Therefore, the approach may be inefficient in mobile scenarios with many repositories. Compared to RID, ERD responses need to be processed and accumulated from all repositories to know which content names are available. Therefore, if all hosts store the same content objects, ERD needs to request and process enumeration responses from all nodes without learning something new. RID is more efficient to detect small differences in collections, because it can ask specifically for new content. Redundant information can be included in Exclude filters of Interest headers to avoid duplicate Data transmissions. RID may also be faster in finding content objects in highly structured name spaces with many name components. In our evaluations, we considered a flat name space where ERD can perform well. In a more structured (hierarchical) name space ERD would require subsequent traversing through all name components until reaching the leaves of a name tree. Therefore, the combination of both approaches, i.e., Leaves First Discovery (LFD), may be promising. An initial RID request may quickly find the complete content name at the leaf level. Then, by expressing enumeration requests with the prefix of the received content object, content names may be retrieved from all repositories.

4.4 Evaluation of Leaves First Discovery

4.4.1 Evaluation Tools

In this section, we evaluate the basic algorithms RID and ERD and compare it to the enhanced LFD algorithm. We have implemented RID, ERD and LFD in CCNx 0.8.2 [27]. The evaluations have been performed in mobile scenarios with NS3-DCE [16]. To perform extensive evaluations with many nodes and multiple parameters in parallel, we have run NS3-DCE on Ubelix [200], the Linux cluster of the University of Bern.

4.4.2 Evaluation Parameters

The evaluation parameters are listed in Table 4.4. We use IEEE 802.11g wireless interfaces and a Log-Distance propagation loss model. The transmission power is set to 16 dBm and the energy detection threshold is set to -76 dBm, which corresponds to transmission ranges of up to 60m. The playground size is set to 150m $\times$ 150m indicating a place of interest where users meet, e.g., convention center, stadium or marketplace. We explore different namespace structures for content discovery (see Subsection 4.4.3). To better investigate the impact of content density and neighbors, content objects are provided by repositories in a static grid of 25 or 100 nodes, i.e., in a grid with 25m or 12.5m distance between nodes. A

Parameter	**Value**
interface	1 $\times$ IEEE 802.11g
propagation loss model	Log Distance
energy detection threshold	-76 dBm
TX power	16 dBm
mobility	Random Waypoint, speed: 1.2 - 1.4m/s pause time: 1s, skip time: 3600s
playground size	150m x 150m
repository nodes	25 nodes grid with 25m distance 100 nodes grid with 12.5m distance
data pause DP	270ms
Interest lifetime	long: 875ms, short: 125ms
retransmission limit T	1
resume interval	2s

Table 4.4: Simulation Parameters for Content Discovery Evaluation with NS3-DCE.

mobile requester performs opportunistic one-hop content discovery, while moving according to the Random Waypoint Model with a pedestrian speed between 1.2 - 1.4m/s. We select Random Waypoint mobility to better evaluate the differences of the described algorithms with dynamic connectivity. More realistic mobility models such as SLAW [132] or SMOOTH [152] create node clusters such that some

nodes may never meet each other and efficient multi-hop communication would be required, which is out of the scope of this work (see Part III for more information on multi-hop communication). The broadcast delay uses a data pause DP of 270ms. The DP value is larger than in the Section 4.3 due to significantly more repository nodes. Furthermore, we set the request delay to $RD = 2DP$ to reduce the number of duplicate transmissions as shown in Section 4.3. However, in contrast to previous evaluations, we only apply the request delay after the first request as described in Section 4.2. The Interest lifetimes are set to $IL_{long} = 875ms$ (long Interests) and $IL_{short} = 125ms$ (short Interests). If a short Interest times out, a long Interest is transmitted. After $T = 1$ timeouts of long Interests, a timeout event is triggered, i.e., the algorithms proceed with the next prefix. If not all content objects could be discovered within one discovery round, discovery is resumed after 2s. Every configuration is evaluated in 100 different runs.

4.4.3 Evaluation Scenarios

The performance of content discovery depends on the structure of the namespace. Therefore, we perform evaluations in diverse namespaces. In each scenario, we set the number of distinct content objects to 100. Please note that the goal of discovery is not to learn all content names in the Internet but only what is locally available, e.g., from a certain provider such as BBC or YouTube. Thus, we evaluate the performance of broadcast requests in a distributed environment where different nodes may store different content objects.

1. **flat namespace:** all 100 content objects are published under the same prefix, i.e., there is no naming structure. Every content has one distinct name component.

2. **hierarchical namespace:** all 100 content objects are published under a hierarchical prefix with 10 name components. We use 4 different hierarchical prefixes with 10 components for the 100 content objects.

3. **mixed namespace:** it is composed of hierarchical and flat names. In total, every prefix has randomly between 1 and 10 components. Every node in the name tree has between 1 and 5 different children. At the leaves, the number of content objects is randomly set between 1 and 20.

The namespaces are designed such that every name component is 12 bytes long. In every scenario, each node contains 4 different content objects. In the 25 nodes network, every content object is stored uniquely at one node. In the 100 nodes network, content is distributed randomly among all nodes, i.e., there are redundancies. We evaluate RID, ERD and LFD in terms of discovery time, transmitted Interest and received Data messages as well as received duplicate Data messages.

4.4.4 Modifications to CCNx 0.8.2

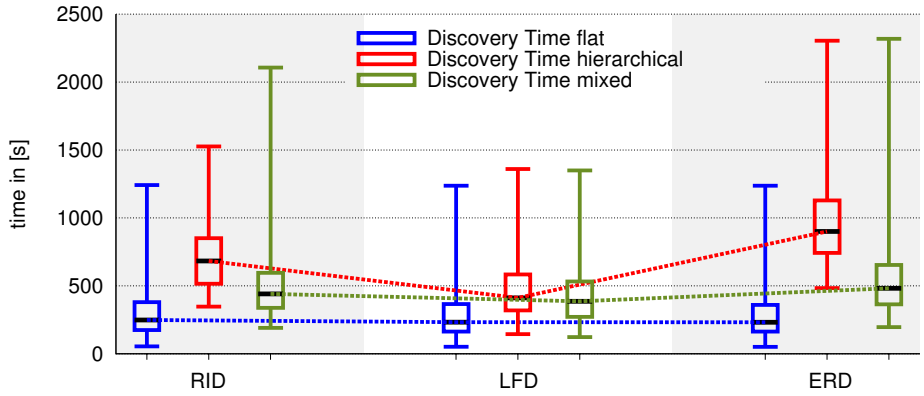
All discovery algorithms transmit Interest messages (with content prefixes) via broadcast to all repositories in the vicinity. This means that repository nodes that contain different content objects may reply to the same Interest with different Data messages (parallel Data transmissions, see Section 4.3). However, only one Data message is forwarded to the Discovery application (the PIT entry is satisfied afterwards), and other received Data messages are considered as *unsolicited content*. In CCNx 0.8.2 (in contrast to CCNx 0.7.2 or older versions), flow balance between Interest and Data messages is enforced by marking unsolicited content as stale immediately after reception, i.e., as if it was expired content. Stale content in the cache is deleted and not returned to Interest messages, even if it would satisfy them. This is a very inefficient strategy because the content has already been received and needs to be retransmitted by content sources. Previous work has shown that it is beneficial to keep unsolicited content in wireless information-centric communication [52, 213, 217, 38] because requesters can retrieve content without knowing exact names or resources, address multiple content sources at the same time and benefit from parallel Data transmissions by exploiting the cache. Therefore, to improve efficiency for wireless communication, CCNx 0.8.2 needs to be slightly modified such that unsolicited content does not become stale immediately.

4.4.5 Discovery Time

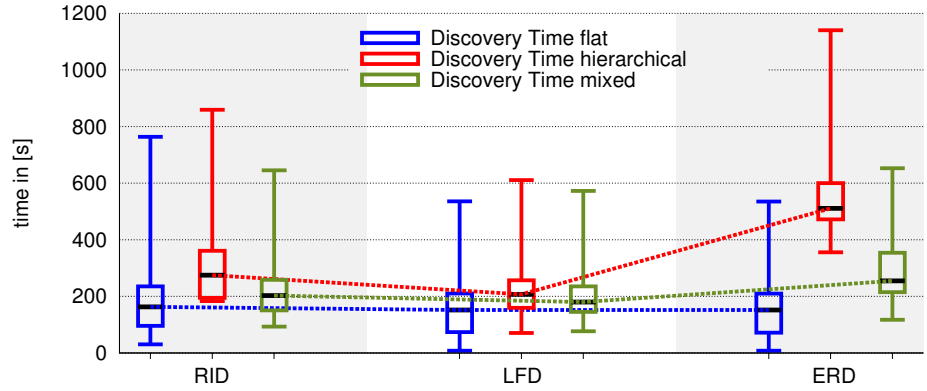
Figure 4.12a illustrates discovery times for 100 content objects in the 25 nodes network. The x-axis shows the results for RID, LFD and ERD and the y-axis shows the discovery time in seconds. Figure 4.12a indicates that flat namespaces result in shortest discovery times. The difference in discovery time between RID, ERD and LFD is insignificant. LFD requires 0.8% more time than ERD and RID requires 5% more time than LFD. Although RID requests each content object separately, the discovery time is only slightly longer than for LFD and ERD. This is due to broadcast requests that can trigger multiple distinct content transmissions, i.e., parallel transmissions, from different content sources (see Section 4.3). If unsolicited content is not discarded, it can be collected quickly by subsequent Interests from the cache.

Discovering content in a hierarchical namespace requires more time because algorithms need to climb up and down the name tree, which includes more waiting times due to timeouts. For example, LFD requires 77% more time in a hierarchical namespace compared to a flat namespace. Figure 4.12a shows that LFD handles naming hierarchy better than RID and ERD: it results in 40% shorter discovery times than RID and even 54% shorter discovery times than ERD.

It may seem surprising that LFD performs even better than RID in the hierarchical scenario, because each node contains 4 content objects with different hierarchical prefixes. This means that every enumeration response with LFD contains only 1 content name. However, LFD can quickly reach the leaf level and then re-



(a) 25 Nodes



(b) 100 Nodes

Figure 4.12: Discovery Time for RID, ERD and LFD in Flat, Hierarchical and Mixed Namespaces.

trieve content names faster than RID because Data messages at the leaf level are smaller, i.e., they contain only names. While RID could benefit largely from parallel transmissions in flat namespaces, fewer parallel transmissions are possible in hierarchical namespaces (only for content with the same prefix).

In mixed namespaces, LFD performs on average 18% faster than RID and 29% faster then ERD. The maximum discovery times of RID and ERD are even 56% and 72% longer than for LFD since more time is required to browse the name tree, which is disadvantageous in case of mobility.

Figure 4.12b shows discovery times for RID, LFD and ERD in a network with 100 nodes. Due to higher content density, discovery times have significantly decreased compared to the 25 nodes scenario in Figure 4.12a. In hierarchical namespaces, discovery times have decreased more than in flat namespaces due to fewer climbing operations, i.e., content can be found quicker. RID benefits the most from a higher content density due to more parallel transmissions, while LFD and ERD benefit less, because the number of nodes increases as well, which means

that more enumerations have to be requested.

Furthermore, the relative differences between flat and mixed namespaces have decreased in the 100 nodes scenario. For example in the 25 nodes network, RID discovery in mixed namespaces results in 69% longer discovery times than in flat namespaces, while in the 100 nodes scenario the difference between mixed and flat namespaces is only 24%. If content density is high, discovery in flat namespaces results in more collisions than in mixed namespaces because more content is requested at the same time (broadcast requests).

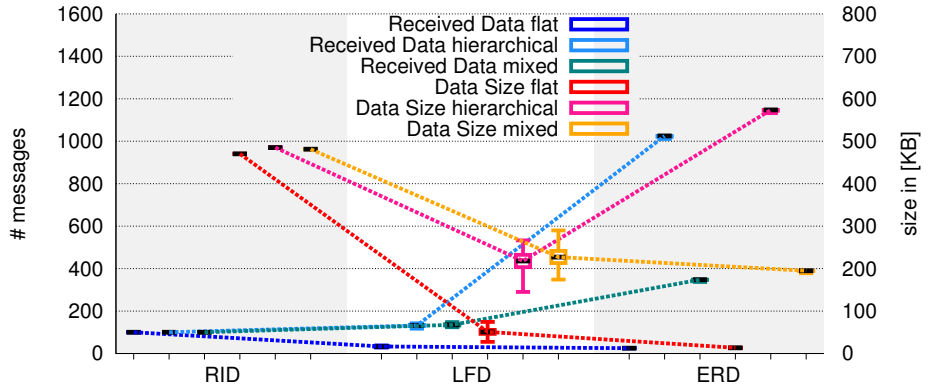
4.4.6 Data Messages

Data messages with RID always contain a payload of 4096 bytes, i.e., first data segment, while Data messages with ERD are smaller because they contain only lists with name components. Figure 4.13a shows, therefore, not only the number of received Data messages at the requester (left y-axis) but also their size in bytes (right y-axis) for the 25 nodes network.

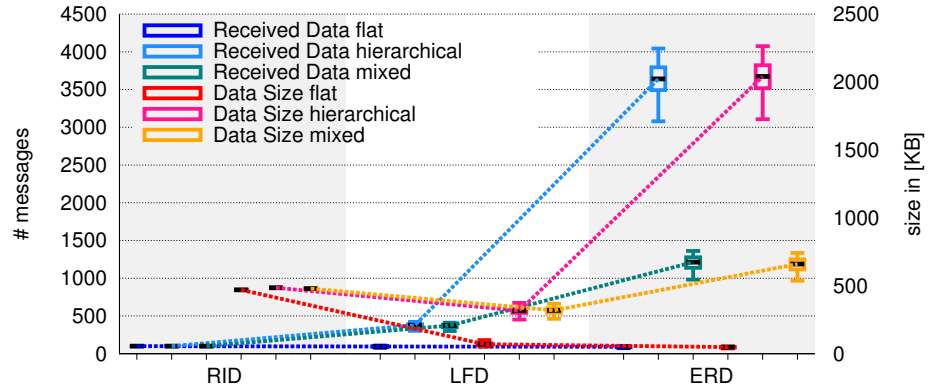
In flat namespaces, ERD results in the fewest transmitted Data messages (and fewest transmitted bytes) because only one list needs to be transmitted per node, while RID needs to transmit one Data message per content object. On average, LFD results in 2.8 times more data bytes and RID even in 34.3 times more data bytes than ERD. The data overhead between LFD and ERD is larger than expected, because LFD needs to transmit an Interest to reach the leaf level. Due to multiple nodes in wireless transmission range, an LFD requester may retrieve 7 - 8 segments in response to this Interest (due to parallel transmissions), although only one Data message would be enough to reach the leaf level. However, ERD does not result in the fewest transmitted bytes in all namespaces. In hierarchical namespaces, ERD results in 2.6 times more transmitted data bytes than LFD and even 18% more bytes than RID. This is because ERD transmits 7.8 times more Data messages than LFD while climbing down the name tree and even 10 times more Data messages than RID. Although individual packets are smaller for ERD than for RID, the sum of transmitted bytes (packet headers and payloads) of all Data messages is larger.

In the mixed namespace, LFD transmits on average 53% fewer bytes than RID but 14.5% more bytes than ERD. LFD transmits slightly more bytes than ERD because it retrieves a Data message for every name sub-tree to reach the leaf level. However, the situation changes for an increased node density. Figure 4.13b shows the number of Data messages in the 100 nodes network. In the mixed namespace, ERD results in the most transmitted Data bytes, i.e., 37% more than RID and even 107% more than LFD.

In hierarchical namespaces, LFD results in 36% fewer bytes than RID and even in 85% fewer bytes than ERD. Only in flat namespaces, ERD performs slightly better than LFD, i.e. 32% fewer bytes, but the difference is negligible (a few KBs) compared to the large overhead in hierarchical and mixed namespaces (several hundreds of KBs), where ERD requires even more Data bytes than RID. In all scenarios, LFD results in significantly fewer Data bytes than RID.



(a) 25 Nodes



(b) 100 Nodes

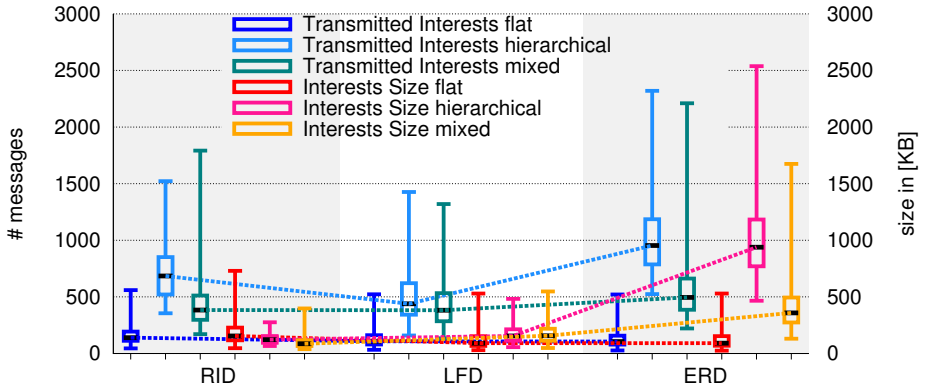
Figure 4.13: Transmitted Data for RID, ERD and LFD in Flat, Hierarchical and Mixed Namespaces.

4.4.7 Interest Messages

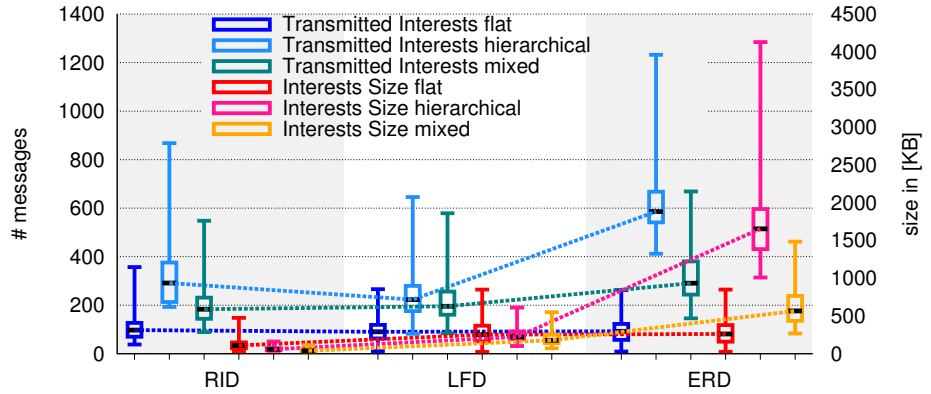
Figure 4.14a illustrates transmitted Interest messages (left y-axis) and their sizes in bytes (right y-axis) in the 25 nodes network. Because Interests only retrieve content, their size may often be neglected. However, Exclude filters may grow with the number of discovered content objects (or the number of repositories) resulting in large Interest messages. We only evaluate Interests transmitted over the wireless medium, i.e., Interests with the lifetime IL_{long}.

The number of transmitted Interests in the flat namespace is similar, LFD sends 21% fewer Interests than RID and 1.5% more Interests than ERD. RID Interests are slightly larger due to larger Exclude filters (more excluded components). Thus, RID sends 36% more bytes in Interests than LFD and ERD.

In the hierarchical namespace, LFD results in 28% fewer Interests than RID and even in 43% fewer Interests than ERD. However, the size of transmitted Interests with RID is 26% smaller than for LFD and even 87% smaller than for ERD.



(a) 25 Nodes



(b) 100 Nodes

Figure 4.14: Transmitted Interests for RID, ERD and LFD in Flat, Hierarchical and Mixed Namespaces.

This is due to two reasons. First, there are four different hierarchical prefixes, thus, fewer components need to be excluded with RID compared to the flat namespace (namespace partitioning). Second, Exclude filters with LFD and ERD are larger because they include repository IDs and not name components. While repository IDs have a static length of 38 bytes, name components have only a length of 12 bytes in our scenarios.

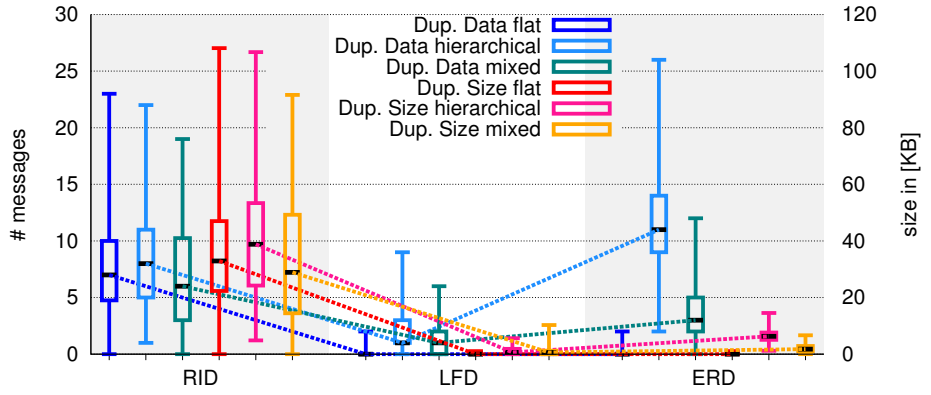
Similar observations can be made in mixed namespaces. However, compared to the hierarchical namespace, LFD and ERD require slightly fewer Interest bytes because more names can be included in the same enumeration responses, i.e., fewer Interests are required.

Figure 4.14b shows the number of Interest messages in the 100 nodes network. Surprisingly, the number of transmitted Interests can be reduced compared to the 25 nodes scenario. For example in the flat namespace, the number of Interests can be reduced by 34% (RID), 27% (LFD and ERD) compared to the 25 nodes

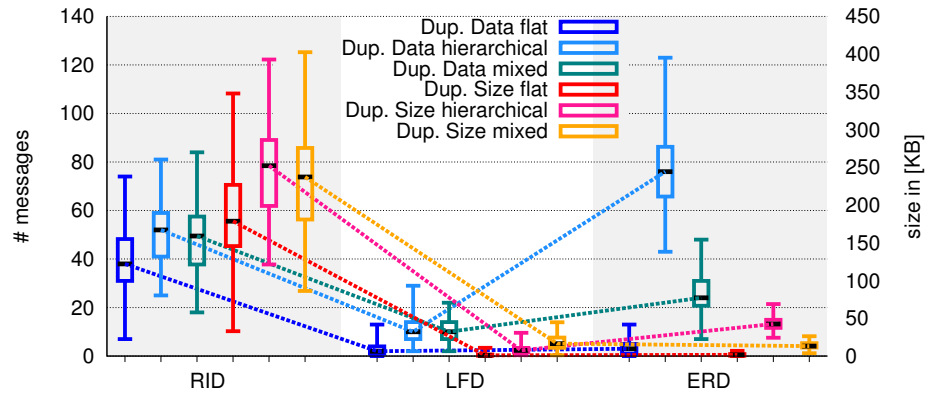
scenario. Due to higher content density more content can be retrieved via parallel transmissions. Consequently, the Interest bytes for RID decrease by 48%. For LFD and ERD, however, transmitted bytes increase despite fewer Interests by 144% (LFD) and 147% (ERD). Due to higher node density, transmitted Interests that are not satisfied from local cache have longer Exclude filters because more repository IDs need to be excluded. Similar observations can be made for the hierarchical and mixed namespace. In the worst case (maximum values), transmitted Interest bytes with ERD can become even larger than transmitted Data messages in bytes (see last subsection).

4.4.8 Duplicate Data

Figure 4.15a shows received duplicate Data messages (left y-axis) and their sizes (right y-axis) at the requester in the 25 nodes network. It may be unexpected to have duplicates in the 25 nodes network where every content is uniquely stored at only one node. However, a node may request content from a repository and then



(a) 25 Nodes



(b) 100 Nodes

Figure 4.15: Duplicate Data for RID, ERD and LFD in Flat, Hierarchical and Mixed Namespaces.

move out of range for reception. Other nodes may receive the content and keep it in the cache. If the requester moves into communication range again, multiple nodes may reply with the same content resulting in duplicate Data transmissions. Although duplicate suppression can reduce the number of duplicates, it cannot completely prevent them (see Section 4.3).

Figure 4.15a shows that most duplicates (except for the hierarchical namespace) are received with RID and the fewest duplicates are received with LFD. Since duplicates with ERD and LFD are rather small, RID always results in the most duplicate Data bytes transmitted. However, compared to the size of transmitted Interest and Data messages, the overhead for duplicate Data transmissions is insignificant in the 25 nodes scenario.

Figure 4.15b shows the number of duplicate Data messages and their sizes in the 100 nodes network. Due to higher node and content density, the number of duplicate Data messages has increased by a factor of 5 or more compared to the 25 nodes network. Figure 4.15b shows that duplicate Data bytes are still negligible for LFD and ERD. Although ERD results in 5 times more duplicate Data bytes than LFD in hierarchical namespaces, which may seem high, it corresponds only to 2.5% of all transmitted Interest bytes with ERD. For RID, however, duplicate Data becomes a significant fraction of network traffic as more duplicate Data bytes are transmitted than Interest bytes.

4.4.9 Discussion

Namespace Design for Disaster Scenarios

Discovery in flat namespaces is considerably faster compared to hierarchical namespaces because no browsing is required. However, if node and content density is high, flat namespaces can result in many duplicate Data transmissions and collisions. In addition, the size of Interest messages increases since more components need to be excluded. Content providers can, therefore, use hierarchical names (depending on the number of content objects they provide) to partition the namespace and limit the number of Data replies.

If time matters, e.g., in emergency scenarios, flat namespaces or only a few hierarchy levels should be preferred. To facilitate content discovery in such scenarios, authorities may define artificial flat namespaces, e.g., */emergency*, and link them to ”real”, i.e., existing and potentially hierarchical, content. The linking can be done by *alias mappings*, which we define as Data messages with artificial alias names that include a list of real content names in the payload. Requesters can then learn content names by inspecting the payload of received alias mappings.

Table 4.5 shows examples of alias mappings that could be defined in an emergency scenario. Alias mappings enable authorities to promote already widely distributed content, e.g., the location map of a building, and mark it as important without re-publishing and, thus, re-signing the content. Hence, existing information such as a location map can be complemented with new instructions tailored

Alias Name	Real Content Names
/emergency	/police/instructions, /building3A/locationMap, /medical-service/first-aid/video
/notfall	/polizei/anweisungen, /building3A/locationMap, /medical-service/first-aid/video
/emergenza	/polizia/istruzione, /building3A/locationMap, /medical-service/first-aid/video

Table 4.5: Examples of Alias Mappings.

to the current situation. Even content from different providers, e.g., police and medical services, can be linked together with the same artificial alias name, e.g., */emergency*, to consolidate important information. In addition, multiple alias mappings can be defined for the same content. For example, as shown in Table 4.5, alias names in different languages */emergency*, */notfall*, or */emergenza* may link to the same content names */building3A/locationMap* and */medical-service/first-aid/video*. Hence, since content retrieval is performed via the real content name learned from the payload of an alias mapping, requesters can still identify identical content in caches despite different alias names. Like all NDN Data messages, alias mappings are signed such that users can lay their trust in alias mappings based on the authority that created it.

Privacy Considerations

Ubiquitous caching in NDN raises several privacy concerns [128, 129, 29, 71] because attackers can monitor cached objects to infer what certain users have requested in the past. In fact, privacy threats are inherent to NDN, i.e., there is a tradeoff between performance and privacy [128, 129, 29].

Since longest-prefix matching of Interest and Data messages (in combination with Exclude filters) enables attackers to identify content in caches without knowing exact names, longest-prefix matching is no longer supported in CCNx 1.0 [150]. Instead, content or service discovery is performed via dedicated Service Discovery Brokers [178], which listen for Interests, e.g., *parc/Services*, to reply with available content or services. Hence, content or service providers can explicitly specify which content they want to be discovered and preserve privacy for all other content. However, disabling longest-prefix matching may still not prevent leakage of private information through periodic cache probing [128, 71].

Most content becomes privacy-sensitive if it can be linked to an individual user [129]. In mobile wireless networks, connectivity to neighbor nodes may change dynamically (in contrast to static networks). Thus, since NDN messages do not comprise endpoint identifiers, cached broadcast traffic may not be mapped to a unique user (except the content publisher). Furthermore, longest-prefix matching with Exclude filters has significant benefits for wireless communication because it enables distributed resource discovery [217, 45] and significant bandwidth savings

[52, 38]. For this reason, longest-prefix matching is still supported in the NDNx framework [14].

To avoid leakage of private information, users may configure access policies to their repositories and distinguish between private and public content [29, 128] preventing private content to be discovered and served from caches. Yet, public content, e.g., area maps, weather forecasts or news, can still be retrieved from repositories and caches in an opportunistic way [160] without device discovery.

4.5 Conclusions

If connectivity to a fixed communication infrastructure is broken, e.g., during or after a disaster, users may not be able to retrieve information from central repositories in the Internet. To provide limited communication, users need to learn names of available content objects in a distributed way before they can (potentially) retrieve them. In this chapter, we have described three algorithms for distributed content discovery, namely RID, ERD and LFD. While RID is based on implicit content discovery and ERD is similar to DNS-SD in flat namespaces, LFD combines elements from both. All algorithms have been evaluated by emulations in mobile scenarios using flat, hierarchical and mixed namespaces.

Evaluations have shown that algorithms designed for flat namespaces do not perform well in hierarchical namespaces. If the namespace structure is unknown, LFD should be preferred because it performs better than ERD and RID in most scenarios. In flat namespaces, which are optimal for ERD, LFD results in only slightly more Data traffic than ERD (same discovery time) but in significantly lower Data overhead than RID. In hierarchical and mixed namespaces, LFD performs significantly better than ERD because fewer Interest and Data messages are required (lower message overhead). Although LFD sends slightly more bytes via Interest messages compared to RID, the overall traffic including Data and duplicate messages is still higher with RID. In addition, LFD results in shorter discovery times than both ERD and RID.

For higher node densities, LFD and ERD can send fewer Interest messages than in sparse densities but the Interest messages become larger due to longer Exclude filters. In hierarchical and mixed namespaces, ERD performs the worst because it may transmit even more bytes via Interest than Data messages. However, even in high node density networks, LFD still results in lower traffic overhead and shorter discovery times than RID.

As future work, an adaptive request strategy for LFD on the leaf level, i.e., send RID or ERD requests depending on whether enumeration responses contain the same information or not, may further improve message efficiency. To support this, it would be beneficial to use hashes from payloads as enumeration identifiers instead of repository IDs. Furthermore, Interests may be extended by a *discovery flag* to avoid the retrieval of content segments but only retrieve meta information, e.g., only the content name.

After content discovery has been performed (as described in this chapter), discovering nodes know what content names are available. Then, they can start content retrieval of specific content objects. In Chapter 5, we investigate opportunistic content retrieval in case of disrupted connectivity to content sources.

Chapter 5

Opportunistic Content Retrieval with Resume Operations

5.1 Introduction

The main goal of opportunistic communication [155] is to exploit contact opportunities between users to support best-effort content and service interactions where fixed network infrastructure may not be available. Broadcast requests enable implicit content discovery (see Chapter 4) to quickly find a suitable content source.

However, during short opportunistic contacts to content sources, content retrieval may not be completed at once. Although received content is cached locally, persistence is not guaranteed for a long time since caches are limited in size and can be overwritten by other content depending on cache replacement strategies. If the disruption between contacts (downloading opportunities) is too long, retrieved partial information may be removed from the cache. If contact times to content sources are always too short to retrieve content at once, content retrieval is never successful. NDN can support the resumption of disrupted content retrievals since content is organized in segments and communication is pull-based, i.e., only missing segments need to be requested. However, to enable requesters to resume content retrievals after long disruptions, e.g., several hours or days, partially received Data needs to be stored on and loaded from persistent storage.

In this chapter, we describe a requester application to support content retrieval for opportunistic networking during short network contacts, where only parts of the content can be exchanged. In Section 5.2 we describe why long-lived NDN messages are not beneficial to support delay-tolerant networking. Then, we describe our design for an opportunistic content-centric retrieval application in Section 5.3. Evaluation results are shown in Section 5.4. We perform all evaluations on PCEngines Alix 3D2 [20] wireless mesh nodes. This allows us also to measure power consumption of CCNx at requesters and content sources during wireless (unicast and broadcast) communication.

5.2 Long-lived NDN messages

In NDN, Interest and Data messages are only valid for a limited time, i.e., *Interest Lifetimes* determine the maximum lifetime of PIT entries and *freshnessSeconds* specify the maximum lifetime of Data messages in the content store. PIT entries maintain soft states (similar to registrations) such that Data messages can flow back to requesters. Hence, increasing Interest lifetimes to obtain long-lived Interests [213, 234, 38] may seem a good idea to enable delay-tolerant networking in the presence of long disruptions but there are drawbacks:

1. Multiple Interests are required to obtain all file segments. Since a requester does not know the length of the requested file until receiving the final segment, proactive transmission of multiple Interests would be required. If all entries are valid for a long time, the PIT size would increase drastically degrading lookup performance.

2. Long-lived Interests stay in the PIT and prevent forwarding of similar Interests because requests are already pending. Forwarding and retransmission is blocked for the duration of the entire Interest lifetime even if the environment has changed due to mobility and the content would be available.

Instead of increasing Interest lifetimes, they could be limited to rather short values but Interests can be re-expressed periodically to account for changes in availability. As we will see later in Section 5.4, short Interest lifetime values are particularly advantageous during broadcast communication due to faster retransmissions in case of collisions.

Similarly, freshnessSeconds could be increased to keep Data messages for a longer time in the cache. However, since caches have limited sizes, Data messages may still be replaced before they expire. To ensure Data availability for a longer time, content needs to be stored persistently.

5.3 Storage Persistence

Every content segment is named individually using a segment number. A NDN requester can request the first segment followed by $n - 1$ subsequent segments depending on the pipeline size n, i.e., the maximum number of segments that can be requested concurrently. In case of disruptions, content retrievals are aborted and can be restarted again at a later time. If disruptions are short, the downloaded segments may still be available in the local cache of a requester and no redundant Interest or Data messages need to be transmitted. However, if disruptions are long, received Data messages may be removed from the cache. To obtain storage persistence, a requester needs to store the partial file and Meta Data on a secondary storage.

In the following two subsections we explain what *Meta Data* is required and describe mechanisms to resume disrupted content retrievals by a sample *Download Sequence*.

5.3.1 Meta Data

For every incomplete and aborted content retrieval, all received partial data for *"name"* is stored in a file *name.part* and Meta Data is stored in a file *name.meta*. The required Meta Data to perform a resume operation is listed in Table 5.1.

1. Name of Content Object
2. Version of Content Object
3. Next Segment
4. File Position
5. Publisher's Public Key Digest
6. Expiration Time

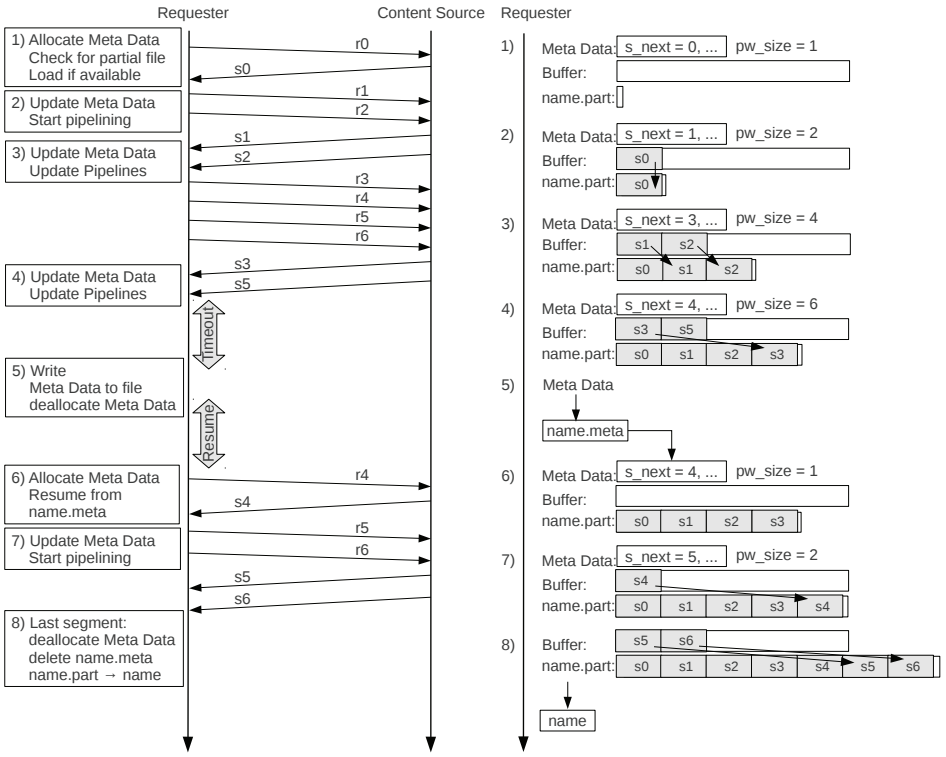
Table 5.1: Meta Data for Persistent Storage after Incomplete Content Retrievals.

Name and version of the content object can be stored together in a string. The name is used to relate the file *name.meta* to the corresponding partial file *name.part*. The third field defines the segment number that needs to be received next (where resumption should be performed). The file position specifies the file offset in *name.part*, i.e., where new segments need to be appended. It depends on the number and sizes of received segments. The publisher's public key digest is used to check that the resumed content retrieval is requesting content from the same publisher. To avoid incomplete files that never get completed or storing Meta Data of real-time traffic, the expiration time indicates a timeout value after which *name.meta* and *name.part* can be deleted. The expiration time can be based on the *reception time* and *freshnessSeconds* of the first received segment. For example, in case of real-time traffic, i.e., if the content is only valid for a few seconds, persistent storage is not required.

5.3.2 Download Sequence

We illustrate content retrieval with an example. Figure 5.1a shows a sample resume sequence and Figure 5.1b depicts the corresponding storage management at the requester.

At the beginning of a content retrieval in step 1), the application checks for available *name.meta* files. If *name.meta* is available, it is loaded, otherwise, content retrieval starts from the beginning by transmitting a request (Interest message) *r0* in segment *s0*. If a content object is available and the segment has been received, the requester can start expressing multiple requests at the same time in step 2). Similar to TCP slow start, the number of concurrently transmitted Interests is increasing exponentially by doubling a pipeline window size pw_{size} for correctly received segments up to the maximum value p_{max}, i.e., the pipeline size. Since



(a) Resume Sequence **(b)** Resume Storage

Figure 5.1: Download Sequence for Content Retrieval with Resume Capability.

requesters do not know the size of a requested content object until receiving the final segment, all Data segments are requested sequentially. The content retrieval application uses a buffer of size p_{max} to temporarily store received segments until they can sequentially be written to *name.part*. In steps 3) and 4) more segments are received and pw_{size} is adapted accordingly.

In case of an Interest timeout, i.e., not receiving the segment before the Interest expires, pw_{size} is reduced to 1 and the corresponding segment is requested again. This strategy targets particularly situations where requesters are disconnected from content sources such that every Interest retransmission would result in a timeout. In case of timeouts due to collisions of some segments, other segments may still be received correctly without transmitting new Interests. If the number of unsuccessful Interest retransmissions exceeds a threshold t_{thresh}, a timeout event, i.e., disconnection from the content source is assumed. In this case as shown in step 5), the Meta Data is stored in *name.meta* and all buffers are released. Therefore, segments that have already been received but not yet written to *name.part* such as segment *s5* are discarded because segment *s4* has not been received. The reasoning behind discarding is the following. First, segment sizes may vary, and thus, the length of a potential placeholder in *name.part* is not known. Second, pipelining works efficiently if it is only increased from a certain value. In case of placeholders

and holes in *name.part*, more state information (received and missing segments) is required. Because the file size is not known, no fixed bit fields indicating received and missing segments (similar to BitTorrent [82]) can be used. However, the number of segments that are not stored due to disruptions is limited by the pipeline size p_{max}. For a pipeline size of 16 and a segment size of 4096 bytes, this corresponds to less than 70KB (worst case) redundant Data transmissions.

Detecting the availability of a content object can be performed via Interest probing. In the simplest way, Interest messages can be transmitted periodically to probe for content, but also more sophisticated mechanisms are possible, e.g., dynamic probing based on previous content retrievals or based on the current location. If a resume operation is performed (step 6), Meta Data is loaded from *name.meta* and the download is resumed from the last missing segment. Then, pw_{size} starts again at 1 and is increased exponentially for correctly received segments (step 7). If the final segment has been received (step 8), which is indicated by a flag, the content retrieval is finished. Then, all Meta Data and buffers are released, *name.meta* is deleted, and *name.part* is renamed to *name*.

5.4 Evaluation

We have implemented content retrieval with resume capability as NDN application in CCNx 0.6.0. The implementation has been tested on PCEngines ALIX 3D2 [20] wireless mesh nodes running on ADAM [188] (see Subsection 3.2.2). All nodes use IEEE 802.11a wireless interfaces configured in ad hoc mode.

5.4.1 Evaluation Scenarios

The evaluation is performed in a static setting of two nodes: one content source shares content via a repository and one requester transmits Interests for the content. In our scenarios, we assume short contacts between requester and content source so that content retrieval can not be completed at once. We implement this by defining *disruption points*, i.e., specific numbers of segments, after which the requester will stop requesting segments emulating a disruption timeout.

In our evaluations, there is always exactly one disruption per measurement and we measure the *effective content retrieval time* based on two content retrievals: First, the content retrieval time until a disruption point is reached and then, the content retrieval time of a second content retrieval that is always successful. After long disruptions, content from the first incomplete content retrieval is not available anymore in the cache. This is enforced by restarting the CCND daemon after a disruption to clear the cache. If resume operations are enabled, the application loads the stored Meta Data before starting the second content retrieval. The content retrieval time in opportunistic networks would also depend on the time a connection is disrupted, i.e. the disruption time. However, the disruption time is an additive constant that could be added to the measured effective content retrieval time.

In dynamic environments where neighbors change frequently, no static unicast FIB entries can be configured. Since NDN messages do not include a destination node address, they can be efficiently transmitted on wireless broadcast media using broadcast MAC frames. However, since MAC acknowledgments are only transmitted in case of unicast communication, automatic retransmissions can not be performed during broadcast communication. Thus, in Subsection 5.4.2, we explore mechanism to increase information-centric broadcast throughput via Interest lifetimes. Then, in Subsections 5.4.3 and 5.4.4, we compare broadcast with unicast communication (for reference purposes) by configuring the FIB accordingly. The two-node scenario indicates the baseline performance of broadcast communication since there is no benefit compared to unicast. However, broadcast performance may improve compared to unicast if more requesters are available at the same time. In our evaluations, the main differences between unicast and broadcast are the mechanisms on the MAC layer. The contention window, which controls the delay until a packet is transmitted on the MAC layer, can only be adapted during unicast. During broadcast, the contention window is by default larger resulting in lower data rates. In addition, broadcast NDN communication is delayed to enable duplicate suppression (see Subsection 2.1.1). In our evaluations, we use the default data pause DP of 10ms.

5.4.2 Interest Lifetime for Broadcast Content Transmission

During broadcast communication, collisions can only be detected by unanswered Interest requests. The Interest lifetime has direct impact on Interest retransmissions and, thus, on throughput, because Interests can not be forwarded in case of existing PIT entries. In this subsection, we evaluate different Interest lifetime values for broadcast communication.

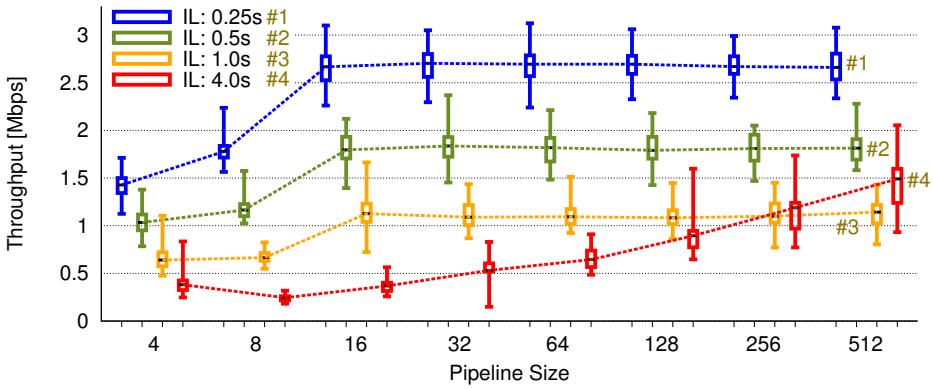


Figure 5.2: Broadcast Throughput for Different Interest Lifetimes (IL).

In Figure 5.2, we evaluate the throughput of a 2 MB file with a segment size of 4096 bytes using Interest lifetimes of 4.0, 1.0, 0.5 and 0.25 seconds. The x-axis

shows different pipeline sizes and the y-axis the achieved throughput. An Interest lifetime of 4 seconds is the default value in CCNx. Transmissions with large Interest lifetimes result in low data rates for small pipeline sizes because of large retransmission delays in case of packet collisions. For an Interest lifetime of 4s, throughput increases with larger pipeline sizes from 16 up to 512 by a factor of 4, i.e., from 0.38 Mbps to 1.50 Mbps. The reason for this increase is the larger number of Interests that are transmitted concurrently until a timeout has been detected. These Interests may retrieve and pre-fetch Data messages that are stored in the content store of a requester (since not all Data transmissions result in a collision). After a timeout has been detected, only Interests for collided Data messages need to be retransmitted and subsequent Interests can be served from the pre-fetched Data messages in the cache. However, this strategy will cause 75% more Interest transmissions compared to a pipeline size of 16 due to more collisions.

While Interest lifetimes of 4 seconds may be reasonable in multi-hop networks, lower values can be used during opportunistic one-hop communication. By decreasing the Interest lifetime from 4s to 0.25s, the throughput increases drastically by a factor of 7.2 for a pipeline size of 16, i.e., from 0.37 Mbps to 2.67 Mbps. Pipeline sizes above 16 and Interest lifetimes shorter than 0.25s do not result in any performance gain.

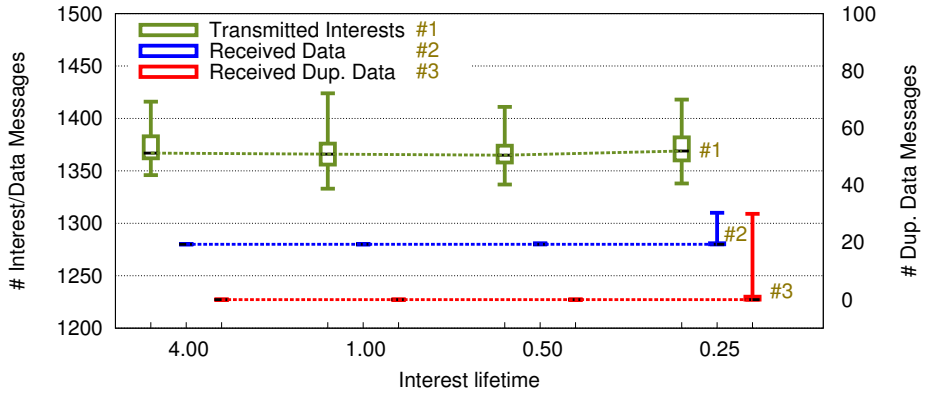


Figure 5.3: Interests, Data and Duplicate Data Messages for Different Interest Lifetimes.

In Figure 5.3 we investigate the transmitted messages during the content retrieval of a 5 MB file using a pipeline size of 16. The x-axis denotes different Interest lifetime values. The left y-axis shows the number of transmitted Interest or received Data messages. The right y-axis shows the number of received duplicate Data messages. The number of transmitted Interests depends on the number of collisions and is the same for all evaluated Interest lifetimes. An Interest lifetime of 0.25s results in a few occasional duplicate Data messages: the median value is 0, the 75-quartile is 1 and the maximum value, which occurred only once in 100 measurements, is 30. The reason for the duplicates are Interests that are retransmitted shortly before the corresponding Data message has been received. Since

content sources do not memorize recently transmitted Data messages, they will respond to retransmitted Interests again as we have already observed in Subsection 4.3.3. In the worst case, an Interest lifetime of 0.25s results in an overhead of around 2% Data messages (duplicates), which corresponds to less than 140 KB. However, the median throughput increases for an Interest lifetime of 0.5s to 0.25s by 48% from 1.79 Mbps to 2.67 Mbps.

5.4.3 Effect of Resume Capability

In this subsection, we investigate content retrieval with resume operations as described in Section 5.3.2 and compare it to *ccncat* [162], a regular CCNx content retrieval applications without resume capability. Figure 5.4 shows effective content retrieval times of a 5 MB file via broadcast communication for different segment sizes and disruption points. The colors represent different segment sizes, i.e. the payload in Data messages without NDN headers containing names, signatures etc. The x-axis denotes different disruption points represented by the received KBs before a disruption occurs and the y-axis shows the content retrieval times in seconds. Content retrievals with resume (continuous lines) and without resume (dotted lines) operation use a stop-and-wait strategy, i.e., a pipeline size of 1, to transmit Interests.

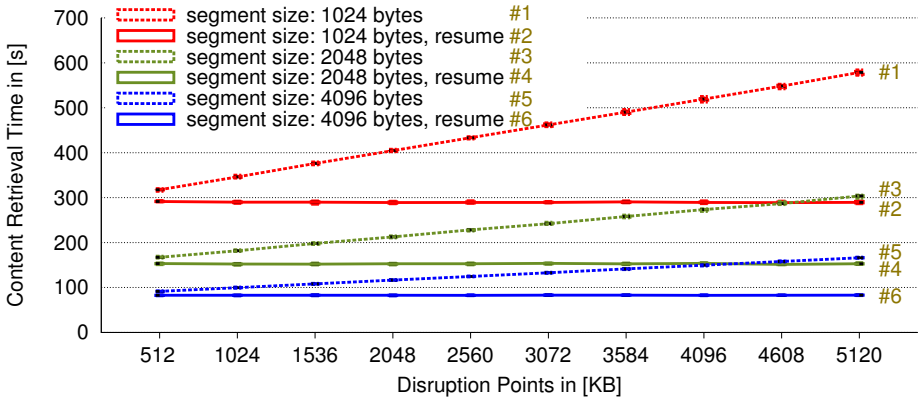


Figure 5.4: Content Retrieval Times of a 5 MB File via Broadcast Communication for Different Segment Sizes and Disruption Points.

Without resume operation, the effective content retrieval times increase the later content retrieval is disrupted because all segments need to be requested again in the second content retrieval. If the first download is disrupted immediately before content retrieval has finished, almost twice the amount of data needs to be transmitted requiring almost twice the amount of time. If resume operations are enabled, the content retrieval time is constant independent of the disruption points since received content segments are persistently stored at the requester. In our evaluations, there is always only one disruption and the second content retrieval is always successful. Therefore, the content retrieval time can be reduced by up to 100%. However, in the worst case, when nodes only meet for a short time and

content retrievals can not be completed at once, content retrievals without resume operation would never be completed.

The processing overhead for handling Meta Data is negligible. The size of the Meta Data depends on the size of the content name and the length of the publisher's public key digest, but it is usually significantly lower than 1 KB. Resume operations do not have a negative impact on content retrievals without disruptions. Evaluations of continuous content retrievals (without disruptions) did not show any increase in content retrieval times caused by Meta Data processing. The MTU of the network cards on the ALIX boards is 2274 bytes, thus, segment sizes larger than 1024 bytes result in packet fragmentation. Since we run CCNx on top of IP (using UDP as transport protocol), packet fragmentation is automatically handled on the IP layer. We can observe that despite fragmentation, larger segment sizes result in shorter content retrieval times due to smaller data and processing overhead since fewer messages need to be transmitted.

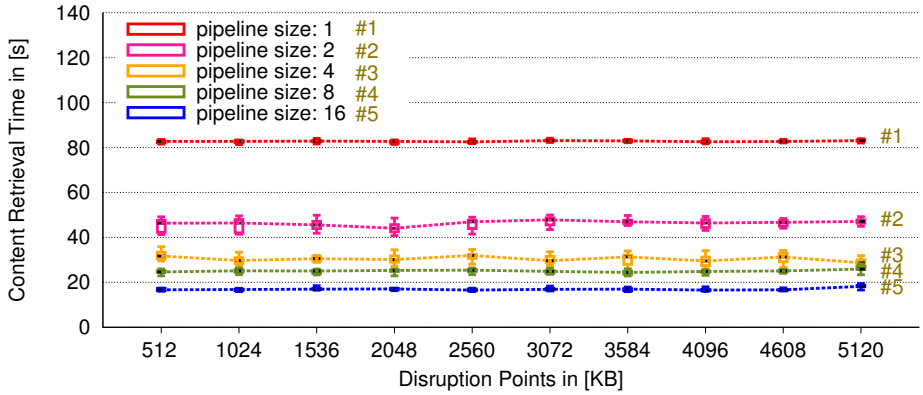


Figure 5.5: Content Retrieval Times of a 5 MB File via Broadcast Communication for Different Pipeline Sizes and Disruption Points.

In Figure 5.5 effective content retrieval times are shown for a 5 MB file using resume operations during broadcast communication. The segment size is set to 4096 bytes and we evaluate pipeline sizes of 1, 2, 4, 8, or 16. Content retrieval times decrease from a pipeline size of 1 to 16 by 80%. The largest relative decrease, i.e., 44% is observed when increasing the pipeline size from 1 to 2 because collisions do not affect subsequent message transmissions in the same restrictive way. When using a pipeline size of 1, no communication is performed in case of a collision until a retransmission is performed.

Figure 5.6 shows effective content retrieval times of a 5 MB file using resume operations during unicast communication. The segment size is set to 4096 bytes and the pipeline size is set 1, 2, 4, 8 or 16. The content retrieval times of unicast decrease compared to broadcast between 36% for a pipeline size of 1 and 53% for a pipeline size of 16. However, we can also observe that content retrieval times during unicast communication do not increase significantly for pipeline sizes lar-

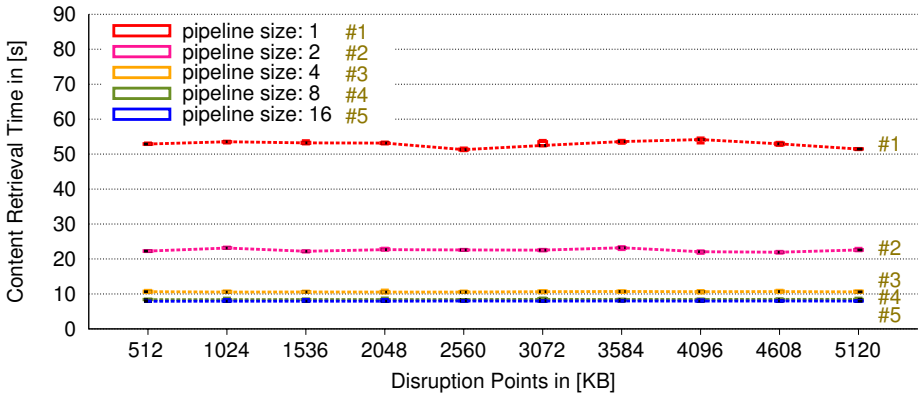


Figure 5.6: Content Retrieval Times of a 5 MB File via Unicast Communication for Different Pipeline Sizes and Disruption Points.

ger than 4. While content retrieval times decrease by 80% from a pipeline size of 4 to 1, they decrease only slightly more for larger pipeline sizes, i.e., by 84% (+4%) and 85% (+5%) for pipeline sizes of 8 and 16 instead of 1. Since unicast Data transmissions do not require forwarding delays, e.g., no broadcast delays to enable duplicate suppression, and unicast data rates are generally higher than broadcast data rates, Data can be returned quicker with unicast than with broadcast communication. Consequently, Interests can be transmitted faster and smaller pipeline sizes are not a limiting factor. For wireless mesh nodes with more processing power than PCEngines Alix 3D2 nodes, differences between pipeline sizes of 4, 8 and 16 may become even smaller (if not negligible) because Interest and Data messages can be processed and transmitted faster.

5.4.4 Power Measurements

We evaluate the power consumption of CCNx 0.6.0 on Alix 3D2 wireless mesh nodes during during wireless communication. The topology comprises three wireless mesh nodes. Two nodes exchange files via broadcast and a third node overhears the broadcast communication without being actively involved in the file exchange. The power consumption of all mesh nodes is measured with a Rigol digital multimeter [22]. In conformance with NDN node roles (see Subsection 2.1.1), we use the following traffic roles to classify the power measurements.

1. **Content Source** transmits Data from a local repository.
2. **Requester** transmits Interests to receive Data in return.
3. **Listener** overhears Data from the environment without requesting it. In case of Interest retransmissions, it may respond content from its cache.
4. **Idle mode** runs CCNx but without receiving or transmitting anything.

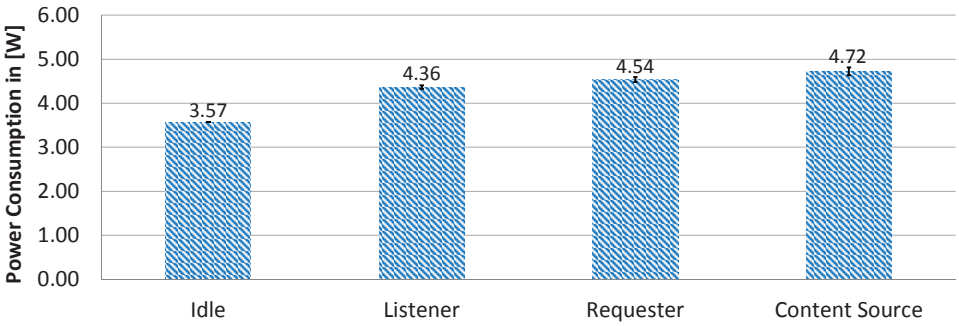


Figure 5.7: Power Consumption for Different Traffic Roles.

The average power consumption for all traffic roles during content retrievals is illustrated in Figure 5.7. The content source has the highest power consumption since it needs to fetch already signed content objects from the repository and transmit it to requesters over the wireless medium. Also, it receives Interests and must perform a longest-prefix match on the name. Power consumption of a requester is only slightly lower than for a content source, because it needs to transmit Interests and verify signatures of received Data messages. As expected, a listener consumes less power than a requester or a content source, because no Interests are transmitted and no signatures need to be verified. However, a listener still needs to parse and process all received Interest and Data messages as well as transmit occasionally a Data message in case of an Interest retransmission (due to a collision). Therefore, a listener node spends only 8% less power than a content source and only 4% less power than a requester. If content retrieval would be performed via unicast from the content source, listener nodes would not receive any messages and could remain in idle mode. Compared to idle mode, a listener has a 22% higher power consumption.

Figure 5.8 shows the measured average energy consumption when retrieving content objects of different sizes via broadcast or unicast communication. Both requesters and content sources require less energy for unicast than for broadcast communication due to a higher throughput. After content retrieval via unicast has finished, requesters and content sources can quickly switch back to idle mode, which has a lower power consumption. The energy overhead of a requester using broadcast instead of unicast communication decreases from 9% (0.25 MB file) to 7% (10 MB file) while the energy overhead of a content source decreases from 41% (0.25 MB file) to 28% (10 MB file). Since the energy consumption increases for larger files, the relative differences between broadcast and unicast decrease for increasing file sizes (smaller percentages). However, potential energy savings (absolute numbers) when using unicast instead of broadcast communication increase for increasing file sizes. Consequently, if broadcast communication is not required, e.g., because there is only one requester, unicast communication is advantageous due to lower energy consumption. However, if multiple nodes request content concurrently, broadcast communication may be more efficient.

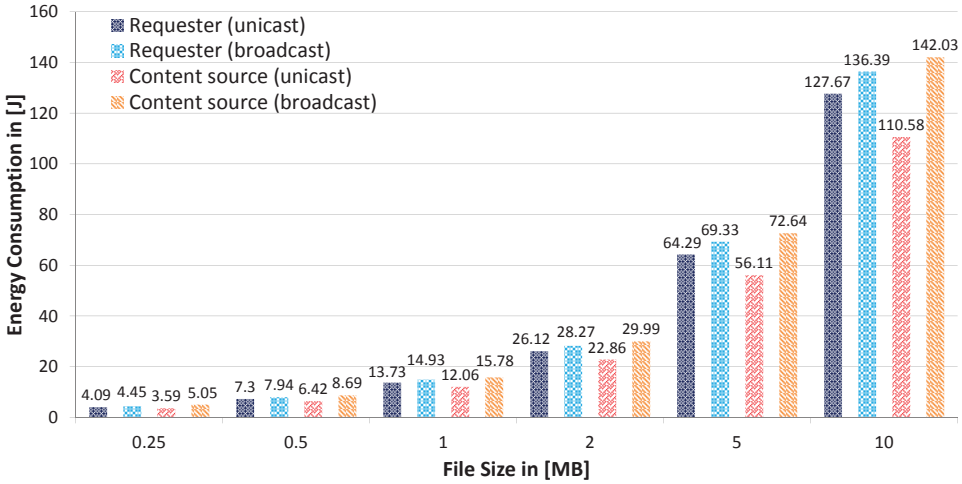


Figure 5.8: Energy Consumption for Content Retrievals of Varying File Sizes with Broadcast and Unicast Communication.

5.5 Conclusions

In opportunistic networks, connectivity to content sources may be intermittent and short such that content retrievals may not be completed at once and need to be resumed. Since NDN caches are only used for short-term storage, persistence is not guaranteed. Therefore, we have developed an application that enables requesters to store partial files persistently. Then, content retrievals can be resumed by maintaining Meta Data of received partial files, which is stored after disruptions. While strategies without resume operations may never be successful, resumed content retrievals result in constant content retrieval times independent of the disruption time from a content source. Evaluations have shown that processing and storage overhead is negligible and does not affect regular content retrievals in any way.

Since content sources are generally unknown, content retrievals need to be performed via broadcast. However, collisions during broadcast communication may result in long retransmission delays degrading the achievable throughput. Since opportunistic communication is performed via one hop, Interest lifetimes can be decreased to a low value to reduce retransmission delays. Evaluations have shown that decreasing Interest lifetimes can increase one-hop broadcast throughput with a pipeline size of 16 by a factor of up to 7.2 without significantly increasing the number of transmitted messages.

We have also investigated information-centric wireless unicast and broadcast communication. Evaluations have shown that unicast communication results in significantly faster content retrieval times and lower energy consumption at requesters and content sources. In addition, listener nodes that receive and process overheard broadcast messages have a 22% higher power consumption compared to idle mode. Such an increased power consumption can be avoided during unicast

communication because only active nodes (requesters and content sources) receive and transmit messages. Thus, for a known content source and only one requester, direct communication via unicast would be favorable. In Chapter 6, we investigate Dynamic Unicast for opportunistic one-hop communication, which dynamically configures unicast faces to neighboring content sources after an implicit content discovery via broadcast.

Chapter 6

Dynamic Unicast for Opportunistic One-hop Communication

6.1 Introduction

Information-centric networking (ICN) is promising for opportunistic communication because it enables dynamic communication based on the availability of content. Instead of detecting or maintaining connectivity to a specific host, content can be retrieved by name (or prefix) from any neighbor node that holds the desired content. Consider for example hitchhikers in rural or mountain areas that are interested in weather information. Instead of connecting via intermittent cellular communication to servers hundreds of kilometers away, it may be more accurate to get temperature information or thermal images from deployed sensors or heat cameras nearby. In addition, if there is no cellular coverage direct local communication may be the only feasible option to retrieve content.

ICN messages do not contain any source or destination identifiers and most wireless ICN works, e.g., [144, 217, 214, 39], use broadcast communication exclusively for all communication. They argue that overheard cached copies can be useful to satisfy future requests from other nodes. However, broadcast requests can trigger transmissions from multiple content sources resulting in collisions and duplicate Data transmissions. With caching in ICNs, every node that overhears broadcast transmission would automatically become a content source increasing content density and collision probability. Furthermore, listener nodes that do not (actively) participate in the communication have a 22% higher power consumption compared to the case when they do not receive broadcast messages (see Chapter 5). Yet, although wireless unicast communication results in shorter content retrieval times and lower energy consumption than via wireless broadcast communication, it is not possible to statically configure unicast faces to content sources in opportunistic networks because neighbor nodes may change dynamically.

In this chapter, we describe *Dynamic Unicast* [42], an approach that enables requesters to dynamically create unicast faces to opportunistic neighbors and, thus, support efficient content retrieval. Assuming a requester knows the name (or pre-

fix) of desired content, it can transmit one-hop broadcast requests for the name to detect whether desired content is available at neighbor nodes. If content is available (in which case a Data message is returned), the next requests can be directly addressed via unicast to the same content source until it becomes unavailable.

6.2 Dynamic Unicast

In this section, we describe *Dynamic Unicast* for opportunistic information-centric content retrieval. Dynamic Unicast is based on *implicit content discovery*: requesters can transmit Interests for a content name via one-hop broadcast to check whether a suitable content source is available. If no response is received, no content source is available, which - in terms of content retrieval - equals the lack of neighbor nodes. If a reply is received, a dynamic FIB entry can be created so that the content can be retrieved directly from the content source.

6.2.1 Implicit Content Discovery via Broadcast

In Chapter 4, we have described opportunistic content discovery algorithms to learn existing content names. If content names are known, a requester can find content at any nearby node by transmitting Interests for the content name via one-hop broadcast. We call this implicit content discovery because a requester can infer from a Data reply that a content source is available.

In NDN, Interests are forwarded based on configured prefixes in the FIB. Since it is not possible to configure all potential content prefixes in the FIB, we implement a *pass-through* for Interests from local applications, which means that these Interests are directly forwarded to a (wireless) broadcast face if no FIB entry is explicitly configured. If a content source is available, it will respond to broadcast Interests with a broadcast Data message in return. By that, all nodes in the vicinity get informed that there is a responding content source and duplicate Data transmissions can be avoided. After content reception, the requester can extract the content source's node ID and the name prefix, i.e., content name without segment number, to create a unicast face.

In this work, we consider the IP address of a node as local node ID. The IP address is not used for global routing but only as temporary content locator, which is not part of the content name protected by the signature, similar to locator-based mobility approaches [107, 168, 108, 125].

6.2.2 Broadcast Data Transmission

Broadcast Interests are addressed to all nodes in the vicinity. If multiple content sources are available, a single broadcast Interest can trigger Data transmissions from multiple nodes. In CCNx, every Data message is scheduled for transmission in a content queue. To avoid duplicate Data transmissions and enable requesters to synchronize to the same content source, broadcast Data transmissions are delayed

randomly. Then, if the same Data message has been received (from another content source) prior to its own transmission, content sources can cancel scheduled Data transmissions by deleting it from the content queue. In CCNx, the multicast delay is randomly selected in the interval $[DP, 3DP]$, where DP can be configured with the environment variable CCND_DATA_PAUSE _MICROSEC. We use the same interval for the broadcast delay and evaluate different values of DP.

6.2.3 Unicast Content Retrieval

After creating a unicast face to the content source, all subsequent requests can be addressed directly to it until the content is complete or a timeout has occurred. In the latter case, the Interest can be forwarded based on forwarding strategies described in the next subsection. Unicast faces should only be used as long as they are valid and should disappear if they are not used anymore. In CCNx, faces have limited lifetime and they age periodically every *update interval*. In addition to the lifetime, faces keep track of all messages that are received since the last update. Dynamic Unicast can exploit these existing procedures to avoid the deletion of expired unicast faces if messages are still being received over it. Therefore, the lifetime of dynamically created unicast faces can be set to a very short value of only a few seconds. The required changes to the CCND are minimal: only content name and node ID need to be extracted from Data messages to register a FIB entry if it does not already exist.

6.2.4 Interest Forwarding Strategy

The default CCNx Interest forwarding strategy rates all available faces and sends Interests first via the face that is considered best. If no answer is received within a certain time (shorter than the Interest lifetime), the same Interest is forwarded on the second best face and so on. To define a unicast-broadcast forwarding strategy, the ranking of outgoing faces needs to be adapted appropriately. We rank faces as follows. First, if there is an internal face, e.g., to a local repository, it has priority over all other faces. Second, if there is a unicast face, it is used next. New unicast faces are included at the beginning of the list and old faces expire quickly if they are not used anymore. Third, if no unicast face is available or no answer is received on the unicast face, the Interest is forwarded on the broadcast face to find new content sources.

If the content retrieval is not successful over any face, there is a timeout event. In this case, the Interest can be retransmitted for n times. To avoid collisions with retransmitted Interests from other nodes, we delay retransmissions by a random value between [0, $r \times t$], where r specifies the rth retry attempt and t is a slot time. If no content is received, after n retransmissions, no content source is assumed in the vicinity and a probing Interest is transmitted periodically after a probing Interval of *PI* seconds.

6.3 Evaluation

6.3.1 Evaluation Settings

In this section, we evaluate the performance of *Dynamic Unicast* using our NDN simulation framework based on OMNeT++ [205] (see Section 3.2 for more information). The parameters used in our evaluations are listed in Table 6.1.

Parameter	**Values**
interface	1 × IEEE 802.11g
data rate	unicast/broadcast: 2 Mbps
error rate	Nist error rate model
transmitter power	0.1 mW
signal attenuation	free space path loss
playground size	50m × 50m, 500m × 500m
mobility	Gauss-Markov, α = 0.9 speed: 1m/s
# requesters	1, 10, 30, 50, 70, 100
# content sources	1, 10, 30, 50, 100
segment size	4096 Bytes
content size	40 KB, 400 KB, 2 MB, 4 MB, 8 MB, 20 MB
data pause DP	10ms, 50ms, 100ms, 200ms, 300ms, 400ms, 500ms
content lifetime CL	60s, 90s, 120s, 150s, 300s, 600s
probing interval PI	4s
max. retransmissions n	2
slot time	10ms

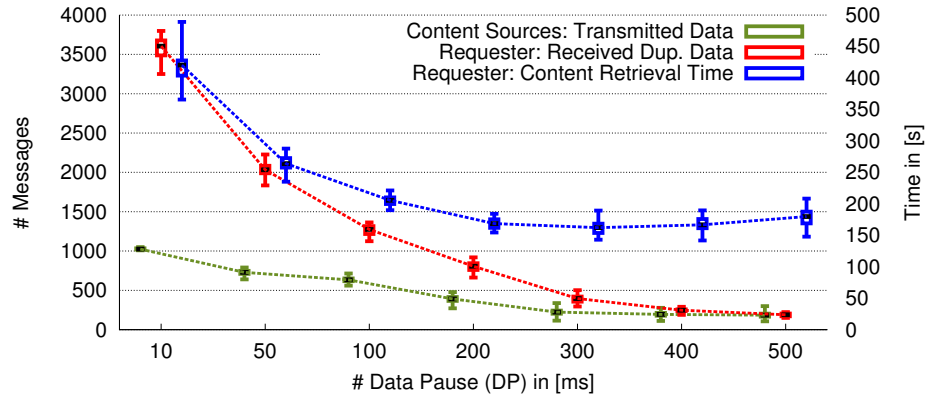
Table 6.1: Simulation Parameters for One-hop Dynamic Unicast.

With our configurations, nodes can overhear messages from other nodes up to a distance of 50m. We set the data rate for both unicast and broadcast to 2 Mbps to have fair conditions for comparison. In real systems the unicast data rate may be considerably higher, because it can be adapted between sender and receiver while broadcast data rates are set to the lowest supported data rate. The segment size corresponds to the effective NDN payload without any NDN headers and the content lifetime corresponds to the *freshnessSeconds* in CCNx defining how long a Data message stays in the content store. All network nodes have one broadcast face configured. Thus, they can ask their opportunistic one-hop neighbors for content. In case of multiple content sources, all content sources hold the same content objects. All requesters move based on the Gauss-Markov mobility model looking for content provided by content sources, e.g., sensors, that are statically placed at

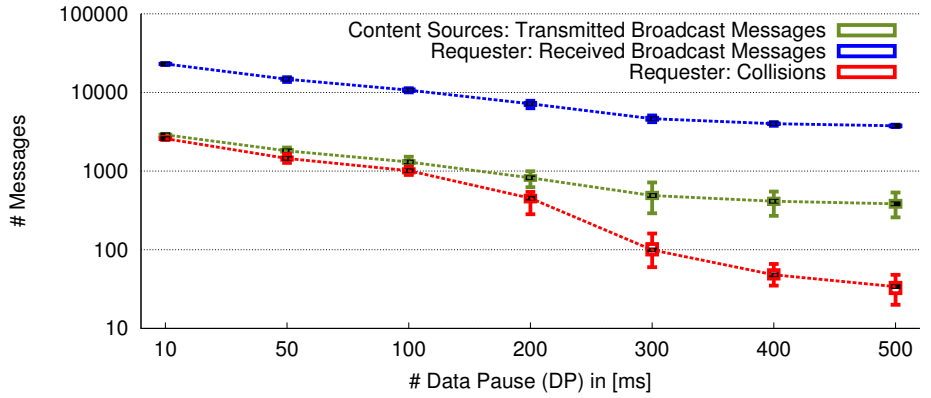
random positions within the playground. Requesters start their requests randomly within the first [1,100] seconds of a simulation by transmitting Interests in the same content name. Interests that time out are re-expressed twice with a delay based on the slot time as explained in Section 6.2.4. If nothing has been received, the next Interest is transmitted after a probing interval of 4 seconds.

6.3.2 Broadcast Delays

We first investigate the impact of broadcast delays using various DP values on wireless broadcast communication. We set the playground size to 50m $\times$ 50m, i.e. all nodes can see each other. We have only one requester node and multiple content sources with the same content. Figure 6.1a illustrates NDN messages and content retrieval times when using broadcast requests in case of 10 redundant content sources. The left y-axis shows the number of transmitted or received messages and the right y-axis shows the content retrieval time in seconds. The x-axis denotes different DP values. If too short delays are used, e.g., 10ms is the default



(a) NDN Messages and Content Retrieval Time.

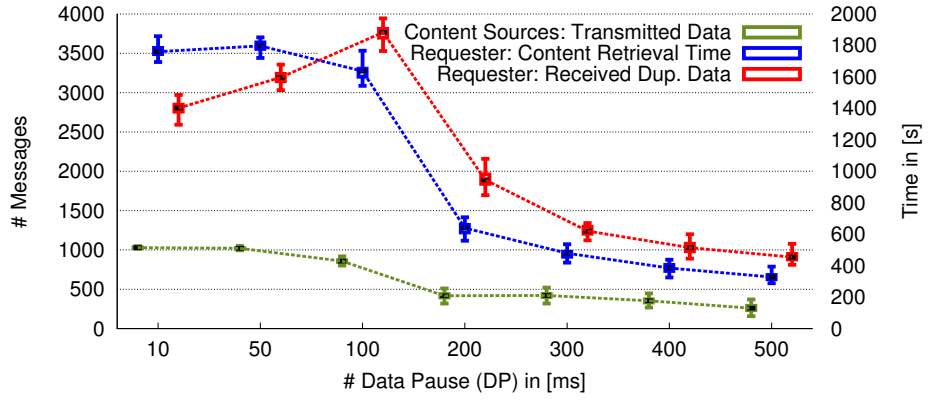


(b) Broadcast Messages and Collisions at the MAC Layer.

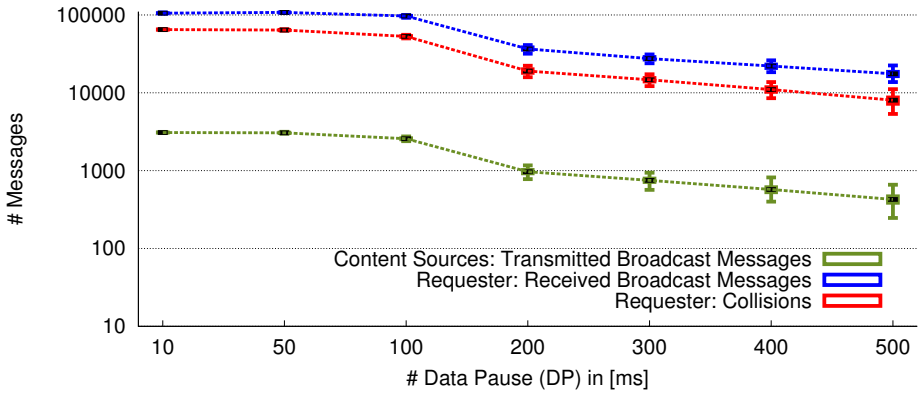
Figure 6.1: Broadcast Requests for 1 Requester and 10 Redundant Content Sources in a Playground of 50m $\times$ 50m. The Content Size is 4MB.

DP value in CCNx, content sources do not have enough time to detect concurrent Data transmissions and respond to the same Interests. As a result, a receiver receives many duplicate Data messages. In this example, a DP value of 200ms results in a content retrieval time that is 60% lower than with 10ms, i.e., 168.6s instead of 420.5s. At the same time, the number of received duplicate Data messages at requesters can be decreased by 78%.

Figure 6.1b shows broadcast messages and collisions at the MAC layer. With a DP value of 200ms, the number of collisions can be reduced by 83% compared to a DP of 10ms. Increasing DP values even more would result in slightly fewer collisions and duplicate Data messages but it cannot avoid them completely. Furthermore, the content retrieval time would slightly increase as well. The optimal DP value depends on the number of available content sources.



(a) NDN Messages and Content Retrieval Times.



(b) Broadcast Messages and Collisions at the MAC Layer.

Figure 6.2: Broadcast Requests for 1 Requester and 100 Redundant Content Sources in a Playground of 50m $\times$ 50m. The Content Size is 4MB.

We have also performed evaluations with more content sources and Figure 6.2 illustrates the results for 100 content sources. Figure 6.2a shows that content retrieval times decrease in case of 100 content sources for increasing DP values up

to 500ms. Thus, in case of 100 content sources, larger DP values result in better results than for 10 content sources. However, due to duplicate suppression, content sources have no means of knowing how many other content sources are available near requesters and can, therefore, not adapt DP accordingly.

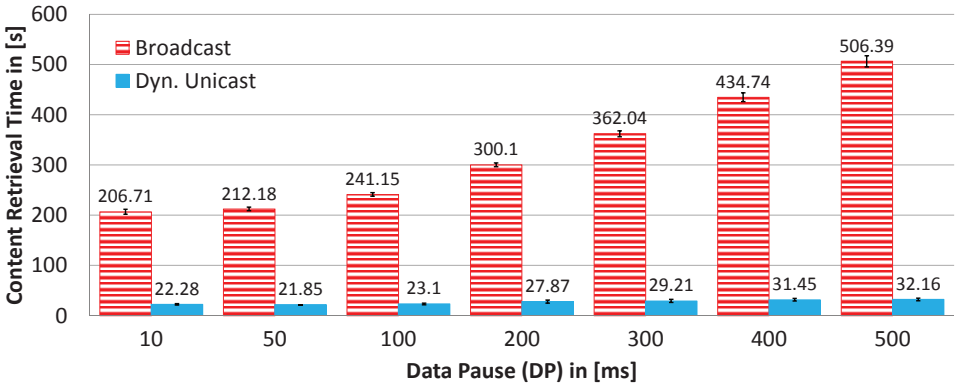


Figure 6.3: Average Content Retrieval Time of a 4MB File between two NDN Nodes: 1 Requester and 1 Content Source.

If only one content source is available, high DP values are disadvantageous because broadcast communication experiences unnecessary delays. In Figure 6.3, we show average content retrieval times between two NDN nodes: one requester and one content source. For DP values of 300ms or higher, content retrieval times via broadcast are more than 75% higher than with a DP of 10ms (minimum). However, large DP values have a smaller impact on Dynamic Unicast because it affects only broadcast messages.

In the rest of this chapter, we set DP to 200ms as tradeoff because it showed significantly better results than smaller values in case of multiple content sources, but does not result in significantly longer content retrieval times if only one content source is available. For the sake of comparability, we set DP to 200ms for both broadcast and Dynamic Unicast although Dynamic Unicast could use even larger DP values without affecting content retrieval times significantly.

In Figure 6.4 we compare Dynamic Unicast with broadcast communication for one requester and 10 content sources in a playground of 50m $\times$ 50m. Figure 6.4 shows that Dynamic Unicast can almost completely avoid duplicate Data transmissions and collisions. Furthermore, Dynamic Unicast can decrease content retrieval times by 87% from 169.8s to 22.7s. The content retrieval times are slightly shorter than with only one content source in Figure 6.3 because Data messages can be transmitted with the shortest random delay (among all 10 content sources). Consequently, retransmissions in case collisions can also be performed quicker.

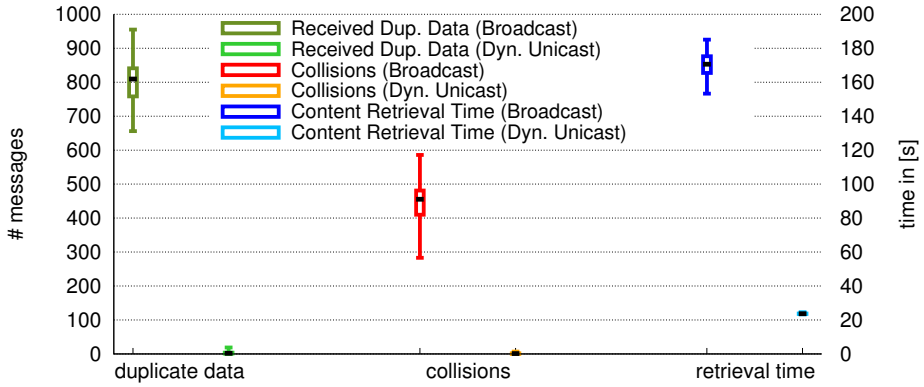
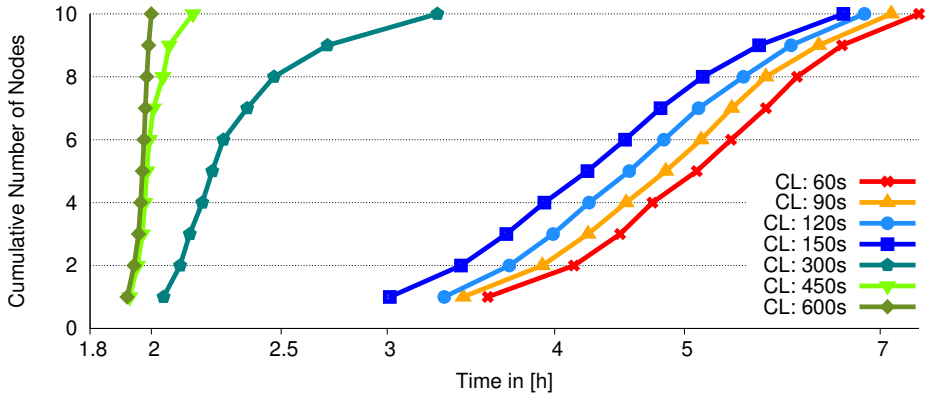


Figure 6.4: Broadcast vs. Dynamic Unicast for 1 Requester, 10 Content Sources and a DP value of 200ms.

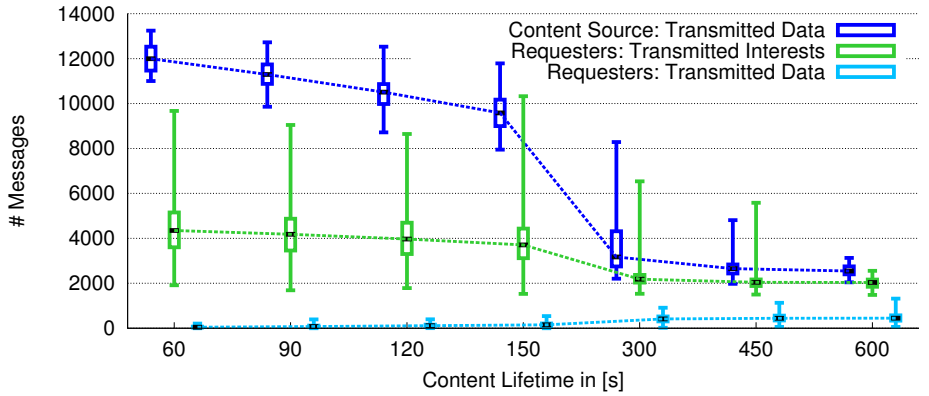
6.3.3 Content Lifetime for Broadcast Content Retrieval

If nodes overhear broadcast communication, they can serve future requests for the same content from the cache increasing content density. Broadcast NDN communication is, therefore, most efficient if Data messages do not disappear too quickly from the cache. In this subsection, we investigate the influence of cached Data messages on broadcast content retrieval. We evaluate different content lifetime (CL) values, i.e. time that Data messages remain valid in the cache. We use a network of 101 nodes: one static content source and multiple (1-100) mobile requesters in a 500m $\times$ 500m playground. Figure 6.5a shows cumulative retrieval times of 10 requesters when using content lifetimes between 60s and 600s. Recall that all requesters start their requests randomly within [0,100s]. Thus, for content lifetime values below 100s, content may have disappeared from caches before some requesters have started their requests for it. Even for content lifetime values slightly above 100s, content may disappear from caches before some requesters have seen a node with a cached copy. This observation is confirmed when observing the transmitted messages at requesters and content sources in Figure 6.5b. Up to content lifetimes of 150s, the content source needs to transmit around 3 - 4 times more Data messages than for a content lifetime of 600s. On the one hand, some requesters can avoid transmitting Interests in the same content because they have overheard the desired content earlier and still have it in their cache; on the other hand, requesters and listeners store content for a longer time and serve it to other requesters from their cache increasing content density.

In this scenario we obtain a pervasive content availability with a content lifetime of 600s, i.e., larger values (not shown in Figure 6.5b) do not result in any improvements. In scenarios with more requesters, pervasive content availability can already be obtained for lower content lifetimes because content is continuously requested and cached by more nodes resulting in a higher content density. For example, in scenarios with 30 requesters, we obtain a pervasive content availability



(a) Cumulative Retrieval Times for all 10 Requesters.



(b) Transmitted Messages by Content Source and each Requester.

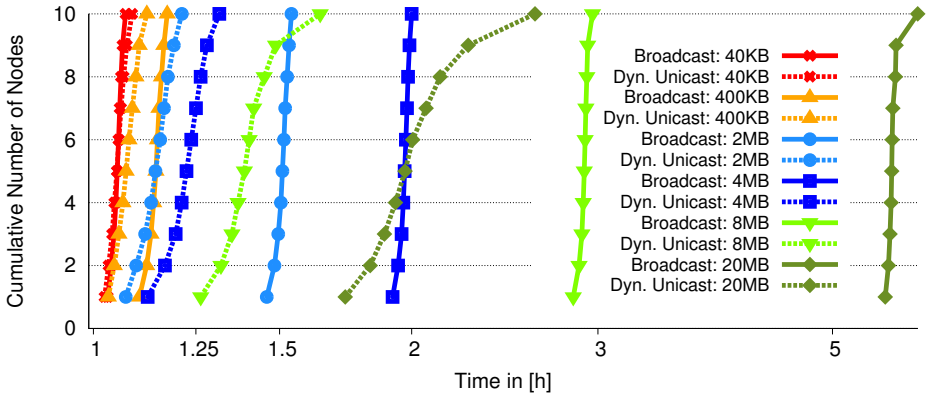
Figure 6.5: Cumulative Retrieval Times and Transmitted Messages for a Content Object of 4MB. There are 100 Mobile Nodes (10 Requesters, 90 Listeners) and 1 Static Source.

already for a content lifetime of 300s.

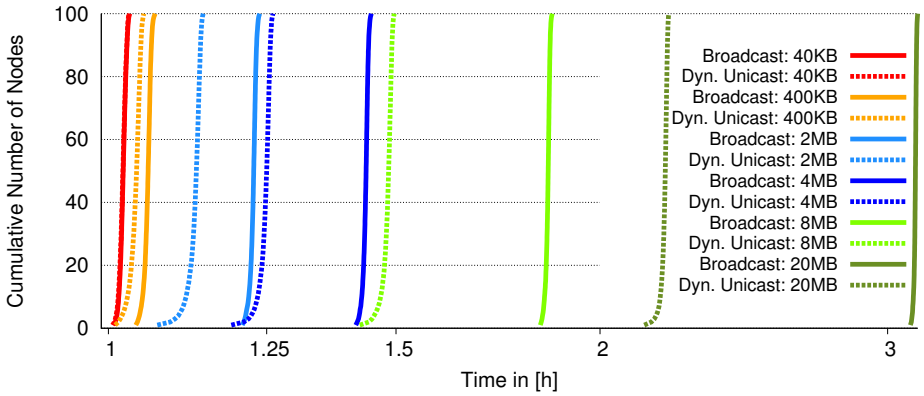
If Dynamic Unicast is used, only requesters receive content and other nodes can not overhear Data transmissions. To obtain the full benefits from NDN broadcast communication, we use a content lifetime of 600s when comparing broadcast to Dynamic Unicast in the next subsection.

6.3.4 Broadcast vs. Dynamic Unicast

In this subsection we compare Dynamic Unicast to broadcast communication in a playground of 500m $\times$ 500m. We use different content sizes between 40 KB and 20 MB, a content lifetime of 600s and a varying number of requesters. In Figure 6.6a we show the cumulative retrieval times for 10 requesters retrieving content objects of varying sizes using Dynamic Unicast or broadcast communication. For very small content sizes of 40 KB, i.e. 10 segments, broadcast content retrieval



(a) Cumulative Retrieval Times for all 10 Requesters.



(b) Cumulative Retrieval Times for all 100 Requesters.

Figure 6.6: Cumulative Retrieval Times of 10 and 100 Requesters retrieving Content Objects of Varying Sizes via Broadcast or Dynamic Unicast from 1 Content Source.

is slightly faster by 1% because such small content objects can spread quickly via broadcast among all nodes, i.e., broadcast delays have only small impact. However, the larger the content objects are, the better is the relative improvement of Dynamic Unicast. For content objects of 400 KB, Dynamic Unicast is already 4% faster, for 4 MB files the improvement is 34% and for 20 MB it is even 57%. For larger content objects, i.e. 8 MB and 20 MB, cumulative retrieval times of Dynamic Unicast have a less steep slope than for broadcast. This means that a few requesters need more time to complete content retrieval than others, while during broadcast, all requesters retrieve the content approximately at the same time. This is because content density is higher if content retrieval is performed via broadcast, i.e., all nodes receive the content while during Dynamic Unicast, some nodes may require more time until finding a node that can provide the content.

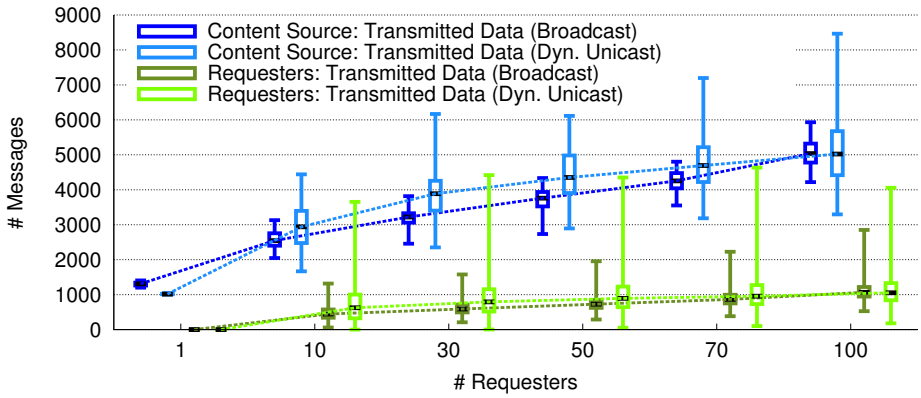
Broadcast communication should be particularly advantageous if many nodes request the same content at the same time. Figure 6.6b shows cumulative retrieval

times for 100 requesters retrieving content objects of varying sizes. Surprisingly, content retrieval via broadcast is still not faster than Dynamic Unicast. Already for a content object of 40 KB, Dynamic Unicast is slightly faster by 0.1%, for 4 MB files it is 13% and for 20 MB it is 30% faster. Because every requester also caches (requested) Data messages during Dynamic Unicast, content density is considerably higher with many requesters. As a result, the slopes of cumulative retrieval times during Dynamic Unicast are as steep as during broadcast.

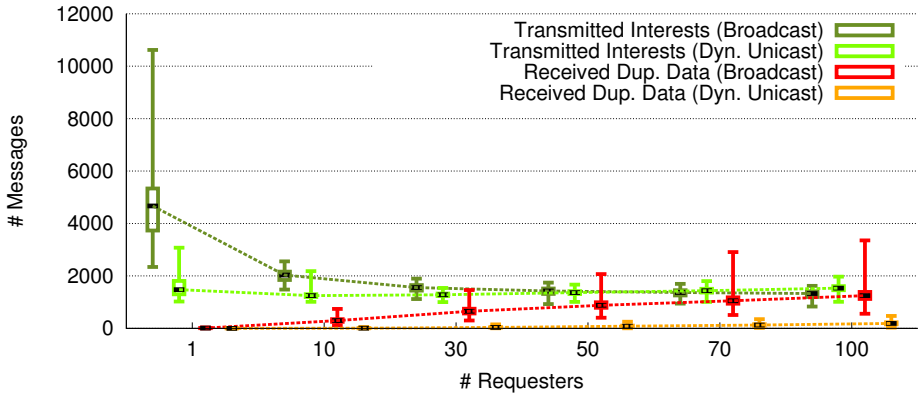
In Figure 6.7a we evaluate transmitted Data messages by the content source and the requesters for a fixed content size of 4 MB. The x-axis shows the number of requesters and the y-axis the number of Data messages. As expected, the content source needs to transmit more Data messages for Dynamic Unicast than for broadcast in case of multiple requesters. However, the increase is not as high as expected. For all evaluated requester scenarios, the average numbers of transmitted Data messages are at most 22% and the maximum numbers are at most 62% higher (30 requesters). For 100 requesters, the average number of transmitted Data messages is only 2% higher and the maximum value is only 43% higher. There are two reasons for this. First, during Dynamic Unicast requesters can also retrieve content from other requesters' caches (high content density), and second, the more nodes are requesting content concurrently, the higher is the probability of collisions and retransmissions during broadcast.

Figure 6.7b shows the number of transmitted Interest and received duplicate Data messages by requesters. The number of duplicate Data messages increases for an increasing number of requesters. In case of 100 requesters, the number of duplicate Data messages can be decreased by 85% when using Dynamic Unicast instead of broadcast. For broadcast, the number of transmitted Interests decreases with increasing number of requesters while for Dynamic Unicast, it increases slightly. However, up to 50 requesters, broadcast communication requires equal or even up to 3 times more Interests than Dynamic Unicast because content retrieval requires more time and may not be finished during short contact times to content sources. Thus, requesters may need to probe and meet the content source several times until retrieving the complete content. For 100 requesters, Dynamic Unicast requires on average only 18% more Interest messages than broadcast.

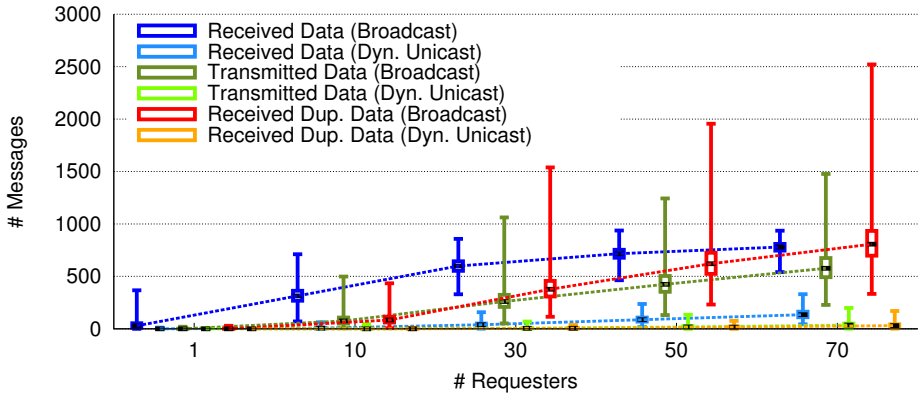
In Figure 6.7c we investigate the number of received and transmitted Data messages by listener nodes that do not actively provide or request content, but only provide overheard content from their caches. The x-axis shows the number of requesters and the y-axis the number of messages. Since there are no listeners in the 100-requesters-scenario, the x-axis shows only results up to 70 requesters. The more requesters are available, the more messages are received and transmitted by listener nodes. We can see that Dynamic Unicast can preserve resources on listener nodes because they receive, process and transmit considerably fewer messages. In case of 70 requesters, listeners receive and process on average 4.6 times more Data messages and even 23.4 times more duplicate Data messages for broadcast than for Dynamic Unicast. Consequently, listeners transmit on average 13.8 times more Data messages when requested via broadcast compared to Dynamic Unicast.



(a) Transmitted Data by Requesters and Content Source.



(b) Transmitted Interests and Received Duplicate Data by Requesters.



(c) Listener Overhead

Figure 6.7: Comparison of Dynamic Unicast and Broadcast for a Content Size of 4MB and a Varying Number of Requesters.

6.4 Conclusions

In Information-centric networks, requesters transmit Interests to receive Data messages from any available content sources in the vicinity. However, it may not be appropriate to keep this flexibility, i.e, to retrieve content from any reachable node, for every transmitted Interest. Since every node may be a potential content source and nodes do not know how many other content sources are available for a requester, they may need to use large broadcast delays to reduce duplicate Data transmissions and collisions. If every Data message is delayed, content retrieval may be considerably longer.

To solve this problem, we have presented *Dynamic Unicast* to dynamically adapt the transmission modes in wireless communication. We have shown that flexibility can be maintained by broadcasting requests only until a content source has been found (Dynamic Unicast). The required modifications to the NDN implementation are minimal (pass-through of Interests from local applications, extraction of content name and node ID to create a unicast face) and can exploit existing procedures to remove unused faces.

Dynamic unicast is clearly more efficient in case of single requesters and multiple content sources, where collisions and duplicate Data transmissions can almost be completely avoided. Surprisingly, our evaluations have shown that Dynamic Unicast is also up to 57% faster than broadcast in networks with multiple concurrent requesters. Nodes that cache content increase content density and every broadcast request triggers more duplicate Data transmissions and collisions. Therefore, broadcast communication is not as efficient as expected in terms of message efficiency if all nodes in the network request content concurrently. When using Dynamic Unicast, content sources send on average only 2% more Data messages and requesters send only 18% more Interests but receive at the same time 85% fewer duplicate Data messages compared to broadcast communication.

Furthermore, if only broadcast communication would be used, listener nodes that do not actively participate in the communication (as content source or requester) would need to process all received Interest and Data messages resulting in higher energy consumption (see also Subsection 5.4.4). In fact, the benefit of caching overheard content at listeners may be limited because listeners may overhear only part of the content and do not actively request missing segments. If listeners reply with overheard Data messages in response to broadcast requests in dense environments, it may result in many duplicate Data transmissions (and collisions). In addition, caches may hold content only for a short time due to their limited size (and cache replacement strategies). For example, if a node receives content at a rate of 54 Mbps (by overhearing content retrievals of multiple requesters), a cache of 2 GB would be filled in 5 minutes and would be completely overwritten within 10 minutes. Thus, if there are no requests in the same content in the meantime, there would be no benefit of caching.

Chapter 7

RC-NDN: Raptor Codes Enabled Named Data Networking

7.1 Introduction

Information-centric networking (ICN) can exploit the inherent broadcast capability of the wireless medium because content can be retrieved from any neighbor node that holds desired content. However, we have seen in Chapters 5 and 6 that ICN is not optimized for wireless broadcast communication because the same content copies are cached and transmitted by many nodes. As a result, limited wireless network bandwidth and cache sizes are not efficiently used because of many duplicate Data transmissions. While we have presented Dynamic Unicast as an alternative to broadcast communication in Chapter 6, we investigate an encoding approach called RC-NDN [51, 190] for increased wireless broadcast efficiency in this chapter.

There have been other research efforts [147, 225, 140, 212, 60, 172] that introduce advanced coding techniques such as network coding [32] in ICN architectures to deal with inefficiencies of the ICN content delivery process. Although reported gains are noticeable, these works deal only with static network environments. Furthermore, these works assume that all network nodes are able and willing to perform re-encoding operations with received packets, which may not be the case due to computational constraints. Moreover, since every NDN Data packet is signed by the content source, content can not be re-encoded transparently at intermediate nodes without requesters noticing, i.e., re-encoded content is equivalent to new content that requires new signatures. Thus, different from existing network coding approaches, RC-NDN considers one-hop wireless broadcast communication and encodes packets only at content sources, i.e., there is no re-encoding at intermediate nodes to avoid new signatures (see Subsection 2.7.2). Furthermore, we focus on networks characterized by significant mobility, e.g., moving cars or bicycles, in contrast to existing network coding approaches for ICN [147, 225, 140, 212, 60, 172] that focus on static network topologies.

The biggest challenge with the introduction of Raptor codes to NDN is the design of a protocol that is compatible with the NDN request-response scheme,

supports pipelining to enable multiple concurrent requests and exploits enhanced packet diversity from Raptor coding.

7.2 Raptor Coding for NDN

In this section, we present our proposed Raptor coding based NDN protocol. We first discuss the integration of Raptor codes into the NDN architecture before presenting required data structures and names. Finally, we present a sample RC-NDN request procedure.

7.2.1 Integration of Raptor Coding into NDN Architecture

The processing of Raptor coded packets can be included either 1) directly in the CCND or 2) in an application module above the CCND. Below we discuss the implications of each design.

1. **RC-NDN processing at CCND:** This means that packet encoding, collection and decoding should be done at the CCND. This approach would have the advantage that Raptor coding is transparent to the application. However, it is associated with two main drawbacks. First, an application would not know how many Interest and Data packets have been transmitted and received. Requesting new messages would, therefore, be the CCND's responsibility. To detect when content retrieval has been completed, an application would need to make regular checks. This means that the CCND would need to keep track of complete and partial content and provide it to applications upon request. Thus, the complexity of CCND message processing would increase significantly. Furthermore, received, decoded, and re-encoded Data packets would need to be stored in the cache, which is problematic because it is only a temporary storage. Hence, Data packets may be deleted before a sufficient number of packets for decoding has been received.

2. **RC-NDN processing at Application Layer:** This means that requesters would store received encoded packets temporarily in the content store. In addition, requesters can store received (encoded and decoded) packets at persistent storage controlled by the application. This approach follows the typical NDN application design where applications have full control when Interests are transmitted to receive Data packets. A disadvantage of this approach is that only requesters and content sources can perform encoding and decoding operations, while intermediate nodes only store received Data packets in and relay them from the cache.

As mentioned in Subsection 2.7.2, re-encoding packets at intermediate nodes requires new signatures, which may raise trust issues. Therefore, we decided to follow the second approach, i.e., to introduce Raptor codes as an application layer

module. We assume that encoded content has the same name prefix, which means that the content is created by the same publisher. In particular, we consider only coding of content segments from the same content object or stream and not coding of different content objects or content published by different content sources.

7.2.2 Data Structures

In the proposed RC-NCN architecture, we do not modify the original NDN Interest and Data messages. The information that is needed to request Raptor coded packets is included in the Interest name as explained in the next subsection. Original Data messages are serialized and encapsulated as payload in a new message structure called Raptor Data message. The structure of Raptor Data messages is similar to original Data messages and contains the same headers and signatures as regular NDN. Additionally, we append a Raptor coding header (RCH) to enable decoding at the requesters side. The RCH conveys the following information:

1. The seed used for generating the Raptor coded packets.
2. The number of generated Raptor coded packets N.
3. The number of original source packets K, i.e., segments.

7.2.3 Naming

Since message processing in NDN is based on names, identifiers of encoded messages should also be included in the name. To differentiate between regular NDN traffic and Raptor encoded packets, we have included an additional marker *_rc* after the file name $fname$. The proposed name structure is as follows:

$$/c_0/\cdots/c_n/fname/\_rc/version/e_n$$

where e_n stands for the nth combination of packets, i.e., encodedID. To avoid the reception of packet duplicates, the requester specifies the encodedID in the Interest name. The first combination e_0 equals the seed of the encoding and e_n is increased for every additional packet combination. Without specifying the encodedID e_n, multiple parallel Interest transmissions (pipelining), are not possible as explained in Subsection 2.7.2.

7.2.4 Request Procedure

We explain the Data request procedure with the help of Figure 7.1. The requester first transmits a general Interest in *<content_name>/_rc/* without encodedID. This Interest requests the lowest available encodedID of the content name at neighbor nodes. The requester receives for example a Data message with the name *<content_name>/_rc/ version/25034*. By looking into the RCH, the requester discovers both the seed that was used in the encoding process and the number of

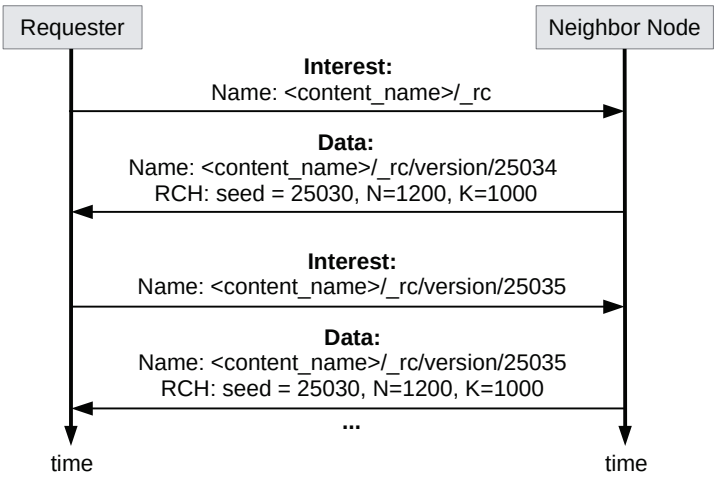


Figure 7.1: Data Request Procedure with RC-NDN. We refer to all name components before *_rc* with <*content_name*>.

encoded packets N. With this information, the requester can directly demand multiple encodedIDs simultaneously by increasing the encodedIDs, e.g., *25035*, *25036*, *25037*, etc., in the next requests. If there is a timeout, i.e., no reply to an Interest due to collisions or content unavailability, the Interest is not retransmitted but the next encodedID is requested by increasing the encodedID. When the maximum encodedID is reached, which is equal to the sum of seed and overall number of encoded packets N, the next encodedIDs to be requested start over from the seed with combinations that have not yet been received.

7.3 Evaluation

In this section, we evaluate the performance of the proposed RC-NDN architecture and compare it to original NDN. We use our NDN simulation framework (see Section 3.2), which has been implemented in OMNeT++ [205] to facilitate network setup and enable high scalability with respect to network size.

7.3.1 Simulation Scenario and Parameters

The simulation scenario is illustrated in Figure 7.2 and the simulation parameters are listed in Table 7.1. We use a Gauss-Markov mobility model with a node speed of *10 m/s*. Nodes can overhear transmissions up to a distance of 240m. All (mobile) requesters are looking for a content object of 4 MB, which is provided by a mobile content source. The requesters randomly start the request phase within the first [1,1000] seconds of the simulations by expressing Interests in the same content name, i.e., they ask for different encodings of the same content. Upon the reception of the first request, the content source encodes the content by means of Raptor coding and broadcasts a Raptor encoded message to all requesters that are in one-hop

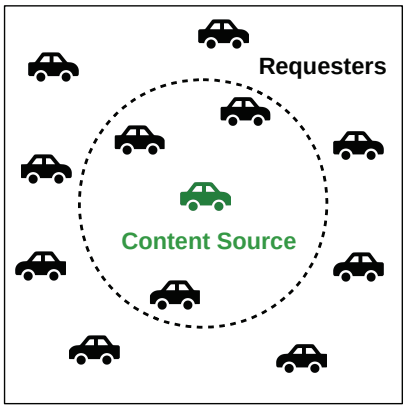


Figure 7.2: Simulation Scenario for RC-NDN.

Parameter	**Values**
interface	1 × IEEE 802.11g
data rate	2 Mbps
error rate	Nist error rate model
transmitter power	2.0 mW
signal attenuation	free space path loss
playground size (square)	500m, 1000m, 2000m 3000m, 4000m
mobility	Gauss-Markov, $\alpha = 0.9$ speed: 10m/s
# requesters	10, 30, 50, 70, 100
# content sources	1 (encoding Data upon first request)
segment size	4096 bytes
content size	1000 segments
content lifetime CL	60s
pipeline size	16
Interest lifetime	4s

Table 7.1: Simulation Parameters for RC-NDN.

distance. We use Raptor codes with 3GPP degree distribution [202]. Only content sources perform encoding, while requesters only decode content and do not perform any re-encoding operations. We delay content transmissions randomly with a broadcast delay in the interval [50ms, 150ms] to enable duplicate suppression. In our original NDN implementation, Interests that time out are retransmitted twice. If the Interests are not satisfied by the second retransmission, the next Interest is transmitted after a probing interval of 4 seconds. In RC-NDN, users request the next encodedID in case of a timeout. All reported results are averages over 100 simulations.

7.3.2 Content Retrieval Times

We first investigate the cumulative retrieval times for both RC-NDN and original NDN. Figure 7.3 shows the cumulative retrieval times of 100 nodes that request content using RC-NDN and original NDN in square playgrounds of various sizes. The reported playground values correspond to the side length that is measured in meters. The x-axis shows the time in seconds and the y-axis the cumulative number of nodes that have completed the content retrieval. In Figure 7.3, we can see that RC-NDN is considerably faster than NDN for all the examined playground sizes. For example, in a playground of 500m length, RC-NDN is 5.8 times faster than NDN, i.e., 990s instead of 5792s, while in a playground of 4000m, it is still 4.7 times faster, i.e., 1381s instead of 6525s.

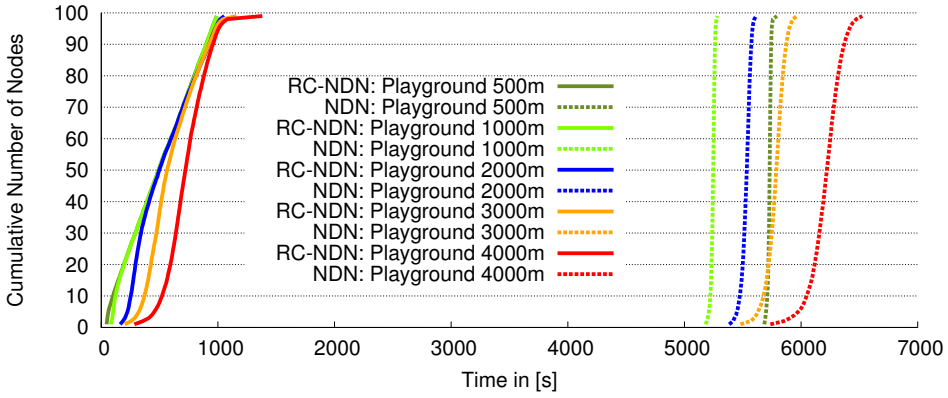
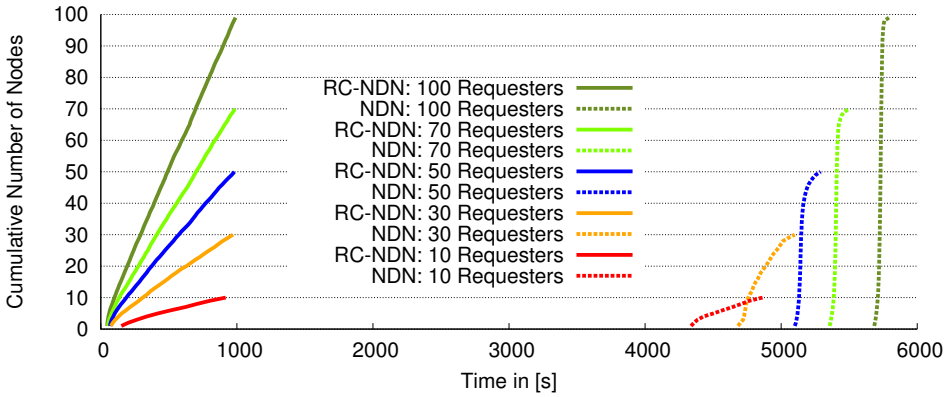


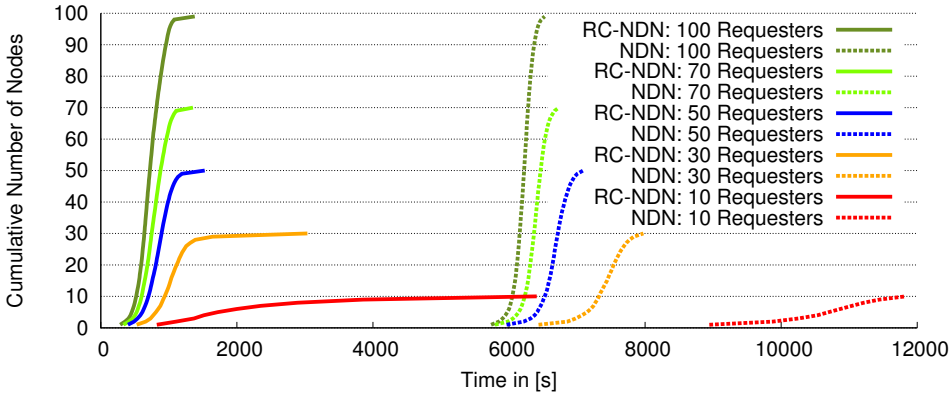
Figure 7.3: Cumulative Retrieval Times (NDN) or Decoding Times (RC-NDN) for 100 Requesters requesting the Same Content from a Content Source.

There are two main factors that limit NDN's content retrieval time in broadcast environments. First, the PIT table prevents the transmission of same Interests for the entire duration of an Interest lifetime (Interest aggregation). This means that a node in vicinity of a content source may not be able to transmit Interests when it receives the same Interests from another node before sending its own. As a consequence, due to the limited wireless transmission range, a node may overhear unanswered Interests from another requester preventing the transmission of its own Interests although a content source would be reachable. Differently from original NDN, in RC-NDN Interest transmissions are less likely to be blocked by the PIT since requesters do not necessarily request content sequentially. Second, in original NDN multiple nodes often simultaneously request the same segment. However, due to the duplicate suppression procedure explained in Subsection 2.1.1 only one segment transmission is performed, which limits the performance of NDN. In RC-NDN, nodes usually request different segments. Thus, multiple transmissions can be performed in parallel such that requesters can profit more from each others' requests. In Figure 7.3, we further observe that retrieval times of RC-NDN increase with increased playground sizes, i.e., there is an increase of the retrieval

time by 39% for a playground with side length 500m to a playground with side length 4000m. A similar behavior is noticed for the NDN protocol. However, if the requester density for NDN is too high, e.g., on the 500m-playground, content retrieval requires more time than on sparser playgrounds, e.g., 1000m or 2000m playgrounds, because of a higher collision probability and more duplicate Data transmissions when requesting and caching the same segments.



(a) Playground with Side Length of 500m.



(b) Playground with Side Length of 4000m.

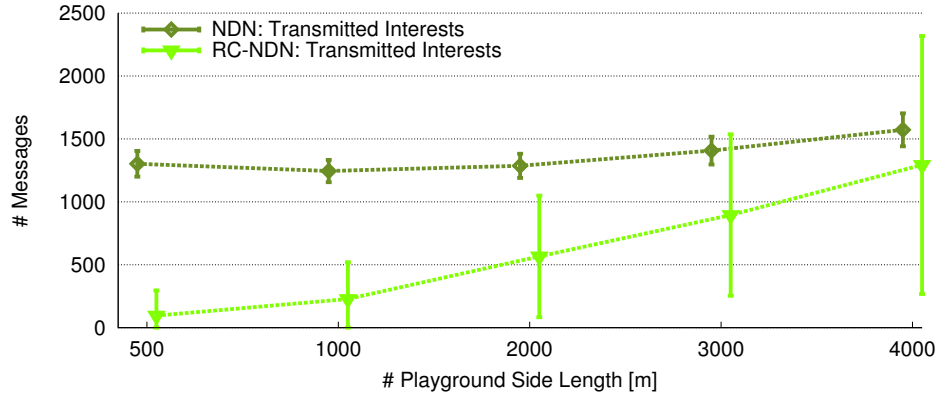
Figure 7.4: Cumulative Retrieval Times (NDN) or Decoding Times (RC-NDN) for Varying Numbers of Requesters.

In Figures 7.4a and 7.4b the cumulative retrieval times of RC-NDN and original NDN are presented when only a fraction of nodes request content in playgrounds sizes of 500m and 4000m. Figure 7.4a shows that when the node density is high, RC-NDN enables all requesters to finish content retrieval approximately at the same time. This occurs independently of the effective number of requesters. In contrast, content retrieval times increase with NDN for an increasing number of requesters, i.e., the curves are shifted on the x-axis. This means that the same number

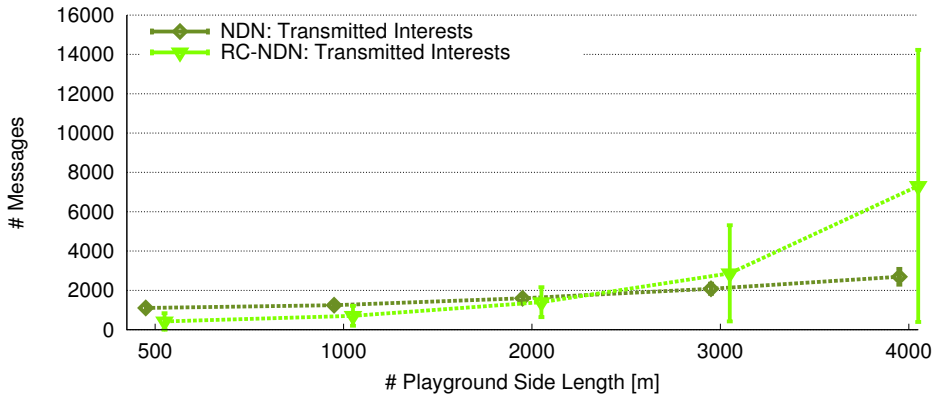
of requesters needs more time to retrieve content when there are more requesters in a network. From Figure 7.4b, we can observe that for low node density values (large playgrounds), the number of requesters has a bigger impact on the retrieval time. The more requesters there are, the faster is the content retrieval. Specifically with RC-NDN, content retrieval in networks with 100 requesters is 4.6 times faster than in networks with 10 requesters, i.e., 1381s instead of 6408s. In NDN, the retrieval times for networks with 100 requesters are 1.8 times faster than networks with 10 requesters, i.e., 6525s instead of 11829s. Thus, even in low density networks, RC-NDN performs better than NDN in terms of content retrieval times, however, the relative improvement is lower than in high density networks.

7.3.3 Transmitted Messages

In this section, we evaluate RC-NDN and NDN in terms of transmitted Interest and Data messages. The more messages are sent, the more energy is consumed for message processing and wireless transmission. In Figure 7.5a, we illustrate the number of transmitted Interests in different playground sizes when 100 requesters



(a) 100 Requesters



(b) 10 Requesters

Figure 7.5: Transmitted Interests by Requesters using NDN or RC-NDN for Different Playground Sizes.

are considered. We note that in dense environments RC-NDN results in a considerably lower number of transmitted Interests. For example, in a playground size of 500m, each requester in NDN needs to transmit 1303 requests (average value) to complete content retrieval, while using RC-NDN only 95 Interests (-93%) are needed on average. RC-NDN requesters can send fewer Interests because, due to increased request diversity, they can benefit more from each others' transmissions. Hence, we conclude that RC-NDN drastically reduces the required number of requests in dense environments. In bigger playgrounds, i.e., 4000m, the average number of transmitted Interests is still 18% lower with RC-NDN than with NDN, i.e., 1293 Interests instead of 1572, but the variation of RC-NDN is higher. The reason for this high variation is our selected Interest transmission strategy for RC-NDN, which does not adapt the pipeline size and continues to send the next Interests in case of timeouts. If no content source is in the vicinity of a node, multiple unnecessary Interests are transmitted, which are not answered.

Figure 7.5b shows the number of transmitted Interests in case of only 10 requesters. In a playground of 500m, requesters with RC-NDN send 62% fewer Interests than NDN. Thus, RC-NDN performs slightly worse compared to 100 requesters, but it still results in significantly fewer Interest transmissions than NDN. Yet, in larger playgrounds (i.e., 4000m) and 10 requesters, RC-NDN sends 2.7 times more Interests than NDN on average. This illustrates that RC-NDN may result in considerably more Interest transmissions than NDN in networks with low content density, where requesters may not regularly meet a content source.

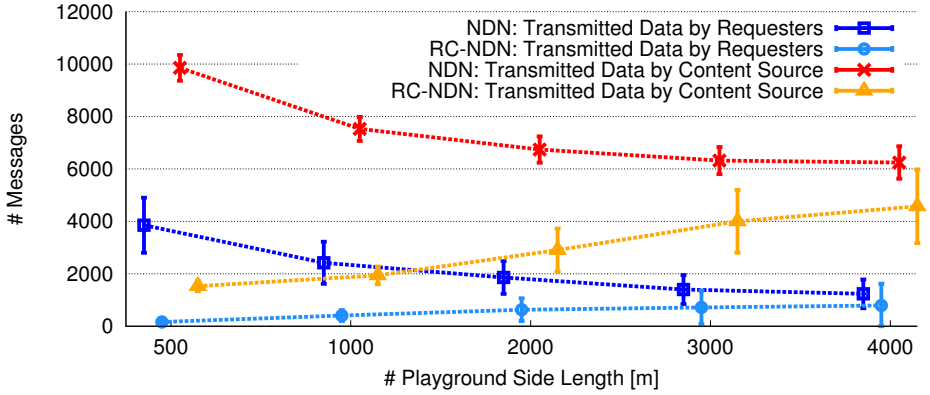


Figure 7.6: Transmitted Data Messages by NDN or RC-NDN for Different Playground Sizes (100 Requesters).

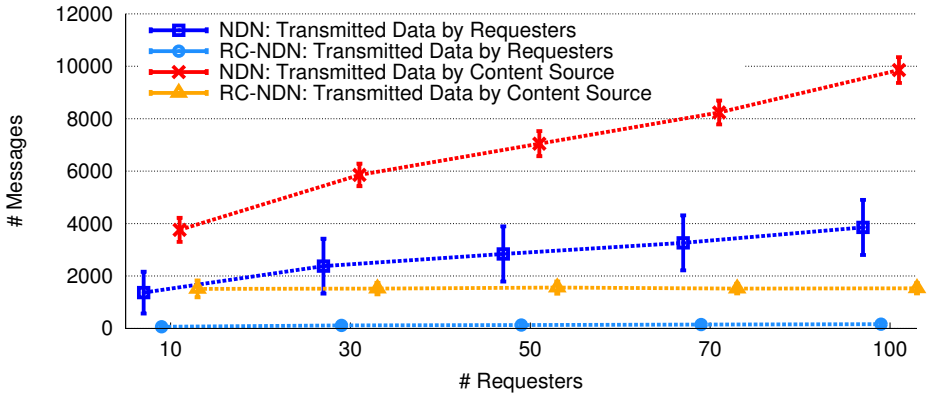
However, Interest messages are in general smaller in size compared to Data messages that contain content segments in the payload, i.e., 50 bytes vs. 4700 bytes. Hence, we next study Data transmissions in detail. In Figure 7.6, we depict the number of transmitted Data messages by requesters and content sources for various playground sizes that contain 100 requesters. Evaluations show that the number of Data messages transmitted by each requester using RC-NDN increases

slightly with the playground size. This is attributed to a lower requester density, i.e., requesters can profit less from each others' transmissions (overhearing) and need to provide content more often to other requesters. With NDN, the behavior is quite different. Since requesters using original NDN demand and store content sequentially, broadcast requests may result in many duplicate Data transmissions in dense environments. In sparse environments (large playgrounds), content density is lower resulting in fewer (duplicate) Data transmissions.

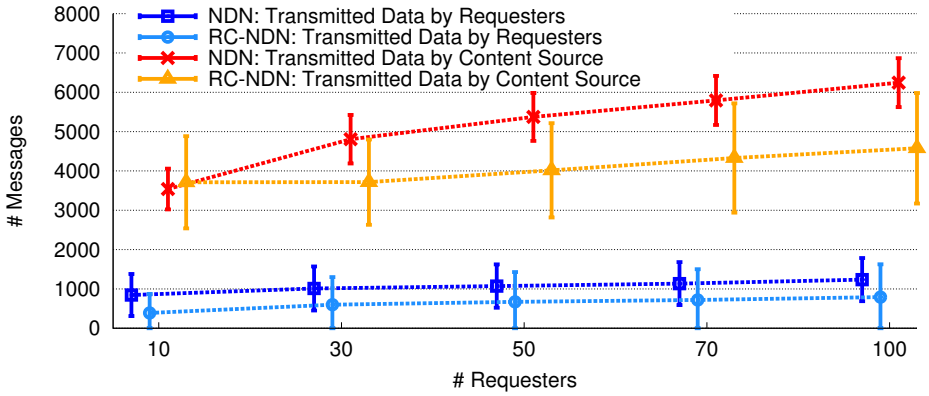
In all scenarios, RC-NDN requesters need to transmit fewer Data messages than NDN requesters. Specifically, in a playground of 500m, a requester using RC-NDN transmits on average 96% fewer Data messages than NDN, i.e., 162 instead of 3854. In a playground of 4000m, a requester using RC-NDN still transmits on average 36% fewer Data messages than NDN, i.e., 791 instead of 1236. Due to increased diversity of Data transmissions, content sources can send fewer Data messages as well. For a playground of 500m, a content source using RC-NDN transmits 85% fewer Data messages than NDN, i.e., 1531 instead of 9857, and for a playground of 4000m a content source sends approximately 27% fewer Data messages than NDN, i.e., 6246 instead of 4577.

In Figures 7.7a and 7.7b, we present the number of Data messages transmitted by requesters and content source in playgrounds of 500m and 4000m for a varying number of requesters. As Figure 7.7a shows, in dense environments the number of transmitted Data messages by content source and requesters using NDN increases with increasing number of requesters, while it increases only slightly with RC-NDN. For 10 requesters, content source and requesters transmit on average 60% and 95% fewer Data messages with RC-NDN compared to NDN. In case of 100 requesters, content source and requesters transmit even 85% and 96% fewer Data messages. These results show that RC-NDN can significantly reduce the number of Data transmissions in dense environments, and by that, reduce the number of (unnecessary) duplicate Data transmissions.

In playgrounds of 4000m with 10 requesters, a content source using RC-NDN needs to send on average only 5% more Data messages than NDN due to a lower requester density and requesters not overhearing each other. Recall that RC-NDN requesters need to receive $\epsilon \times K$ additional packets before decoding content (see Subsection 2.7.3). However, even in sparse environments, if the number of requesters increases to 30 requesters or more, a content source sends up to 27% (for 100 requesters) fewer Data messages with RC-NDN compared to NDN.



(a) Playground with Side Length of 500m.



(b) Playground with Side Length of 4000m.

Figure 7.7: Transmitted Data Messages by Content Source and Requesters using NDN or RC-NDN in Playgrounds with Side Lengths of 500m and 4000m for Varying Numbers Requesters.

7.4 Conclusions

In this chapter, we have studied the application of Raptor codes to NDN to decrease content retrieval times in wireless networks characterized by high mobility. From the evaluation, we have seen that the introduction of Raptor codes in NDN protocols offers large gains compared to original NDN in terms of content retrieval times and the number of transmitted Interest and Data messages. Larger gains are noticed in dense networks even when the number of requesters is very small. Interestingly, the performance of RC-NDN increases with the number of requesters. The superior performance of RC-NDN over NDN is attributed to the inherent inefficiency of NDN to deal with a high number of requesters. In such cases, the PIT in NDN blocks multiple Interests for the same segments. Therefore, due to limited wireless transmission ranges, a node that overhears (unanswered) Interests from

another requester may not transmit its own requests even if a content source would be available. Additionally, due to the sequential requesting policy in NDN, many redundant Data messages are transmitted and cached. This leads to an increased number of collisions, duplicate Data transmissions and eventually to longer content retrieval times. The improved performance of RC-NDN renders it particularly appropriate for dense urban environments, such as train stations or sports stadiums, where multiple requesters can benefit from each other when requesting popular content.

Although RC-NDN offers noticeable gains in sparser environments, these are smaller than in dense networks. In case of only a few requesters, RC-NDN may even perform worse than NDN in terms of Data transmissions because requesters may not overhear each others' transmissions and need to receive slightly more Data messages to successfully decode content. In addition, requesters may transmit considerably more Interests than with NDN. This is due to our Interest transmission strategy that does not adapt the pipeline size when timeouts occur, but proceeds with the next encoded message. We argue that Exclude filters may increase the size of Interest messages and prevent pipelining. However, there is a trade-off between Interest and pipeline size. Hence, in case of multiple timeouts in a row it may be better to adapt the transmission strategy to send one large Interest with Exclude filters than many small Interests. As soon as content is received, multiple Interests can be transmitted again (pipelining).

We close this section with the following remark. Although we have considered one-hop communication and content is Raptor encoded only at the content source (recall that network coding at requesters or intermediate nodes is not permitted to avoid new signatures and establish transitive trust models), we expect that the introduction of network coding capabilities to requesters and intermediate nodes could further enhance the performance, especially in mobile and wireless multi-hop networks.

Part III

Information-Centric Wireless Multi-hop Communication

If requesters and content sources are not within one-hop distance to each other, Interest and Data messages need to be forwarded via intermediate nodes. In Chapter 8, we describe multi-hop broadcast communication based on overheard Data transmissions and preferred forwarders. As alternative to broadcast communication, we investigate Dynamic Unicast for mobile wireless multi-hop routing in Chapter 9. To increase throughput during information-centric wireless multi-hop routing, we study adaptive Interest lifetimes in Chapter 10. Wireless multi-hop routing may not be possible in case of intermittent connectivity. Therefore, we present agent-based content retrieval for delay-tolerant networks in Chapter 11. Finally, in Chapter 12, we describe a persistent caching extension that can be combined with short-term caching and stores popular delay-tolerant content at edge routers near requesters.

Chapter 8

Information-Centric Wireless Multi-hop Communication via Preferred Forwarders

8.1 Introduction

Wireless community networks [93] offer a cost-effective option to interconnect users where communication infrastructures by commercial Internet providers are not available. If one of the users has access to an Internet gateway, it can share this connection with other users in the community network; but even if there is no connection to the Internet, users can still communicate and share content among each other. Since wireless community networks are not managed by a central authority and are dynamic in nature (wireless interferences, local congestion and node failures), ICN is a promising networking approach to direct Interests towards available content sources. If requesters can not find desired content within wireless one-hop distance, Interests need to be forwarded towards content sources potentially multiple hops away. To enable ICN routing, forwarding tables need to be configured. For static ICN Internet communication, prefixes can be configured in FIBs with the help of routing protocols [216, 112] and nearby copies can be found via cache synchronization [219] or redundant random searches [75]. However, in wireless community networks, node connectivity and content availability may change over time. In this case, proactive periodic exchange of routing information for every possible content prefix may overload the network.

Most information-centric wireless routing protocols [144, 214, 41, 100, 101] are based on broadcast communication to find content copies at nearby nodes and improve scalability in case of multiple concurrent requesters. Yet, these approaches have limitations: most of them [213, 144, 214, 41, 41] do not consider FIBs to route Interests and they either consider only one-hop communication to wired infrastructures [213], consider routing on a high-level [206] with endpoint identifiers [144, 41] or forward messages based on location information included in names [214] or appended to MAC messages [100, 101].

However, when adding endpoint identifiers to messages [144, 41], communication is not strictly information-centric anymore, which means that ICN loses its flexibility, i.e., it matters from which node content is received. Consequently, handover mechanisms are required similar to current host-based communication and Interest aggregation may be less efficient in case of redundant communication between different endpoints. Furthermore, routing protocols that append location information to messages [214, 100, 101] impose additional requirements on the hardware, which may not be met by all devices, e.g., devices need to know their exact location and need to have sufficient processing capabilities to route every single message based on a node's location.

In this chapter, we investigate a different approach [43] without endpoint identifiers and location information. Nodes overhear broadcast Data transmissions to configure prefixes in FIBs. Then, we describe and evaluate different strategies for prefix registration and evaluate optimizations to improve multi-hop communication in case of multiple potential forwarders.

8.2 Multi-hop Communication with Overhearing

Information-centric multi-hop communication requires a mechanism to configure forwarding prefixes in the FIB and selecting forwarders out of multiple potential options to avoid duplicate transmissions. We consider the extraction of content names from overheard broadcast Data messages and include it temporarily into the FIB such that Interests can be forwarded towards the content source. To enable overhearing without configured FIB entries (and in the absence of other nodes requesting content), Interests from local applications, i.e., running on the device, can always be forwarded to a broadcast face to probe for content in the environment. We call this *pass-through* of local Interests. Interests from other nodes are only forwarded if a matching FIB entry exists. To configure forwarding entries towards persistent content sources, Interests forwarded via pass-through always ask for content from repositories but not from caches, i.e., forwarding entries can then be configured based on returned Data messages. Additionally, we add a hop counter to Interest messages to limit the propagation of Interests. A hop counter in Interest messages is also added in the CCNx 1.0 protocol [149].

8.2.1 Prefix Registration and Forwarding Strategies

Since Interest messages can not be forwarded on the face from where they have been received, wireless multi-hop communication requires at least two (broadcast) faces for communication: one for receiving and one for transmitting / forwarding messages. To receive messages on either face, both broadcast faces need to be established. In addition, to forward Interests towards content sources, eligible prefixes need to be configured to those faces in the FIB. Thus, if a node receives Interests on one face, they can be forwarded on the other face.

We describe three different strategies for prefix registration below. All forwarding strategies require the configuration of two broadcast faces to receive content. The difference between the strategies is how prefixes are associated to those faces.

1. Two Static Forwarding Faces (2SF)

Every prefix needs to be configured on both broadcast faces such that Interests can be forwarded on the opposite face on which they were received. In this strategy, we do not apply any overhearing mechanisms. Instead, all prefixes are statically configured to both broadcast faces of all nodes, e.g., in a configuration phase by network administrators, because no feedback can be received from the network. This may result in large FIB tables to route all potential prefixes (even if the corresponding content is never requested). In addition, it is more difficult to adapt to changing topologies since routing protocols are required for periodic updates. Please note that we use 2SF only for reference purposes.

2. Two Dynamic Forwarding Faces (2DFo)

In this strategy, overheard prefixes are added to both broadcast faces similar to *2SF*. However, in contrast to *2SF*, prefixes are only added dynamically to the FIB if the content is available, i.e., it has been overheard. The approach uses pass-through and hop counters. Similar to Chapter 6, a pass-through forwards Interests from local applications via broadcast face if no FIB entry is configured. However, different from Chapter 6, we use two broadcast faces instead of one (to enable multi-hop communication). Then, forwarding via pass-through works as follows: if an Interest on face 1 has not been answered, the Interest is automatically transmitted via face 2. To avoid that forwarders discard it as duplicate Interest, the requester needs to change the nonce in the retransmitted Interest.

3. One Dynamic Forwarding Face (1DFo)

In contrast to *2DFo*, with *1DFo* overheard prefixes are only registered at the face on which Data transmission has been overheard, but not at both faces. Similar to *2DFo*, pass-through and a hop counter is used with *1DFo*.

We explain forwarding with the help of Figure 8.1, which shows four nodes and their transmission ranges (dashed circles). Nodes in transmission range of the content source can broadcast an Interest on face 1 (pass-through) and all nodes in range of the content source, e.g., forwarder 1 in Figure 8.1, can overhear the Data transmission to configure the prefix in the FIB, i.e., they can configure the prefix on face 1 denoted by the solid gray circle in the figure. Nodes further away, e.g., forwarder 2 in Figure 8.1, need to send Interests on face 2 so that nodes near the content source, e.g., forwarder 1, can forward it via face 1. Then, since they receive Data on face 2, they can configure the prefix on face 2 denoted by the white circular ring. As shown in the figure, only two broadcast faces are required to forward

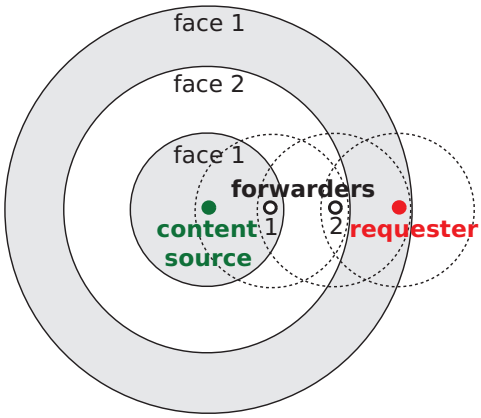


Figure 8.1: Multi-hop Forwarding via Alternating Faces.

Interests from a requester multiple hops away by alternating the forwarding faces in the nodes along the path.

8.2.2 Prefix Configuration via Overhearing

Prefixes are configured by extracting the content name from overheard (broadcast) Data transmissions. Only new content is registered in the FIB, i.e., only if the Data message is not already stored in the content store or the local repository. To limit message processing, only every n-th Data message can be processed by applying a modulo operation on segment numbers of received Data messages. Hence, since content is requested sequentially, no additional state information is required to count the number of received messages.

If the 2DFo strategy is used, the prefix is registered to both broadcast faces. If the 1DFo strategy is used, the prefixes are only registered to the face from which the content has been received and to 1 face at maximum. If no prefixes are configured, Interests from local applications can be forwarded via pass-through. If an Interest needs to be forwarded via pass-through, the header field *AnswerOriginKind* is set to 0, which means that no cached answer from content store is accepted.

8.2.3 Interest and Data Forwarding

To avoid duplicate Interest (and Data) transmissions, the propagation of Interest messages needs to be controlled. The basic idea is the following: every node delays Interest transmissions randomly by an *Interest Forwarding Delay (IFD)*. Based on overheard Interest and Data messages, the IFD is increased or decreased. If a forwarder continuously receives Data to its transmitted Interests, it assumes to be a *preferred forwarder*.

Interest Forwarding Delay

To avoid duplicate Interest transmissions, every Interest is forwarded with an IFD, which is randomly selected within a specified interval $[0, IFD_{max}]$. Based on overheard Interest or Data messages, IFD_{max} is modified. If Interests are transmitted and Data is received in return, IFD_{max} is decreased. By this, we can minimize the influence of large forwarding delays and reduce the time until the next Interest can be sent resulting in higher throughput. If the same Interest is overheard from another node before it has been transmitted, the Interest transmission is delayed by a waiting time WT, i.e., *delayed transmission*, and IFD_{max} is slightly increased. If another Interest is overheard during WT, delayed Interests are discarded. Otherwise, if no Data transmission is overheard, delayed Interests are forwarded.

The values are configurable but in our current implementation, we set IFD_{max} initially to 100ms. We halve IFD_{max} if a Data message is received in return and no other Interest has been overheard. If an Interest has been overheard IFD_{max} is slightly increased by 1ms. In this work, we use a fixed Interest lifetime IL of 4s and delay overheard Interests randomly by WT within the Interval $\left[\frac{IL}{4}, \frac{IL}{4} + IFD_{max}\right]$. However, more accurate waiting times WT may be based on previously measured round-trip-times (RTTs).

Preferred Forwarders

Interests are forwarded based on the IFD. At some point, a forwarder may receive a Data message for which no Interest has been forwarded locally. Then, the forwarder knows that there is another node that forwarded the Interest faster. It assumes itself to be *non-preferred* and adds an additional fixed delay (in our implementation 100ms) to every Interest transmission. Due to the fixed delay, non-preferred forwarders will only attempt to forward Interests if preferred forwarders have not retrieved content already. If preferred forwarders have moved away or their Interests collide, non-preferred forwarders can forward their Interests (*delayed transmission*). To avoid unnecessary switching, e.g., due to occasional collisions, a non-preferred forwarder can only become preferred, if it has performed a delayed transmission for N (in our implementation: N=3) subsequent times.

Data Transmission

Similar to NDN, broadcast Data transmissions experience a broadcast delay to enable duplicate suppression. Furthermore, received Interest messages can be satisfied from cache. However, there is one modification to NDN: received Data messages are only further propagated if the corresponding Interests have been forwarded and have not only been received or scheduled for transmission.

8.2.4 Data Structures

Our proposed approach uses the existing NDN data structures, namely the FIB, the PIT and the CS. For efficient multi-hop communication, we use the following two additional hash tables.

Interest Table (IT)

An IT entry is identified by the complete Interest name (including segment number). It points to the corresponding PIT entry and includes additional information for Interest forwarding. The information includes the number of times the Interest has been overheard, a list of faces where the Interest has been forwarded and multiple flags to specify whether: 1) a pass-through was required, 2) the Interest has been sent or only scheduled for transmission and 3) a delayed transmission was required due to an overheard Interest. An IT entry is created whenever a new Interest is received and forwarded. Cleanup can be performed at the same time as for the PIT, i.e., when an Interest has been consumed or has expired depending on the Interest lifetime.

Content Flow Table (CFT)

The CFT is used to maintain the current IFD_{max} value and to remember whether a node is a preferred node or not. A CFT entry is identified by the Interest prefix (complete Interest name without segment number) and it is created the first time when an Interest has been forwarded for an existing FIB entry. Similar to FIB entries, CFTs have an expiration time, which can be regularly updated and checked at the same time as the FIB expiration. In our implementation, we set the expiration time to 500s but also larger values are possible. CFT entries keep track of existing forwarding faces and unsuccessfully forwarded Interests: if S (in our implementation: S=16) subsequent unsuccessful Interest transmissions have been detected over a specific face, the corresponding dynamically created FIB entry can be unregistered and removed from the CFT. If the CFT does not specify another outgoing face, the complete CFT entry is deleted.

8.3 Evaluation

We performed evaluations by simulations using our NDN framework based on OMNeT++ [205]. More information on the simulation framework can be found in Section 3.2.

8.3.1 Simulation Scenario

The evaluation topology is shown in Figure 8.2. The small solid circles represent network nodes and the dashed circles denote different network partitions. All network nodes inside a network partition can directly communicate with each other. The white solid circles are only part of one partition and can not directly reach nodes in other partitions. Black solid circles in the intersections between partitions are potential forwarding nodes that can relay between white solid nodes. In total, there are 150 nodes: 25 nodes within each partition (white circles) and 10 potential forwarders (black circles) in each intersection A, B, C, D, M of neighboring partitions. The content source is always placed in partition 1 and a requester (white node) in partition 1 requests content such that forwarders (black nodes) in the intersections A, B and M can overhear it. This step is only used to populate the FIB in partition 1.

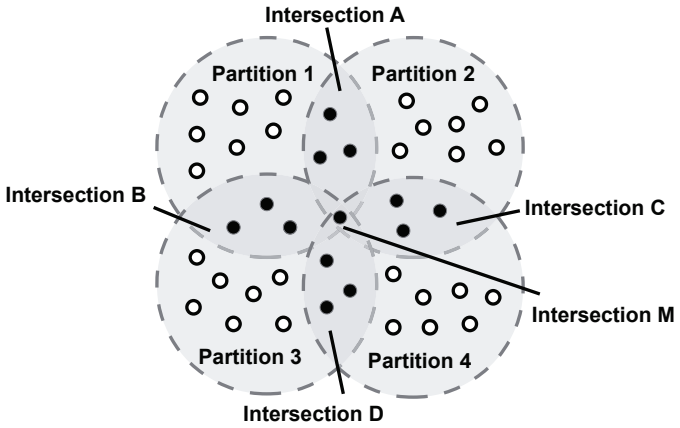


Figure 8.2: Network with 4 Partitions: all nodes in the same partition can see each other but not nodes from other partitions.

The common evaluation parameters are listed in Table 8.1. All network nodes have one IEEE 802.11g network interface and two broadcast faces configured to enable multi-hop communication. The broadcast data rate is set to 2 Mbps. The segment size corresponds to the payload of a Data message, i.e., without NDN headers, and the content lifetime defines how long content stays valid in the content store, i.e., corresponding to *freshnessSeconds* in CCNx. The data pause DP (for the broadcast delay) is set to 100ms. We evaluate multiple requests from different partitions and the request interval denotes the time interval between the requests. For example, for a request interval of 2000s, content from previous requests can not be found in the nodes' caches anymore because it has expired (content lifetime is significantly shorter than the request interval). To evaluate competing concurrent flows, where part of the content can be found in the cache, we also evaluate a request interval of 30s. While the content source is in partition 1, requesters are always selected among the "white nodes" within the partitions 2, 3 or 4 to en-

Parameter	Value
interface	1 × IEEE 802.11g
data rate	broadcast: 2 Mbps
segment size	4096 bytes (RTS/CTS disabled)
content size	4 MB (1000 segments)
pipeline size	16
content lifetime CL	600s
data pause DP	100ms
request interval RI	2000s, 30s
Interest lifetime	4s (default)
prefix validity time in FIB	40'000s

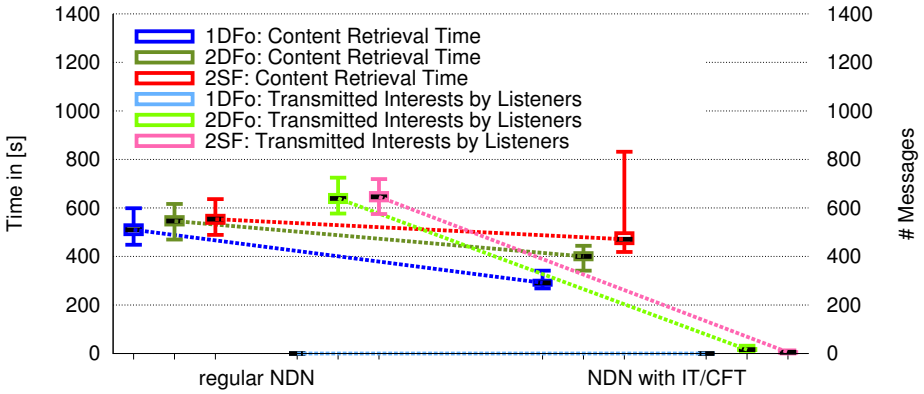
Table 8.1: Simulation Parameters for Multi-hop Broadcast with IT/CFT.

force multi-hop communication. We use the default CCNx Interest lifetime of 4 seconds. For the 1DFo, 2DFo and 2SF strategies, message processing is implemented as described in Section 8.2. The lifetime of overheard prefixes in the FIB can be much larger than content lifetimes of Data messages in the cache because name prefixes are significantly smaller than cached content objects (multiple Data messages with payloads). Therefore, we set the lifetime of overheard prefixes in the FIB to 40'000s but delete entries after 16 (consecutive) unsuccessful Interest forwardings. For every configuration, 100 simulation runs have been performed.

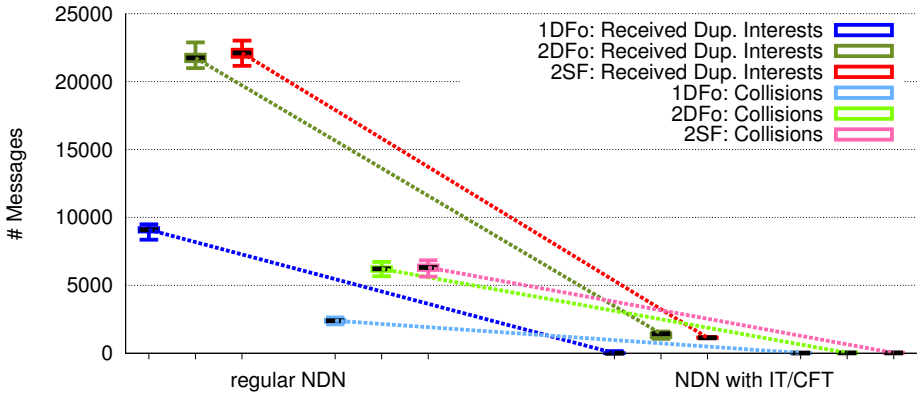
8.3.2 Multi-hop Communication with Interest Table

In this subsection, we evaluate the performance gain of IT/CFT compared to regular NDN message processing. To populate FIB entries for 1DFo and 2DFo strategies, a requester in partition 1 requests content from a content source in the same partition. After 2000s, a requester in partition 2 is requesting the same content. Although the content does not exist in caches anymore, FIB entries in intersections A and M are still valid. In Figure 8.3a, we compare 1DFo, 2DFo and 2SF in terms of content retrieval times and transmitted Interests. The content size is 4 MB, which corresponds to 1000 segments. The left y-axis shows the requester's content retrieval time in seconds and the right y-axis the transmitted Interest messages by listeners, i.e., nodes that do not actively request content, in partition 2. Figure 8.3a shows that the 1DFo strategy results in 6% shorter content retrieval times than 2DFo and 8% shorter content retrieval times than 2SF with regular NDN. However, when applying IT/CFT, content retrieval times of 1DFo can be drastically reduced by 47% compared to regular NDN. The differences compared to other strategies become larger with applied IT/CFT, i.e., 1DFo performs 27% faster than 2DFo and even 38% faster than 2SF.

For regular NDN, 1DFo performs better than 2DFo and 2SF because all "white"



(a) Requester's Content Retrieval Time and Listeners' Transmitted Interests in Partition 2.



(b) Received Duplicate Interests and Collisions at Content Source in Partition 1.

Figure 8.3: Comparison of Regular NDN to NDN with IT/CFT.

nodes in partition 2 have only one face configured. Since Interests can not be forwarded on the face of reception, Interest forwarding in partition 2 can be effectively avoided with 1DFo. As Figure 8.3a shows for the 1DFo strategy, listener nodes in partition 2 that do not have a direct connection to the content source do not forward Interests but only forwarders (black nodes in Figure 8.2) do. When applying IT/CFT, transmitted Interests by listeners could be reduced for 2DFo and 2SF because listeners realize that they are non-preferred nodes. However, because forwarding via two faces results in more forwarded Interests and overheard Interests, Interest forwarding delays IFD_{max} of 2DFo and 2SF are higher than with 1DFo, which explains the longer content retrieval times.

Figure 8.3b shows the number of received duplicate Interests and collisions at the content source in partition 1. With regular NDN, the number of duplicate Interests is high because all forwarders in intersections A and M receive the Interests at the same time and forward it almost immediately before overhearing Interest forwardings of other nodes. In the 1DFo strategy, every forwarder transmits on

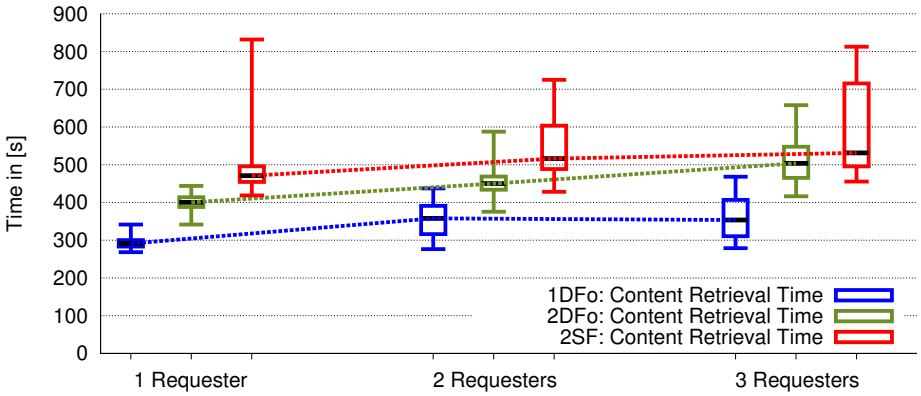
average 636 Interests, while with the 2DFo and 2SF strategies 934 Interests are forwarded by forwarders and 440 Interests by listener nodes in partition 1. As a result, a content source receives 58% fewer duplicate Interests and experiences and 61% fewer collisions with the 1DFo strategy compared to 2DFo and 2SF. With IT/CFT, the number of duplicate Interests for all three strategies can be drastically reduced by at least 93% compared to regular NDN because non-preferred forwarders transmit only a few Interests, however, duplicate Interest transmissions cannot be avoided completely.

We would like to emphasize that multiple forwarders are beneficial for multi-hop forwarding with IT/CFT. Recall that forwarders delay Interest transmissions if they overhear the same Interest. Because these forwarders can transmit Interests via *delayed transmission* if they do not overhear Data transmissions, the communication can recover faster from collisions on the path. For example, in the current scenario with 20 potential forwarders in intersections A and M, content can be retrieved 21% faster with 1DFo compared to a scenario with only 1 forwarder.

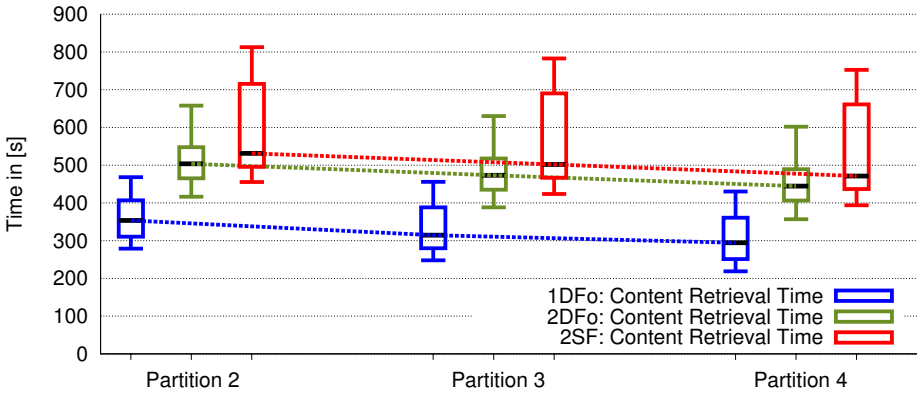
8.3.3 Multi-hop Communication with Multiple Requesters

In this subsection, we evaluate the influence of multiple concurrent requesters to multi-hop communication. We only show evaluation results with IT/CFT because it performed better than regular NDN. In addition, we only show evaluation results with concurrent requesters in different partitions, i.e., partitions 2, 3, 4. Concurrent requesters in the same partition would receive the content without any efforts due to broadcast communication and caching. Similar to the last subsection, a requester in partition 1 requests content such that FIB entries at forwarders can be configured. The concurrent requests start after a delay of 2000s to ensure that no Data messages are cached. Then, requests in partitions 2, 3, 4 (in this order) start with an interval of 30s, which means that some content can be retrieved from cache but requesters quickly catch up resulting in concurrent request streams.

Figure 8.4a shows the content retrieval time of a requester in partition 2 if there is one concurrent request from partition 2, two concurrent requests from partitions 2 and 3 or three concurrent requests from partitions 2, 3 and 4. Figure 8.4a shows that a requester in partition 2 needs 22% more time with 1DFo if there is a concurrent request from partition 3. Please note that only nodes in intersection M can receive Interests from both partitions 2 and 3, while nodes in intersection A only receive Interests from partition 2 and nodes in intersection B only receive Interests from partition 2. Two concurrent requesters in partition 2 and 3 can, therefore, result in two competing request streams, which may result in slightly longer Interest forwarding delays IFD_{max} and consequently, slightly longer content retrieval times. Another requester in partition 4 does not further increase content retrieval times with 1DFo. Thus, content retrieval times do not grow linearly with increasing number of requesters. All concurrent requesters finish their content retrieval at the same time, which means that effective content retrieval times of requester 2 and 3 are approximately 30s and 60s shorter as Figure 8.4b shows.



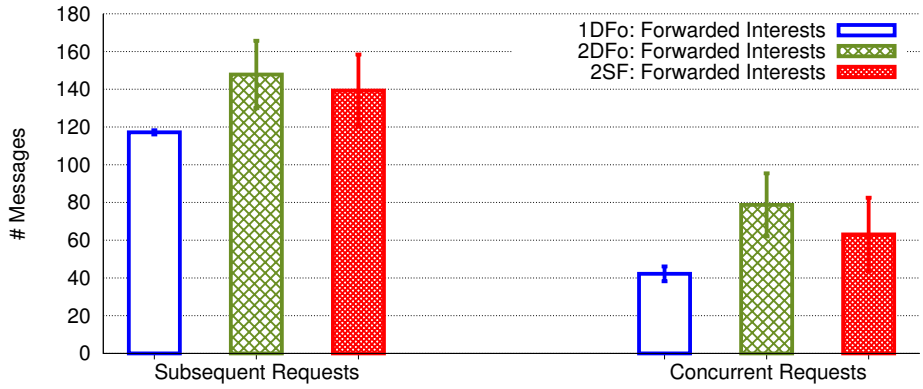
(a) Content Retrieval Time of Requester in Partition 2 for Additional Requesters in Other Partitions.



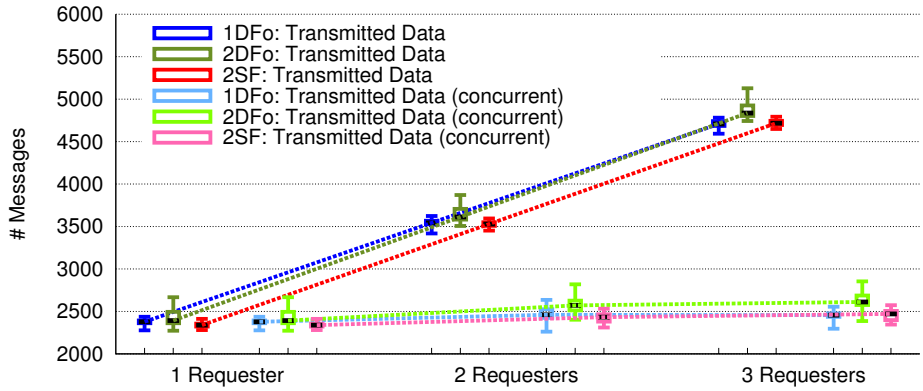
(b) Content Retrieval Time of 3 Concurrent Requesters in Different Partitions.

Figure 8.4: Content Retrieval Times for Concurrent Requests.

Figure 8.5a shows the average number of transmitted Interests by forwarders in intersections A, B, M when using the 1DFo, 2DFo and 2SF strategies with three requesters in partitions 2, 3 and 4. The bars on the left side show transmitted Interests if the requests are subsequent with an interval of 2000s such that requesters cannot profit from each other. The bars on the right side show transmitted Interests if requesters concurrently retrieve content with an interval of 30s. Figure 8.5a illustrates that with 1DFo, the transmitted Interests can be reduced by 64%, i.e., to almost a third of the traffic, in case of concurrent requests compared to independent (subsequent) requests. This demonstrates that 1DFo can efficiently handle multiple concurrent request streams. For 2DFo and 2SF, the reduction is smaller because forwarders can forward Interests on two faces. Forwarding in NDN is continuously updated based on past experience and every node forwards Interests first over the face that it considers best. Different nodes may have different views on which face is ”best”. If requesters in partitions 2 and 3 transmit Interests on different faces, intermediate nodes occasionally forward Interests for the same segments from both partitions. Therefore, the reduction is only 47% with 2DFo and only 55% with



(a) Transmitted Interests by Forwarders (average) for 3 Requesters.



(b) Transmitted Data by Content Source in Partition 1.

Figure 8.5: Transmitted Messages for Subsequent and Concurrent Requests.

2SF. The absolute number of transmitted Interests is a slightly larger with 2DFo compared to 2SF because IDF_{max} is slightly larger with 2SF.

Figure 8.5b shows transmitted Data messages by the content source in partition 1 for multiple subsequent and concurrent requests. The number of transmitted Data messages is proportional to the number of received Interest messages at the content source. Figure 8.5b illustrates that all three strategies can limit the number of transmitted Data messages for concurrent requests at an approximately constant level. The 2DFo strategy has the highest increase of transmitted Data messages of 9% from 1 to 3 requesters. For 2SF, the increase is 6% and for 1DFo, it is only 3%.

8.4 Discussion

In this chapter, we have enabled multi-hop forwarding via two alternating broadcast faces. It would be possible to use only one forwarding face and enable FIB forwarding on the same face of reception. However, this would have implications on PIT processing, e.g., routing loops could be introduced if Interests would be for-

warded on the same face from where they were received. To avoid inefficiencies, the number of Interest forwardings would need to be limited. If each received Interest would only be forwarded once (minimum for multi-hop), it would be similar to our 2DFo strategy. Our evaluations have shown that 1DFo outperforms 2DFo, which justifies the use of two forwarding faces that may be artificially created on the same wireless interface.

One may argue that nodes need to overhear content (for prefix registration) before Interests can be forwarded over multiple hops towards a content source. However, the same applies to existing approaches that use endpoint identifiers [144, 41], i.e., each intermediate node needs to know the content source in order to forward content. Also, routing schemes based on location [100, 101] need to learn the location of the content source, e.g., via flooding to configure GeoFaces [101] towards the content source. Thus, for our approach we see three options to enable forwarding. First, one could advertise name prefixes near content sources, but in contrast to current routing protocols [216, 112], nodes should not learn the entire topology but only the immediate neighborhood via broadcast. Second, one could add an additional message flag to enable flooding over a limited number of hops (pass-through) if no forwarding entry is configured. Third, agent-based content retrieval (see Chapter 11) may be used for content retrieval in delay-tolerant networks over multiple nodes.

The lifetime of overheard prefixes in the FIB is still subject to more investigations and may depend on network dynamics. We believe that negative acknowledgments (NACKs) as proposed in [228] to indicate unavailability of content would not be a good option in broadcast environments: although one node can not forward the Interest, another might be able to do so. Listening to NACKs from single nodes may, therefore, not be meaningful. Therefore, in this work, we tracked content responses for transmitted Interests and deleted forwarding entries if a consecutive number of Interests has timed out.

8.5 Conclusions

We have described a new approach for information-centric wireless multi-hop communication based on overhearing. In contrast to existing approaches, no endpoint identifiers are used but overheard prefixes are registered in the FIB. Evaluations have shown that the current NDN architecture does not work well for wireless multi-hop communication if multiple paths (forwarders) are available. Every node that receives an Interest and has a forwarding entry, may forward it resulting in many duplicate Interest transmissions. We have shown that additional data structures to maintain Interest forwarding delays can reduce the number of duplicate Interest transmissions by 93% and reduce the number of collisions by 61%.

Furthermore, we have implemented two dynamic prefix registration and Interest forwarding strategies based on overhearing, i.e. 1DFo and 2DFo, and compared it to a static strategy, i.e., 2SF. Evaluations have shown that 1DFo outper-

forms both 2DFo and 2SF in terms of both content retrieval times and transmitted (Interest and Data) messages. The 1DFo strategy results in 27% faster content retrieval times than 2DFo and even 38% faster content retrieval times than 2SF. Interest aggregation and caching works efficiently even in case of multiple concurrent request streams. However, Interest aggregation is more efficient with 1DFo than with 2DFo or 2SF because Interests are forwarded over the same faces. As a consequence, the number of transmitted Data messages stays approximately constant for an increasing number of concurrent requesters.

The main challenge of the described approach is the fact that content needs to be overheard to configure forwarding prefixes. While this works well for popular content, it may be more difficult for unpopular content that is requested infrequently. Furthermore, the described approach is efficient in rather static scenarios or mobile scenarios with low host churn, e.g., cars on a highway, but it may degrade in highly mobile scenarios, if preferred forwarders may regularly move away.

Chapter 9

Dynamic Unicast for Wireless and Mobile Multi-hop Networks

9.1 Introduction

Since ICN messages do not contain any source or destination identifiers, authentic content can be cached and retrieved from any node. If the connectivity to a content source breaks, e.g., because the distance becomes too long, it is not required to establish a new path to the same content source or perform handovers to connect to a new content source. Instead, content requests can implicitly find new content sources nearby because requested content is returned if a source is available.

In Chapter 8 we have described an approach to enable information-centric multi-hop communication via broadcast. By overhearing broadcast communication, preferred forwarders can be identified, which are responsible to forward messages between requesters and content sources. While this is efficient in rather static networks, e.g., community networks, communication performance may degrade in mobile ad-hoc networks if preferred forwarders change frequently resulting in long forwarding delays.

In Chapter 6, we have introduced *Dynamic Unicast* for opportunistic one-hop communication and found that it is superior to one-hop broadcast because no forwarding delays are required. In this chapter, we consider Dynamic Unicast for wireless multi-hop communication. For this, requesters can broadcast Interests to find content sources and (if Data is received in return) configure dynamic FIB entries to next hops towards content sources. Then, the next Interests can be transmitted directly to the same content sources until they become unavailable and new sources need to be found via broadcast. In contrast to existing wireless ICN routing protocols, Dynamic Unicast does not require modifications to ICN messages such as end-point identifiers [144, 41] or location information [214, 100, 101], but stores forwarding information dynamically in local FIB entries.

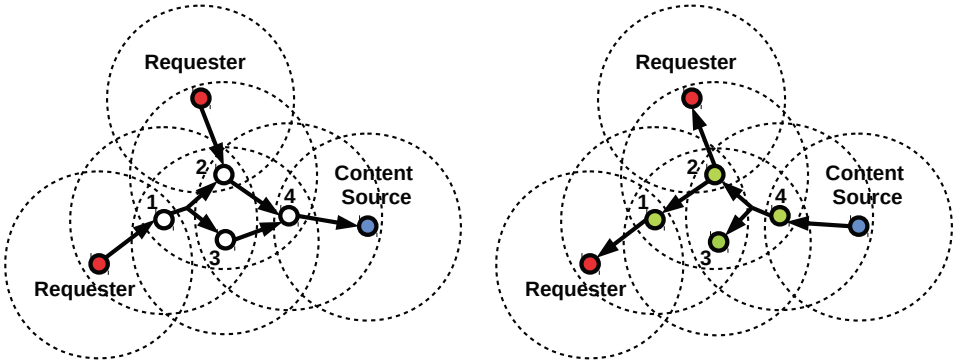
The work in this chapter [54, 221] differs in multiple ways from Dynamic Unicast in Chapter 6. First, we focus on multi-hop but not one-hop communication. This requires new FIB forwarding strategies, FIB update mechanisms and a pass-

through extension for multi-hop. Second, we implement and evaluate a Content Request Tracker (CRT) to perform unicast or broadcast transmissions depending on the number of concurrent requesters. Third, we integrate all described mechanisms in CCNx and do not use a network simulator as in Chapter 6. This increases the accuracy and credibility of our results because the same source code can be deployed on real wireless devices. Forth, we evaluate the described mechanisms in diverse scenarios incorporating different topologies, node velocities and content source densities (in Chapter 6 the evaluation has been performed with only one content source). In particular, we investigate whether dynamically created paths are resilient to mobility and whether Dynamic Unicast scales with an increasing number of requesters compared to broadcast communication.

9.2 Multi-hop Dynamic Unicast

In this section, we describe *Dynamic Unicast*, which creates dynamic unicast faces to locally available content sources that are discovered via broadcast requests.

9.2.1 Protocol Overview



(a) Requesters flood Interest Messages.

(b) Content Source replies Data via Broadcast. Nodes register Unicast Face after receiving Data Message.

Figure 9.1: Implicit Content Discovery via Broadcast.

Figure 9.1 illustrates implicit content discovery with Dynamic Unicast. Requesters broadcast Interest messages, which are forwarded by other nodes in transmission range. If multiple nodes receive an Interest via broadcast, e.g., nodes 2 and 3 in Figure 9.1a, they may forward it simultaneously resulting in a duplicate Interest transmission. If either node 2 or 3 forwards the Interest slightly before the other node, a duplicate Interest transmission may be prevented because the same Interest is already included in the PIT of the other node. Similar Interests from

multiple requesters can be aggregated in the PIT such that only one Interest is forwarded, e.g., at node 2 and 4 in Figure 9.1a. If Interests reach a content source, Data messages return on the reverse path via broadcast as illustrated in Figure 9.1b. All nodes that overhear the Data transmission can configure a unicast face to the previous hop in the FIB. Duplicate Data transmissions, e.g., at node 3 in Figure 9.1b, may be prevented since broadcast Data transmissions are delayed to enable duplicate suppression (see Subsection 2.1.1).

After the Data packet has reached the requesters, a hop-by-hop unicast path to the content source is configured in the FIBs of all intermediate nodes. Thus, the content retrieval can be performed via unicast as illustrated in Figure 9.2. Although transmissions are performed via unicast, Interest aggregation from different requesters and Data caching in intermediate nodes is still supported.

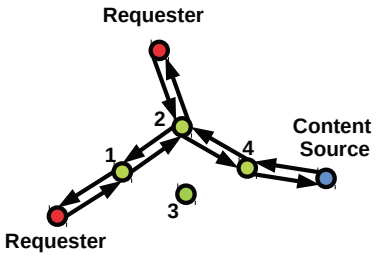


Figure 9.2: Content Retrieval via Configured Unicast Faces. Interest Aggregation and Caching is still possible.

9.2.2 Enabling Multi-hop Communication

To discover available content sources and enable Dynamic Unicast, broadcast Interests need to be transmitted (see Subsection 9.2.1). However, if no broadcast FIB entries are configured for the name prefix, the corresponding Interests are dropped. To address this issue, we have implemented a *pass-through* mechanism, which redirects Interests without matching FIB entry to a broadcast face. In Chapter 6, we enabled pass-throughs only for local applications but in this work we extend it for Interests from other hosts to enable multi-hop communication. To limit Interest forwardings by nodes, which can not reach a content source, users can define an upper limit of Interests that can be forwarded via pass-through. For example, a node may only allow one pass-through per content name, i.e., more Interests can only be forwarded if Data returns and a prefix is configured in the FIB or if the Interest expires and is removed from the PIT.

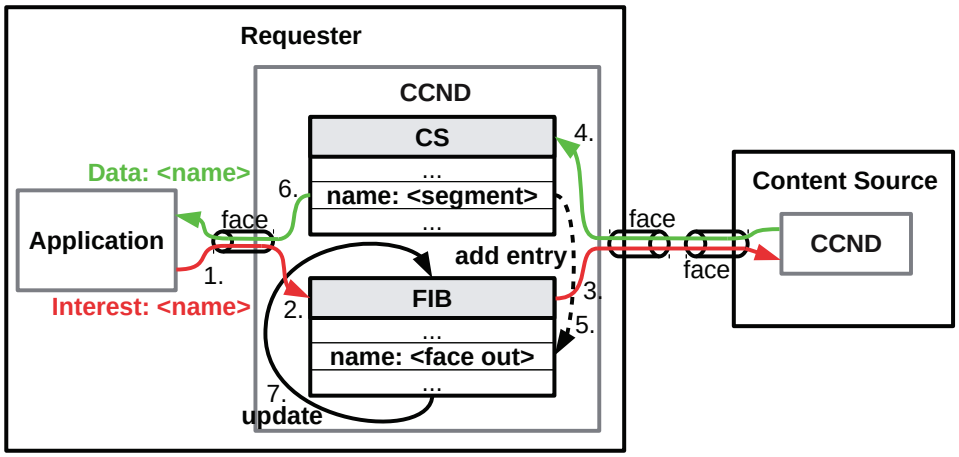

Requester
CCND
CS
...
name:
...
4.
Data: <name>
6.
face
Application
1.
Interest: <name>
2.
add entry
FIB
...
name: <face out>
...
3.
5.
7.
update
face
face
Content Source
CCND

Figure 9.3: Overhearing of Data Messages to configure Dynamic Unicast Faces in the FIB.

9.2.3 Dynamic Prefix Registration

Figure 9.3 illustrates the prefix registration process at a requester. Applications initiate content retrieval by transmitting Interests via internal face to the CCND (step 1), where a FIB lookup is performed (step 2). If the FIB does not contain a unicast face to a content source, the Interest is transmitted via broadcast (step 3) towards any content source in the vicinity (pass-through). Forwarded Interests are always included in the PIT (not shown in Figure 9.3). After receiving a Data message on the reverse path (step 4), dynamic FIB entries can be configured towards the content source (step 5) and the Data is forwarded further (step 6) based on PIT information. To maintain accurate forwarding information, the FIB is regularly updated (step 7, see Subsection 9.2.4). Dynamic FIB entries based on received Data messages are only configured if the corresponding Interests have been transmitted and the Data is new, i.e., not already in the cache indicating that the Data has been received previously. To configure a dynamic FIB entry, the node ID of the previous hop and the content prefix (content name without segment number) are extracted. In our implementation, we use IP addresses as node IDs, however, other node IDs such as MAC addresses or descriptive names would also be possible. These node IDs are not included in NDN messages but can be extracted from headers of packets that have transported NDN messages over the previous hop, e.g., IP or MAC packets. Hence, we do not use IP addresses for global end-to-end routing but only to identify a next hop towards a content source. As mentioned in Chapter 6, node IDs can be viewed as temporal content locator similar to locator-based mobility approaches [107, 168, 108, 125]. In multi-hop communication a “next-hop”-node might not necessarily be the content source itself but could only be a forwarding node. However, from a requester’s perspective, a differentiation is not required because a forwarding node acts as content source for a requester.

9.2.4 Updating the FIB

In mobile networks, connectivity to other nodes may change quickly. Thus, it is crucial to remove outdated information from the FIB as quickly as possible because Interests transmitted over broken paths increase message overhead and transmission times. We delete expired information in two cases.

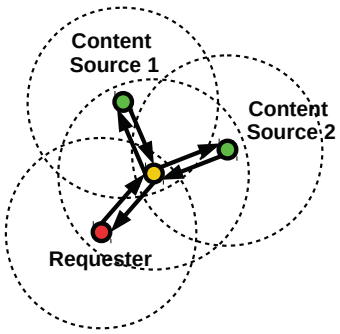


Figure 9.4: Retrieving Content from two Content Sources via Neighbor Node.

First, we perform periodic update operations in the FIB to track the number of received messages over a face. If no messages have been received via a dynamic unicast face for some time, e.g., if the neighbor is not in range anymore, the corresponding face is automatically deleted. If faces are deleted, the corresponding FIB entries need to be updated and entries without valid faces are removed. This mechanism is already available in CCNx since dynamic unicast faces are already used in CCNx, i.e., they are created to return Data via unicast when Interests have been received via unicast. However, we reduce the period to check whether a face is still used from 16s to 4s to detect path breaks quicker (since path breaks may occur frequently in mobile networks). Consequently, also valid paths are removed quicker if they are not used anymore, limiting the number of active forwarding entries in the FIB. In the worst case, i.e., too early deletion of a FIB entry, another Interest needs to be broadcast to establish a new path to a content source. However, this is not necessarily a disadvantage because it enables requesters to find new (and potentially closer) content sources.

Second, a requester may retrieve different content objects from the same neighbor node via different paths, e.g., from two content sources via a neighbor node as shown in Figure 9.4. Then, it is possible that a source, e.g., content source 2, may move away but the neighbor node is still in range to receive and forward messages from and to content source 1. Thus, the requester would still keep the face to the neighbor node (because it receives messages) although content from source 2 can not be retrieved anymore. This illustrates that automatic deletion of dynamically created unicast faces (case 1 above, and one-hop communication in Chapter 6) is not enough to support route maintenance for multi-hop communication. Consequently, we add a short prefix validity time of 5 seconds to each dynamically created FIB entry, i.e., slightly more than the default Interest lifetime of 4 seconds

such that retransmissions (in case of collisions) can be satisfied from nearby caches but broken paths still expire quickly. Whenever a Data message is received over the configured face, the lifetime of the configured FIB entry is extended. Otherwise, the prefix is deleted from the FIB after 5 seconds.

9.2.5 Interest Forwarding Strategies

Since multiple faces can be configured for the same prefix in the FIB, forwarding strategies are required to define over which faces Interests are forwarded. We evaluate two forwarding strategies as described below.

Single Face Forwarding (SFF)

This strategy establishes a single path from a requester to a content source. Every Interest is first forwarded over the "best" face and if nothing has been received in return, it is forwarded via broadcast (fallback). If a unicast face is available, it is considered the "best" face (priority over broadcast). If multiple unicast faces are available, the face that has been (successfully) used the last time is the best face. This is slightly different from Chapter 6, where new unicast faces have priority over existing unicast faces. We decided to prefer successfully used unicast faces to avoid frequent path fluctuations, resulting in more stable multi-hop paths.

Parallel Face Forwarding (PFF)

This strategy can establish multiple paths between requester and content source. If there are multiple unicast faces, the PFF strategy sends the Interests over all unicast faces in parallel and not only over the "best" face as the SFF strategy. Interests are first forwarded via all unicast faces and if nothing has been received (on either face), they are forwarded via broadcast (fallback).

9.3 Content Request Tracker (CRT)

If there are many concurrent requesters for the same content, e.g., for a video broadcast, a single broadcast transmission may be more efficient than multiple unicast transmissions. Therefore, we introduce a *Content Request Tracker (CRT)* as optional extension of the CCND message forwarding engine. A CRT is a hash table, which maintains one CRT token (struct containing multiple connection parameters) for each actively requested content prefix. A CRT can be applied only at source nodes (CRT-S) or both source and requester nodes (CRT-SR) as described in the following subsections.

9.3.1 CRT at Source (CRT-S)

Figure 9.5 illustrates message processing at a content source. If a content source receives a unicast request, it checks the FIB if the content is locally available in the repository (step 1). If the content is available, the corresponding CRT token is loaded or a new CRT token is created if it does not yet exist (step 2). At the content source, the CRT token maintains information about active unicast connections, i.e., number and node IDs of current requesters, and the time when the last request has been received. The Interest can then be forwarded to the repository (step 3). Before Data can be returned to an individual requester based on PIT information (not shown in Figure 9.5), the CRT is consulted (step 4). If a certain number of *MAX_CRT* different unicast requests has been received for the content prefix, the content source transmits the Data message via broadcast (step 5). In this case, all pending Interests in the PIT, i.e., with unicast faces that would be satisfied by the broadcast transmission, are removed (not shown in Figure 9.5).

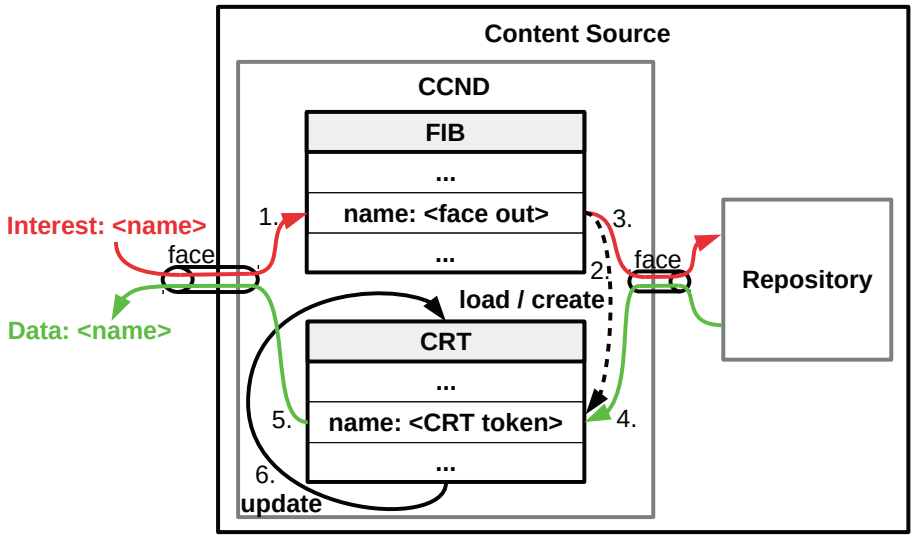


Figure 9.5: CRT at Source to support CRT-S and CRT-SR.

The CRT is regularly updated (step 6) by deleting CRT entries for content prefixes if no new requests have been received for a time T (in our implementation: 1 minute). Furthermore, CRT entries are deleted whenever the content source transmits the final segment of a content object.

9.3.2 CRT at Source and Requester (CRT-SR)

At the source, CRT-SR uses the same message processing as CRT-S. Figure 9.6 illustrates message processing at a requester. After a Data message has been received via broadcast, the requester checks in the PIT whether a unicast request has been transmitted for it (not shown in Figure 9.6). Then, the requester checks the

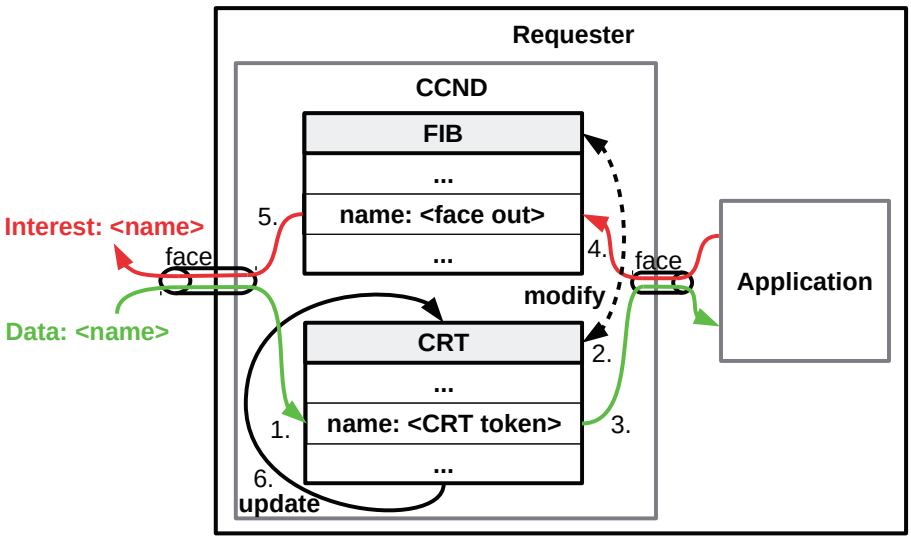


Figure 9.6: CRT at Requester to support CRT-SR.

CRT for the content prefix (step 1). A requester keeps track of subsequent broadcast responses to unicast requests in a CRT token, i.e., last received segment number and number of subsequent Data messages. Multiple broadcast Data responses would indicate that a content source may have switched to broadcast (CRT-S) due to many concurrent unicast requests. Thus, for a certain number of *MAX_CRT_REQ* subsequent broadcast responses to unicast requests at requesters (in our implementation: MAX_CRT_REQ = 2), dynamically created unicast faces are removed from the FIB (step 2). Then, the Data is forwarded to the application (step 3) triggering the next Interest (step 4), which is transmitted via broadcast (step 5). To avoid fluctuations between unicast and broadcast, new dynamic unicast faces are only created at a requester if no CRT token is available for the content name, i.e., the requester has not switched back to broadcast deliberately. The update process of CRT-SR (step 6) is identical to CRT-S.

9.4 Evaluation

We have implemented Dynamic Unicast (DU) with Single Face Forwarding (SFF) and Parallel Face Forwarding (PFF) as well as CRT-S and CRT-SR in CCNx 0.8.2 [27]. However, all described communication mechanisms can also be supported by CCNx 1.0. In dynamic networks, unicast faces can not be statically configured due to changing connectivity. Therefore, we compare DU to broadcast communication (using unmodified CCNx as reference) in our mobile scenarios. All evaluations have been performed with NS3-DCE [16] on a Linux cluster [200]. By that, we evaluate the same source code on simulated nodes that would run on real wireless devices.

9.4.1 Evaluation Parameters

The wireless configuration is listed in Table 9.1. Every node has an IEEE 802.11g wireless interface and we use a log distance propagation loss model. With the selected parameters, the transmission range is approximately 130m (outdoor scenario). The unicast data rate is adapted automatically based on the distance between two nodes. Since the broadcast data rate can not be adapted, it is usually set to the lowest supported rate. In every evaluation, requesters retrieve a 5 MB file (1280 segments à 4096 bytes) from one or multiple content sources. The freshnessSeconds, i.e., how long each segment is valid in the cache, is set to 300 seconds.

Parameter	**Value**
Wireless Standard	IEEE 802.11g, 2.4 GHz
Modulation	ERP-OFDM, min. data rate: 6 Mbps max. data rate: 54 Mbps
Propagation Loss Model	Log Distance with Exponent: 3.0 Reference loss: 40.0 dB
Error Model	NIST Error Rate Model
TX power	16.0206 dBm (default in NS3)
RX/TX gain	1 dB (default in NS3)
Energy Detection Threshold	-86.0 dBm
CCA Model Threshold	-90.0 dBm

Table 9.1: Wireless Configuration for Evaluations with Multi-hop Dynamic Unicast.

We evaluate wireless (multi-hop) communication in different mobile and static scenarios with multiple requesters, forwarding nodes and content sources. The scenarios and selected topology parameters are explained in the following subsections. Since end-to-end paths during multi-hop communication can be disrupted, we use a content retrieval application (see Chapter 5), which persistently stores received segments at a requester. Then, even in case of long disruptions (when cached content may be deleted), content downloads can always be resumed from where they were stopped. DU establishes a path between requester and content source such that only nodes on the path receive and forward messages (a new path is established if the old path breaks). In contrast, with broadcast all nodes receive messages and decide individually whether they forward them or not. To compare the two schemes we evaluate the Interest and Data overhead as defined in Section 3.3. We have evaluated the message overhead separately for content sources, requesters and forwarder nodes (neither requesters nor content sources). Every configuration in each scenario has been evaluated in 100 different runs and the boxplots show the average message overhead of all evaluation runs.

9.4.2 Multi-hop Communication

To evaluate the performance of DU and broadcast for a varying path length, we compare them to unicast communication in a line topology as illustrated in Figure 9.7. Neighboring nodes have a distance of 75m such that only direct neighbors can communicate with each other. Since both forwarding strategies (SFF and PFF) behave the same (only one neighbor), we evaluate the SFF strategy only. Table 9.2 lists the scenario parameters, i.e., we evaluate multi-hop communication in a static scenario for up to 15 hops. The FIB entries for unicast routing have been manually configured, while DU establishes unicast faces dynamically.

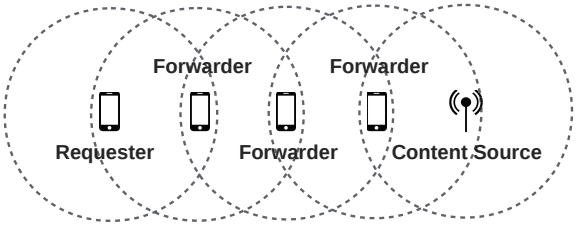


Figure 9.7: Line Topology with 4 hops. Only Direct Neighbors can communicate.

Parameter	Value
Nodes	1 static requester 1 static source 0-14 static forwarders
Mobility	no mobility, static

Table 9.2: Evaluation Parameters for Line Topology.

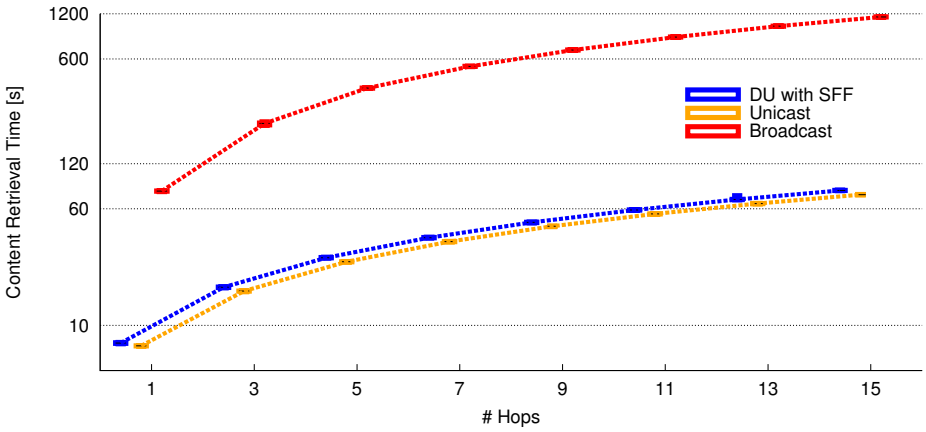


Figure 9.8: Multi-hop Communication via Broadcast, Unicast and Dynamic Unicast using the SFF Strategy.

Figure 9.8 shows content retrieval times of a requester to retrieve a 5 MB file over multiple hops. Content retrieval via broadcast takes considerably more time than via unicast due to two reasons. First, broadcast Data transmissions are delayed in CCNx to enable duplicate suppression. Although we used the (default) minimum broadcast delay (CCNx data pause) of 10ms, these broadcast delays have significant impact on throughput because they are added at every hop. Second, broadcast data rates are in general lower than for unicast because they are set to the lowest supported rate while unicast rates can be adapted based on the distance (see Table 9.1 for our wireless configuration).

Consequently, content retrieval times for an increasing number of hops increase faster for broadcast than for unicast communication. For example, content retrieval via broadcast over 15 hops requires on average 14.5 times more time than over 3 hops, while content retrieval via unicast (no delay) requires only 10.4 times more time. We have also compared DU with unicast for reference purposes. The time overhead is low, i.e., it varies between 5.7% (3 hops) and 6.3% (15 hops) more time for DU compared to unicast. However, unicast routing is only possible in static scenarios because faces need to be manually configured in the FIB. For this reason, we compare DU only with broadcast (and not unicast) in the remainder of this chapter.

9.4.3 Multiple Sources and Requesters

We evaluate one-hop and multi-hop communication in networks with multiple requesters, content sources and mobile forwarder nodes.

Evaluation Scenario

Natural disasters, e.g., floodings, earthquakes, or wars may destroy communication infrastructures such that links to central servers are broken. In such scenarios, ICN may enable users to retrieve local emergency information from deployed wireless mesh nodes acting as content sources (see Chapter 4).

Figure 9.9 illustrates the evaluation scenario. We use a square playground with a side length $l_{playground}$ of 1000m and assume that content sources are deployed in a grid. Depending on the number of content sources $n_{sources}$, the playground is divided into smaller regions and each region has a content source in the middle. The side length l_{region} of these smaller regions is calculated by

$$l_{region} = \frac{l_{playground}}{\sqrt{n_{sources}}} \tag{9.1}$$

For example, Figure 9.9 shows 4 content sources that are placed in the middle of regions with side lengths $l_{region} = 500m$.

Table 9.3 lists the scenario parameters. Within the playground, there are 1-30 mobile requesters who want to retrieve the same 5 MB content concurrently as well as 30 mobile nodes, which do not perform active requests but can forward received

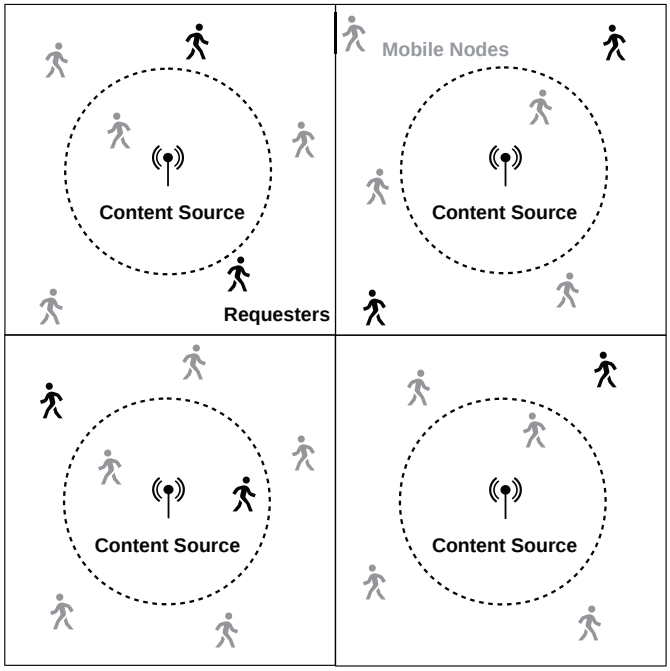


Figure 9.9: Grid Topology: Content Sources are placed in a Grid, here shown for 4 Content Sources.

Parameter	Value
Nodes	1-36 static sources 1-30 mobile requesters 30 mobile nodes (forwarders)
Playground	Side Length: 1000m
Mobility	Random Waypoint Mobility node speed: 1.2m/s node pause: 0s

Table 9.3: Evaluation Parameters for Grid Topology.

Interests from requesters. We evaluate the performance of concurrent requests in topologies with 1-36 static content sources (although 16 content sources are already sufficient to ensure one-hop distance to a content source on the entire playground).

One-hop vs. Multi-hop

In Chapter 6, we have considered DU only for one-hop communication. Thus, we first explore the performance of multiple requesters using one-hop and multi-hop DU with SFF and PFF against one-hop and multi-hop broadcast.

Figure 9.10 illustrates the cumulative content retrieval times for 30 concurrent requesters and 1 content source. The cumulative content retrieval time denotes

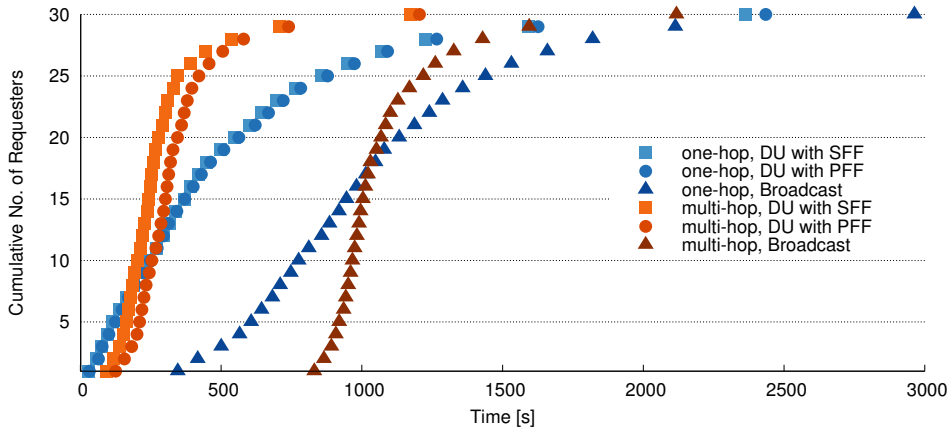


Figure 9.10: Cumulative Content Retrieval Times for One-hop and Multi-hop Communication in case of 30 Requesters and 1 Content Source.

the time (x-axis) at which a certain number of requesters (y-axis) has received the content. Figure 9.10 shows that DU with SFF and PFF perform similar for one-hop communication but SFF performs slightly better than PFF for multi-hop communication. Furthermore, there is a clear benefit (shorter retrieval times) for multi-hop communication because requesters may not always be in direct (one-hop) transmission range of a content source. However, the first few requesters can retrieve content faster with one-hop communication. These requesters are in a one-hop transmission range from the content source and can retrieve content directly. With multi-hop communication, more Interest messages are transmitted in the network, which means that there is less time for Data transmissions on the shared wireless medium. In addition, requesters may retrieve content indirectly via forwarders although one-hop communication would be possible.

We have also performed evaluations where requesters set the AnswerOriginKind field in Interest messages [5] to enforce content retrieval from content sources. However, it resulted in worse performance because i) Interest retransmissions could not be satisfied from caches and had to be forwarded to the content source and ii) requesters could not benefit from nearby nodes that have retrieved the content already. A better solution to limit Interest propagation and improve download performance over multiple hops may be the introduction of hop counters in Interest messages as included in CCNx 1.0 [149]. We discuss hop counters in Section 9.5.2.

Multi-hop Communication

Figure 9.11 shows the cumulative content retrieval times of 30 requesters that retrieve a 5 MB file via multi-hop communication. We evaluate different content source densities (1 to 36 content sources) but do not show the results for 36 content sources because they overlap with the results for 16 content sources. As expected, the content retrieval times decrease with increasing number of content sources.

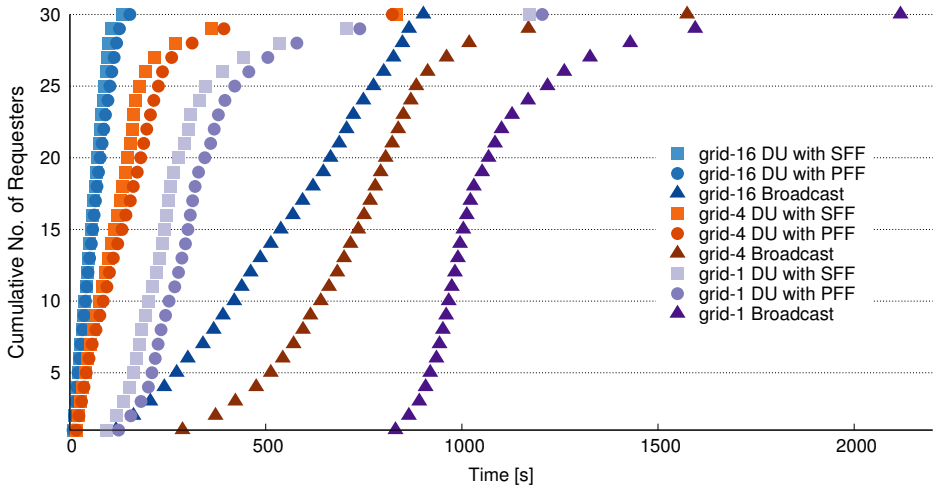


Figure 9.11: Cumulative Content Retrieval Times for Different Numbers of Content Sources (Grid Topology) and 30 Requesters.

DU is better than broadcast for all source configurations, even when content density is low, e.g., for 1 content source. Although broadcast Data transmissions can be overheard by multiple nodes, which cache the content, content retrieval times via broadcast are longer than with DU due to two main reasons. First, DU can exploit short contact times to content sources better due to (potentially) higher data rates and no forwarding delays. Second, overheard and cached content is beneficial in case of multiple requesters (popular content). However, for multiple requesters, the content density is also high with DU. Although requesters cannot overhear and cache unicast Data transmissions from other requesters, they can still cache content, which they requested themselves, resulting in shorter path lengths for other requesters. For all source configurations, multi-path forwarding (DU with PFF) results in worse performance than single-path forwarding (DU with SFF). To better understand the reasons for this, we evaluate the message overhead.

Figure 9.12 shows the Interest overhead of mobile nodes with DU and broadcast. The Interest overhead for DU is considerably lower than for broadcast because messages are mostly transmitted over established paths. The more content sources are in the playground, the shorter are the forwarding paths and the fewer Interests are forwarded with DU. The PFF strategy results in slightly more Interest transmissions than SFF. For broadcast, the number of Interest transmissions does not decrease significantly for an increasing number of content sources because Interests cannot be addressed to a specific source but are flooded in the network.

The more Interest messages are forwarded by mobile nodes, the more Data messages are transmitted as Figure 9.13 shows. Since not every Interest retrieves Data, the number of transmitted Data messages is slightly lower than the number of Interest messages. Yet, many Data transmissions by mobile nodes are not necessarily bad if it results in less Data transmissions by content sources.

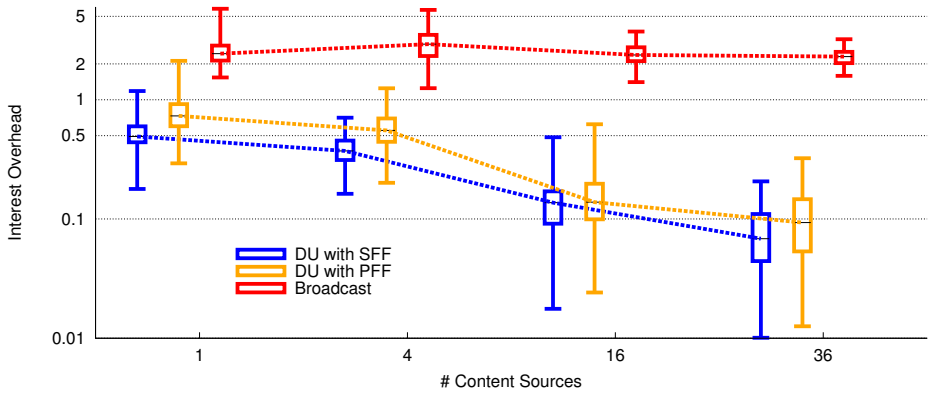


Figure 9.12: Interest Overhead by Mobile Nodes (Forwarders) for 30 Concurrent Requesters.

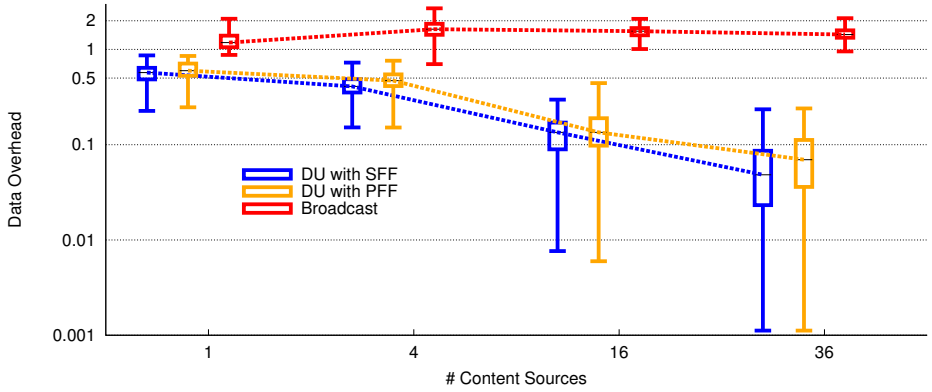


Figure 9.13: Data Overhead by Mobile Nodes (Forwarders) for 30 Concurrent Requesters.

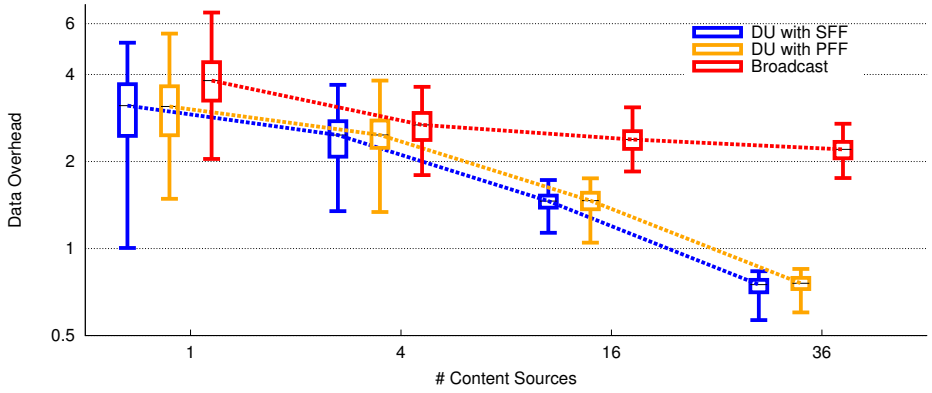


Figure 9.14: Data Overhead by Content Sources in case of 30 Concurrent Requesters.

Figure 9.14 shows the Data overhead by content sources for 30 concurrent requesters. Content sources have a significantly lower Data overhead value than 30, which would be the overhead if downloads from 30 requesters would be independent of each other as in host-based communication. Surprisingly, content sources send fewer Data messages with DU than with broadcast despite multiple concurrent requests. While the difference between broadcast and DU is rather low for 1 content source (18% fewer Data messages with DU), the differences increase for higher content densities, e.g., 66% fewer Data messages with DU for 36 content sources. Although DU establishes unicast paths, not every Data message needs to be retrieved from a content source but can also be obtained from caches of other requesters or mobile forwarders.

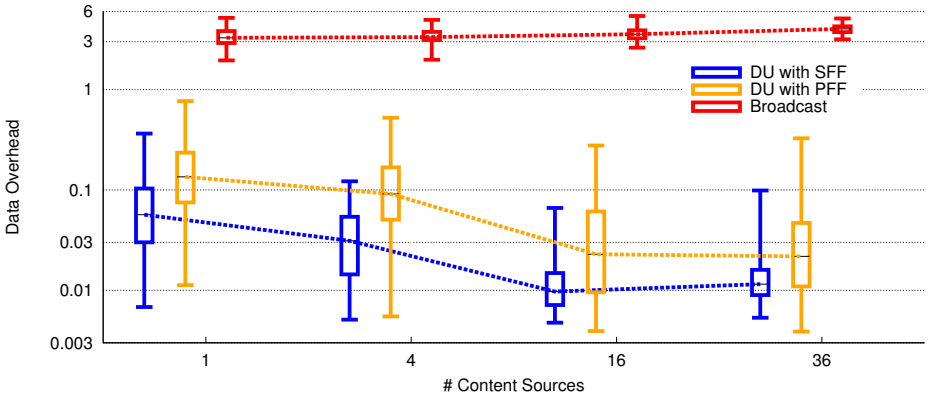


Figure 9.15: Duplicate Data Overhead at Requesters in case of 30 Concurrent Requesters.

Figure 9.15 shows the duplicate Data overhead at requesters. While the number of duplicate Data messages increases with broadcast by 23% from 1 to 36 content sources, received duplicate Data messages decrease with DU by 80%. For broadcast, a higher content density yields more content sources that are addressed by broadcast requests at once (more duplicates). For DU, however, a higher content density yields shorter path lengths resulting in fewer Data transmissions by forwarders. Only if the number of content sources is larger than required, e.g., for 36 instead of 16 sources, the number of received duplicate Data messages increases slightly with DU due to broadcast transmissions, which are required to establish unicast paths.

9.4.4 Mobility during Multi-hop Communication

We investigate the impact of mobility on route persistence during wireless NDN multi-hop communication. Figure 9.16 illustrates the investigated scenario. To enforce multi-hop communication, a static requester and a content source are placed at opposite corners of a square playground (10m to the borders) in 500m distance to each other. Table 9.4 lists the evaluation parameters. There are 50 mobile for-

warders, which move according to the Random Waypoint mobility model and make occasional breaks, i.e., no mobility. The node pause denotes the maximum break time, e.g., a node pause of 3600s means that a node randomly waits between 0s and 3600s. As reference, we also evaluate a static scenario where static forwarders are randomly distributed in the playground.

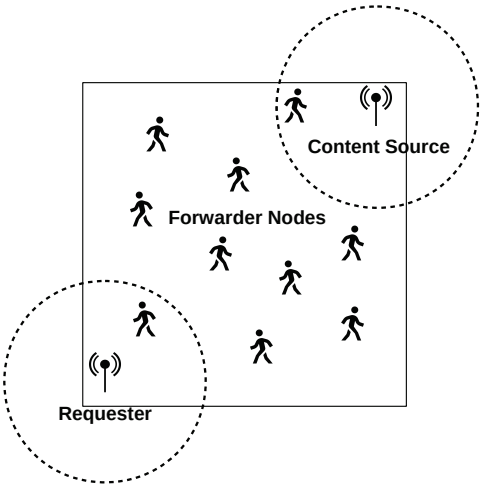


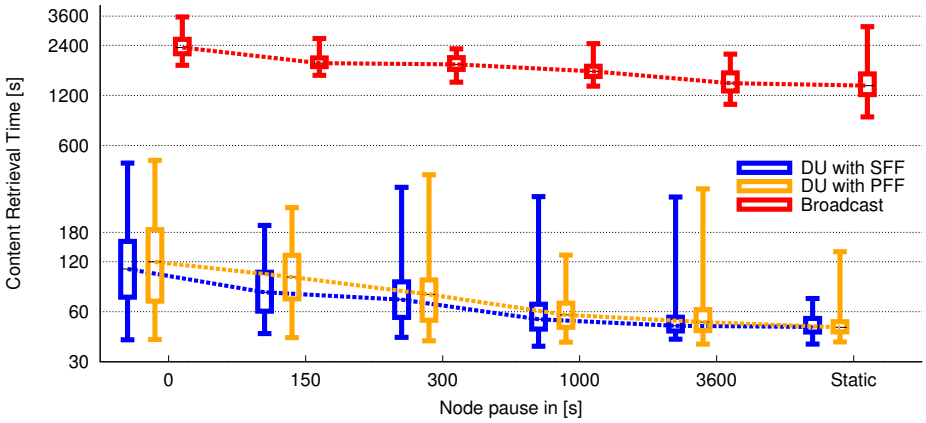
Figure 9.16: Multi-hop Topology with One Static Requester, One Static Content Source and Multiple (Mobile) Forwarder Nodes.

Parameter	Value
Nodes	1 static source 1 static requester 50 forwarder nodes
Playground	side length: 374m distance: 500m (source - requester)
Mobility	no mobility, static
	Random Waypoint Mobility node speed: 1.2m/s, 14m/s node pause: 0-3600s

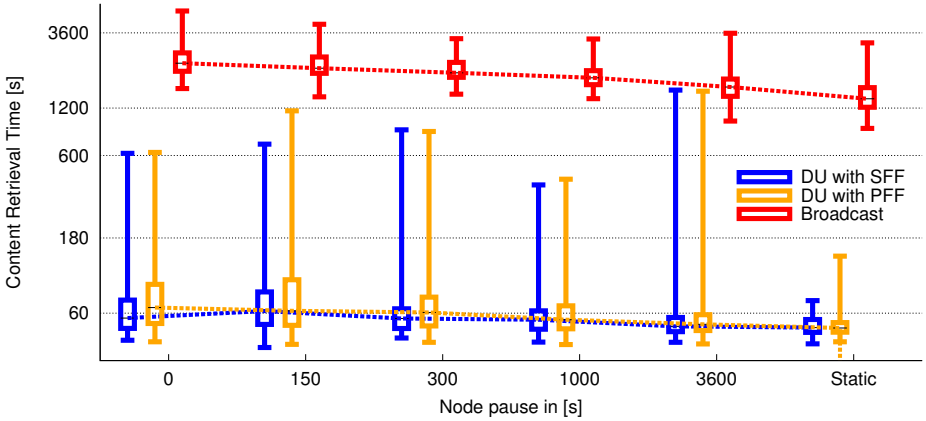
Table 9.4: Evaluation Parameters for Mobility Scenarios.

Figure 9.17a shows content retrieval times of a requester retrieving a 5 MB file from the content source when mobile nodes move with 14m/s. The x-axis denotes the node pause times and the rightmost graphs show the static case (no mobility). For DU with SFF, content retrieval times decrease from a high mobility scenario (node pause: 0s) to a static scenario by 55% and for DU with PFF they decrease by 60%. However, content retrieval times decrease even with broadcast by 41% from the high mobility (node pause: 0s) to the static scenario.

Hence, we have repeated the same evaluation with a slower node velocity of



(a) Mobile Nodes move with a Velocity of 14m/s.

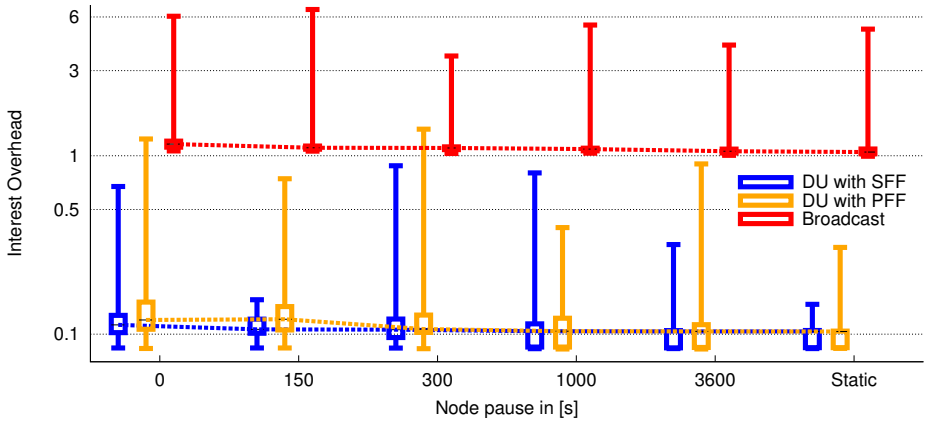


(b) Mobile Nodes move with a Velocity of 1.2m/s.

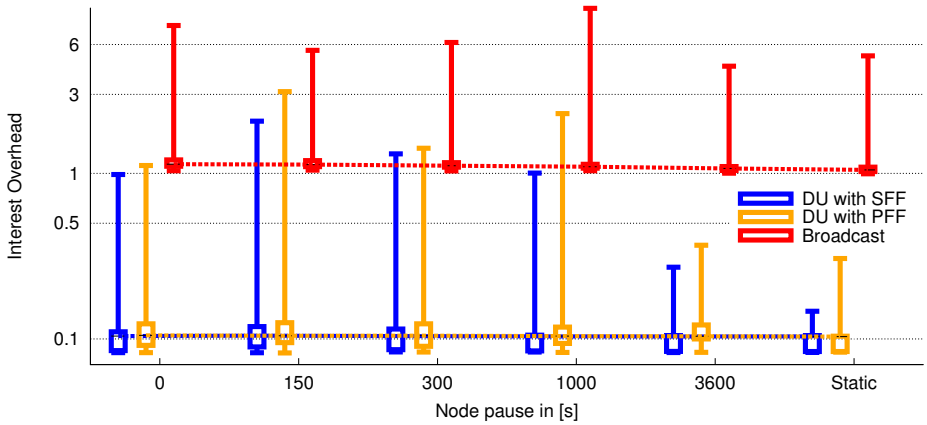
Figure 9.17: Content Retrieval Times for a Static Requester retrieving Content from a Content Source at a Distance of 500m. There are 50 Mobile Nodes (Forwarders).

1.2m/s and show the content retrieval times in Figure 9.17b. The content retrieval times decrease only by 13% (DU with SFF) and 26% (DU with PFF) from the high mobility to the static scenario. However, the maximum content retrieval times (worst cases) are longer compared to 14m/s because bad node formations, where multi-hop communication is temporarily not possible or only over many hops, stay for a longer time.

Figure 9.18a illustrates the Interest overhead by mobile forwarders with a velocity of 14m/s. DU with SFF results in only 9% more Interest transmissions in high mobility compared to the static scenario, while DU with PFF results in 16% more Interests and broadcast results in 11% more Interests in case of high mobility compared to the static scenario. As expected, the increase of Interest messages from static to a high mobility scenario is lower with a velocity of 1.2m/s as Figure 9.18b



(a) Mobile Nodes move with a Velocity of 14m/s.

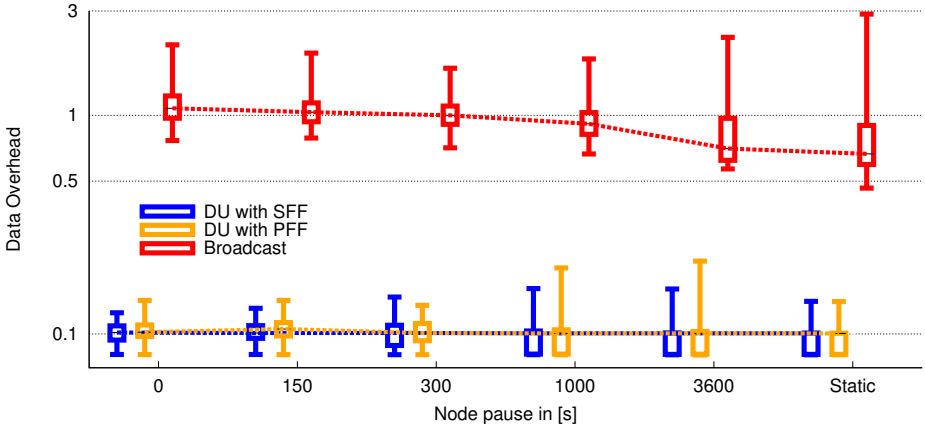


(b) Mobile Nodes move with a Velocity of 1.2m/s.

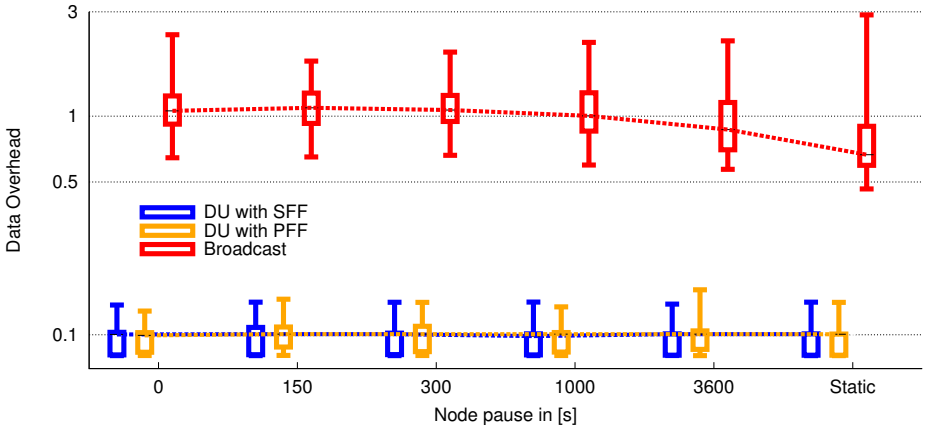
Figure 9.18: Interest Overhead during Multi-hop Communication by Mobile Nodes (Forwarders).

confirms: for DU with SFF it is only 0.4% more Interests and for DU with PFF it is only 1.6% more Interests in high mobility compared to static scenarios. This illustrates that the main reason for longer content retrieval times in high mobility scenarios are path breaks. In our evaluations, we used the default Interest lifetime of 4 seconds, which means that timeouts and retransmissions are only triggered after 4 seconds. However, if Interest lifetimes would be adapted based on estimated round-trip times (RTT), communication may recover faster from path breaks resulting in shorter content retrieval times in high mobility scenarios (see Chapter 10 for adaptive Interest lifetime algorithms based on estimated RTTs).

Figure 9.19a illustrates transmitted Data messages by mobile forwarders with a velocity of 14m/s. The evaluation results for a node velocity of 1.2m/s are nearly identical as Figure 9.19b shows. We can observe that the number of transmitted



(a) Mobile Nodes move with a Velocity of 14m/s.



(b) Mobile Nodes move with a Velocity of 1.2m/s.

Figure 9.19: Data Overhead during Multi-hop Communication by Mobile Nodes (Forwarders).

Data messages with DU is almost constant, i.e., for a node velocity of 14m/s there are 0.9% (SFF) and 1.3% (PFF) more Data transmissions in high mobility compared to static scenarios. This illustrates that breaking of symmetric Interest-Data forwarding paths is not an issue for DU. In contrast, for a node velocity of 14m/s, broadcast results in 62% more Data transmissions in high mobility compared to static scenarios. Since broadcast delays for broadcast transmission result in longer (and more varying) round trip times, broadcast forwarding paths are susceptible to mobility.

9.4.5 Multiple Concurrent Requests for Multi-hop Communication

We explore the scalability of wireless NDN communication by evaluating multiple concurrent requests over multiple hops. Figure 9.20 illustrates the evaluation topology. Similar to the last subsection, requesters and content source are static and there are multiple forwarding nodes. However, the number of static requesters varies between 1 and 48. The evaluation parameters are listed in Table 9.5. Besides static forwarders, we also evaluate mobile forwarders with a velocity of 1.2m/s.

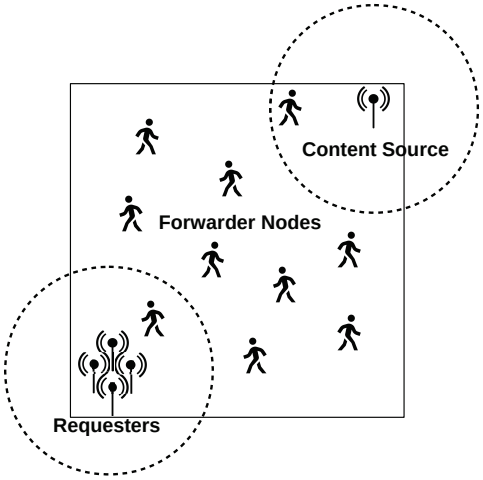
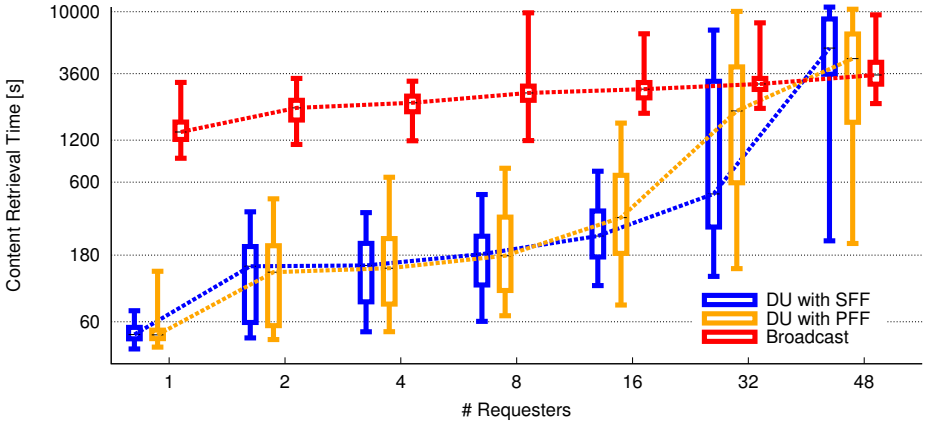


Figure 9.20: Multi-hop Topology with Multiple Requesters, Forwarder Nodes and One Content Source.

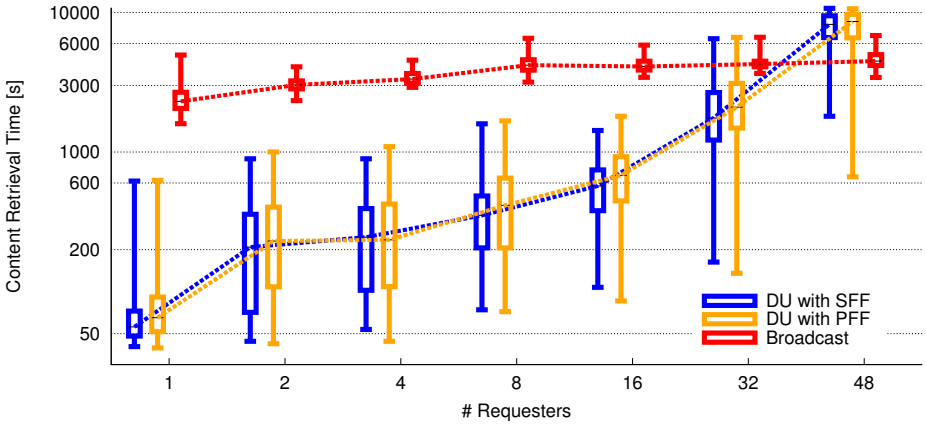
Parameter	Value
Nodes	1 static source 1-48 static requesters 50 forwarder nodes
Playground	side length: 374m distance: 500m (source - requesters)
Mobility	no mobility, static
	Random Waypoint Mobility node speed: 1.2m/s node pause: 0s

Table 9.5: Evaluation Parameters for Multiple Requesters.

Figure 9.21a shows content retrieval times of multiple requesters that retrieve the same 5 MB content object via static forwarders from the content source. From 1 to 32 requesters, the content retrieval times increase by a factor of 10.2 for DU with SFF, i.e., less than linear, and only by a factor of 2.2 for broadcast. However, even for 32 concurrent requesters, DU with SFF results in 88% shorter content retrieval



(a) Static Forwarding Nodes

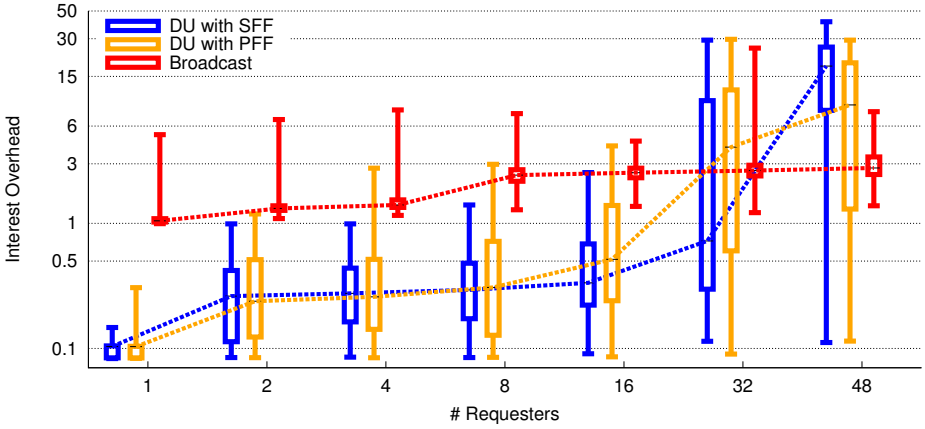


(b) Mobile Forwarding Nodes move with a Velocity of 1.2m/s.

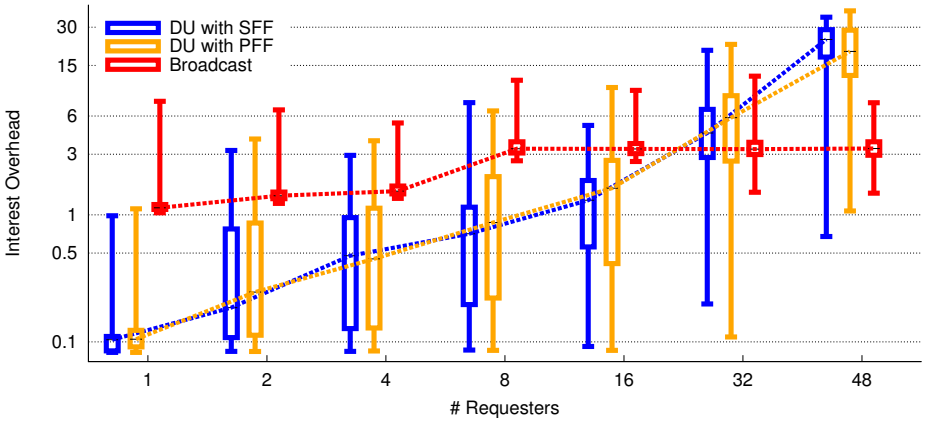
Figure 9.21: Content Retrieval Times for Multiple Requesters during Multi-hop Communication. The Requesters are at a Distance of 500m from the Content Source.

times than broadcast. For 48 concurrent requesters, DU with SFF performs worse than broadcast because the network is overloaded by the requests. Evaluations with mobile forwarders in Figure 9.21b look similar but content retrieval times for DU with SFF perform slightly worse compared to the static case. However, for 32 concurrent requesters, DU with SFF has still 59% shorter content retrieval times than broadcast.

Figure 9.22a illustrates the Interest overhead by static forwarders. From 1 to 32 requesters, the Interest overhead increases only by a factor of 7 because similar Interests can be aggregated in the PIT. However, for more than 32 concurrent requests (or the PFF strategy), the network gets overloaded such that Interests can not be efficiently aggregated anymore. For broadcast, the Interest overhead increases only by a factor of 2.5 from 1 to 32 requesters, but this is mainly because Interest



(a) Static Forwarding Nodes

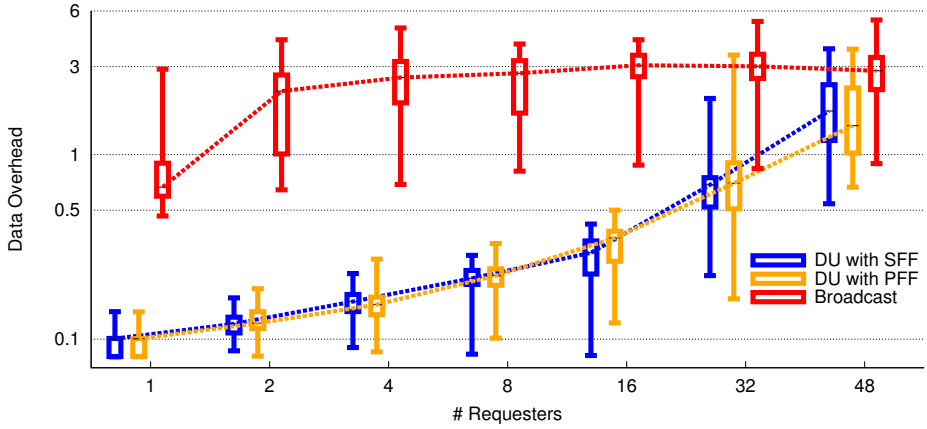


(b) Mobile Forwarding Nodes move with a Velocity of 1.2m/s.

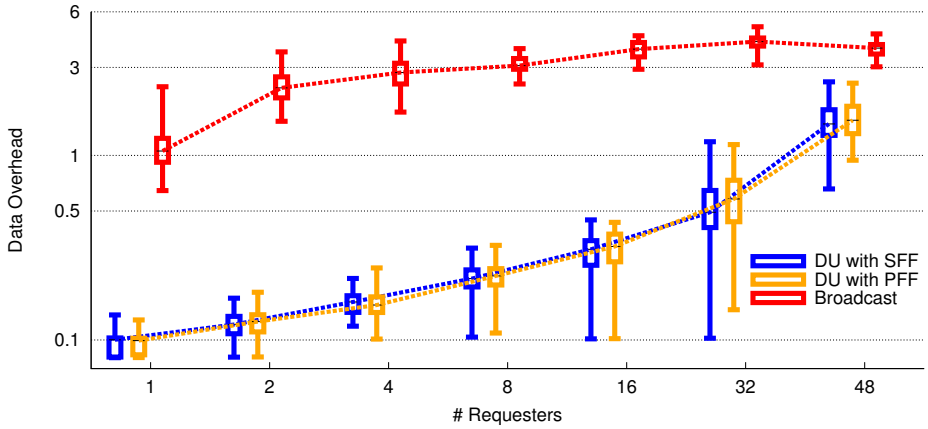
Figure 9.22: Interest Overhead of Forwarders During Multi-hop Communication for Multiple Requesters.

transmissions are already on a high level for only a few requesters. For mobile forwarders (Figure 9.22b), the Interest overhead of DU with SFF increases more, e.g., by a factor of 12.5 from 1 to 16 requesters, because Interests cannot be aggregated as efficiently as in the static case.

Figure 9.23a depicts the Data overhead by static forwarders. The data overhead increases only by a factor of 17 (DU with SFF) from 1 to 48 requesters due to shorter path lengths (caching). Even for 48 concurrent requesters, the Data overhead of DU with SFF is 40% lower than for broadcast. In the mobile scenario in Figure 9.23b, the Data overhead increases even only by a factor of 14.8 (DU with SFF) from 1 to 48 requesters because some redundant unicast paths (due to multiple requesters) may break more easily with mobility and retransmissions can be satisfied by nearby caches. Since Interest messages are significantly smaller than



(a) Static Forwarding Nodes

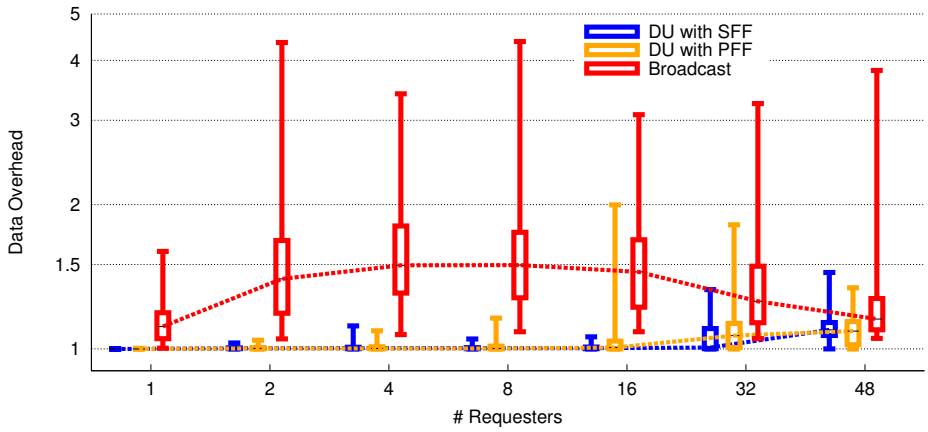


(b) Mobile Forwarding Nodes with a Velocity of 1.2m/s.

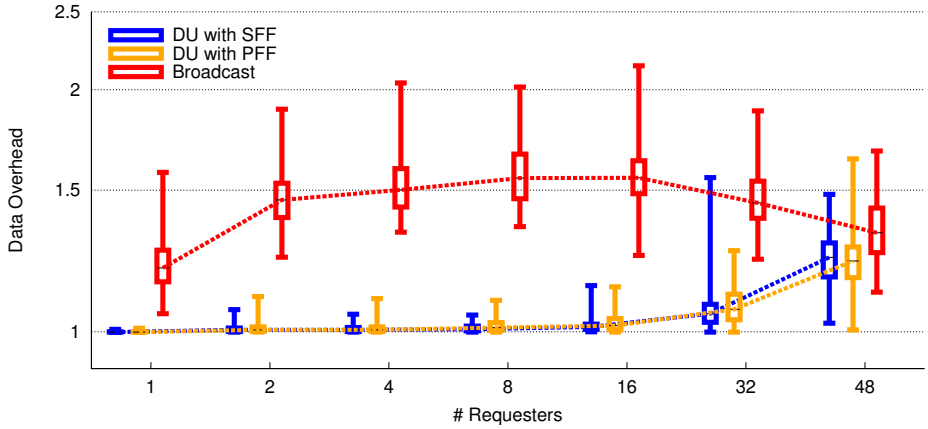
Figure 9.23: Data Overhead of Forwarders During Multi-hop Communication for Multiple Requesters.

Data messages, i.e., 50 bytes vs. 4500 bytes, a reduction of Data transmissions has a larger impact on the network traffic than fewer Interest transmissions. Thus, even for 48 concurrent requesters, DU results in lower network traffic than broadcast.

Figure 9.24a presents the Data overhead of the content source for static forwarders. The results for mobile forwarders in Figure 9.24b look very similar. Since multiple requests can be mostly satisfied by intermediate caches, content sources do not need to send more Data messages for concurrent requesters. Using DU with SFF for static forwarders, a content source sends less than 1% more Data messages up to 32 concurrent requesters and 13.5% more Data messages for 48 concurrent requesters. In all our evaluations, broadcast results in more Data transmissions by the content source. Although the Data overhead decreases with broadcast for 16 (and more) concurrent requesters in Figure 9.24a, this is mainly due to an over-



(a) Static Forwarding Nodes



(b) Mobile Forwarding Nodes with a Velocity of 1.2m/s.

Figure 9.24: Data Overhead of Content Source During Multi-hop Communication for Multiple Requesters.

loaded medium by flooded Interest and (duplicate) Data messages, i.e., each node has less time to access the medium and forward messages. Consequently, fewer Interests are forwarded to the content source and most Interests can be satisfied by intermediate nodes (forwarders or other requesters).

9.4.6 Content Request Tracker

We evaluate the performance of a Content Request Tracker (CRT) for multiple concurrent requesters in one-hop distance from a content source as shown in Figure 9.25. Table 9.6 lists the evaluation parameters. Between 1 and 100 requesters are placed at a distance of 75m from the content source.

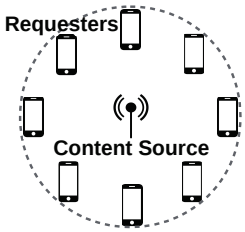


Figure 9.25: Static Topology: Multiple Requesters placed at Equidistance on a Circle around a Content Source.

Parameter	Value
Nodes	1 static source 1-100 static requester
Playground	circle around source distance: 75m (source - requesters)
Mobility	no mobility, static

Table 9.6: Evaluation Parameters for CRT Scenarios.

Parameter Selection for CRT

Different parameters could be selected as decision criterion for unicast or broadcast transmissions. For example, if the number of transmitted Data messages would be considered, transmissions would always be performed via broadcast for more than 1 concurrent requester. In this work, we select the content retrieval time as decision criterion.

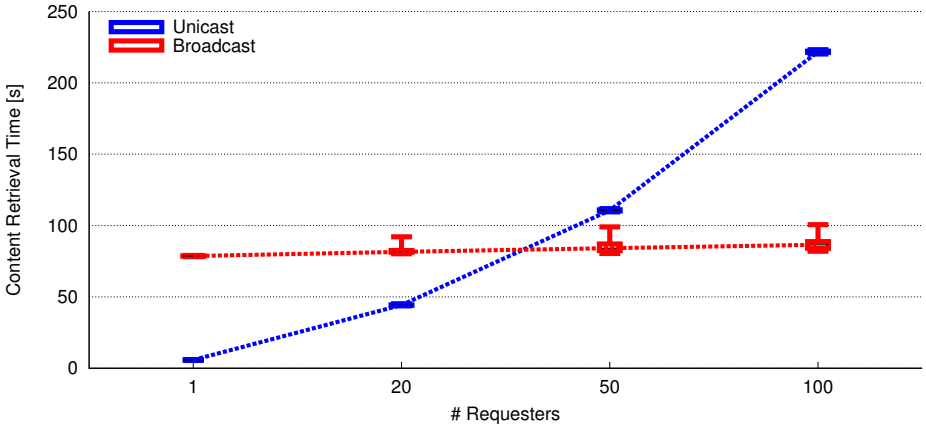


Figure 9.26: Content Retrieval Times for Static Requesters in One-hop Distance from a Content Source.

Figure 9.26 shows the content retrieval times (y-axis) for multiple concurrent requesters (x-axis) using broadcast or unicast transmissions. For only a few re-

questers unicast content retrievals are faster than broadcast but for more than 50 requesters, broadcast is faster. Based on this observation, we set *MAX_CRT* to 40 concurrent requests. At requesters we set *MAX_CRT_REQ* to 2 subsequent broadcast Data replies to unicast requests.

CRT-S vs. CRT-SR

Figure 9.27 shows content retrieval times for multiple concurrent requesters with unicast, broadcast, CRT-S (CRT only at content source) and CRT-SR (CRT at content source and requesters). Surprisingly, CRT-S becomes worse than unicast above

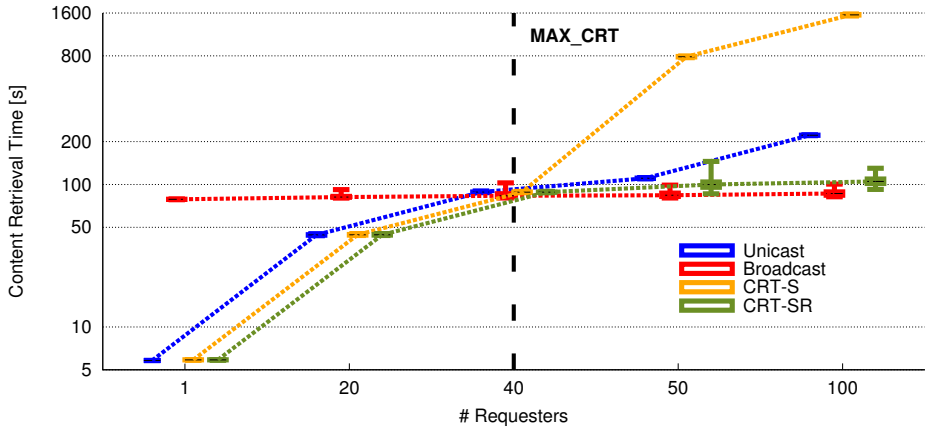


Figure 9.27: Content Retrieval Times of Multiple Requesters using Broadcast, CRT-S, CRT-SR and Unicast.

MAX_CRT. This is because requesters still transmit requests via unicast while the content source responds via broadcast (lower data rate than unicast). Then, if unicast Interests from requesters do not arrive at exactly the same time, the content source may broadcast the same Data messages multiple times as response to each Interest because it does not remember already transmitted Data messages. To improve the performance, requesters need to switch back to broadcast requests as well. CRT-SR performs similar to broadcast for many concurrent requesters, i.e., CRT-SR requires only 22% more time than broadcast for 100 requesters. Compared to unicast, CRT-SR results in 53% shorter content retrieval times for 100 requesters.

Figure 9.28 shows the Data overhead of the content source for multiple requesters. Below *MAX_CRT*, the number of Data messages transmitted by a content source using CRT increases linearly with the number of requesters similar to unicast. However, for more than *MAX_CRT* concurrent requesters, content sources with CRT-SR send only 15% more Data messages compared to broadcast because requesters switch to broadcast communication. Thus, the parameter *MAX_CRT* sets an upper bound on Data transmissions with CRT-SR. Lower *MAX_CRT* values may result in lower upper bounds but in longer content retrieval times.

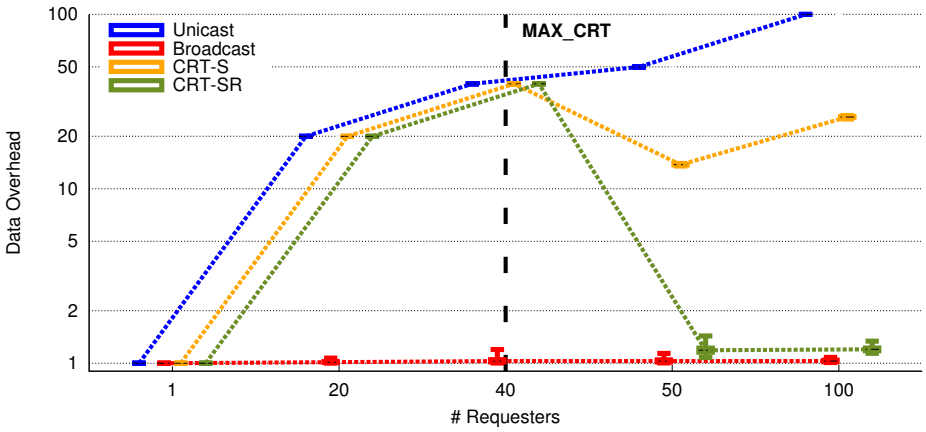


Figure 9.28: Data Overhead of Content Source for Multiple Requesters using Broadcast, CRT-S, CRT-SR and Unicast.

9.5 Lessons Learned

9.5.1 Efficiency of Dynamic Unicast

Multi-hop broadcast communication results in significantly longer content retrieval times than with Dynamic Unicast because of broadcast delays at each node and lower data rates. There is an overhead of Dynamic Unicast compared to unicast due to path setup via broadcast, but it is low, i.e., only 6% longer content retrieval times than for unicast in static multi-hop environments. In mobile networks, Dynamic Unicast may require more path setups than in static environments. However, unicast-only communication would not be possible at all because content sources cannot be configured statically.

9.5.2 One-hop vs. Multi-hop Communication

We have seen that multi-hop communication has clear advantages over one-hop communication because content can be retrieved from far away. However, one-hop communication is more efficient for very high content densities because multi-hop communication may establish non-optimal communication paths, e.g., to a requester's cache multiple hops away although there is a content source in one-hop distance. Enforcing content retrieval from (persistent) content sources may result in shorter path lengths for high content densities but results in worse performance for low content densities because users cannot benefit from cached content. Hop counters in Interest messages as supported in CCNx 1.0 may be a good way to limit Interest propagation. For example, requesters could perform Interest probing and successively increase the hop counter until they receive content back. However, such a strategy would require adaptation mechanisms for mobility (hop distances may change) and more complex Interest aggregation strategies in the PIT.

9.5.3 Routing in Mobile Networks

Evaluations have shown that Dynamic Unicast with single path routing (SFF strategy) performs better than multi-path routing (PFF strategy) in terms of content retrieval times and message overhead. Furthermore, Dynamic Unicast performs better than broadcast even for low content densities because short contact times to content sources can be better exploited. Although broadcast communication is efficient to quickly find a content source, it is not required to perform all communication via broadcast. Particularly for high content densities, Dynamic Unicast shows clear benefits compared to broadcast due to fewer duplicate Data transmissions.

9.5.4 Impact of Mobility

Dynamic Unicast performs best in static scenarios but the performance degrades only slightly if nodes move with pedestrian speeds. However, for vehicular speeds, transmission times increase significantly compared to static scenarios. The main reasons for the degradation are path breaks that are only detected after Interests have timed out. While the number of Interest transmissions increases slightly for more mobile scenarios, the number of transmitted Data messages remains nearly constant. Thus, adaptive Interest lifetimes based on measured round-trip times (see Chapter 10) are required to enable more seamless communication in high mobility scenarios. Surprisingly, the performance of broadcast communication decreases also with mobility. In particular, the number of transmitted Data messages during broadcast communication increases significantly in high mobility compared to static scenarios (in contrast to Dynamic Unicast), which indicates that symmetric Interest - Data forwarding paths may break more easily due to broadcast delays.

9.5.5 Scalability for Multiple Requesters

Evaluations with multiple requesters have shown that Interest messages can be efficiently aggregated in the PIT such that transmitted Interests do not drastically increase with increasing number of requesters (as long as the medium is not overloaded). In case of mobility, Interest aggregation works slightly less efficient since neighboring nodes, i.e., forwarding paths, may change slightly. Furthermore, Dynamic Unicast over multiple hops results in fewer Data transmissions than broadcast even for many concurrent requesters in static and mobile scenarios. Due to caching, most requests can be satisfied by intermediate caches and do not need to be forwarded to a content source.

9.5.6 Replacing Multiple Unicast Transmissions with One Broadcast Transmission

We have observed that replacing multiple unicast Data transmissions with one broadcast transmission as proposed with the Content Request Tracker is quite complex. It is not enough if only a content source switches to broadcast when re-

questers still transmit their requests via unicast. However, if requesters switch to broadcast requests as well, a content source has no means of knowing how many nodes are still interested in the content, i.e., whether broadcast is required. Particularly in mobile environments where connectivity changes frequently, the decision whether to use broadcast or unicast cannot be done once but needs to be re-evaluated periodically. Thus, instead of using CRT-SR in combination with Dynamic Unicast, i.e., reverting to broadcast communication in case of multiple requesters, it may be more efficient to avoid Dynamic Unicast, i.e., individual unicast links, for certain content prefixes in the first place, e.g., because the content is of high importance such as in emergency or disaster scenarios.

9.6 Conclusions

In this chapter, we have investigated information-centric routing in mobile and wireless ad-hoc networks. While broadcast is beneficial to quickly find a content source, it is not required to perform all message transmissions via broadcast. Instead, Dynamic Unicast enables requesters to retrieve content from the same content source until it becomes unavailable. We have described two Interest forwarding strategies for Dynamic Unicast as well as an optional Content Request Tracker to enable one broadcast transmission instead of multiple independent unicast transmissions. All mechanisms have been implemented in the CCNx framework and evaluated using NS3-DCE.

Evaluations have shown that NDN can improve scalability of wireless multi-hop communication since multiple requests can be aggregated and content can be retrieved from caches such that only a fraction of requests needs to be forwarded to content sources. Dynamic Unicast results in significantly shorter content retrieval times and fewer Data transmissions than broadcast for high content densities, but surprisingly it performed also better than broadcast for low content densities.

In future implementations, adaptive Interest lifetimes based on round-trip times may help to to detect path breaks quicker and, thus, improve performance of Dynamic Unicast in case of high mobility. Furthermore, the usage of different MAC protocols can be investigated such as IEEE 802.11p for vehicular networks or new MAC protocol designs tailored for information-centric wireless communication. In addition, hop counters in Interest messages may limit Interest propagation but they require new Interest aggregation strategies in the PIT. In particular, since requesters may have different distances from content sources, Interest aggregation mechanisms may need to handle different hop counter values of received and already forwarded Interest messages.

Chapter 10

Adaptive Interest Lifetimes to Support Information-Centric Wireless Multi-hop Communication

10.1 Introduction

Wireless communication is error-prone and collision probability increases in wireless multi-hop communication because received and forwarded packets can collide. In addition, connectivity to content sources may change with mobility such that established paths may break as shown in Chapter 9. To avoid throughput degradation, it is crucial to detect path breaks or collisions quickly such that retransmissions can be performed quickly. In ICN, retransmissions can only be performed if Interests have expired. For example, short Interest lifetimes enable fast recovery from collisions and increase throughput in opportunistic one-hop communication as we have shown in Chapter 5. In multi-hop scenarios, Interests need to be large enough to enable Interest messages to reach a content source multiple hops away and enable Data messages to return back to a requester. The closer an Interest lifetime matches a round-trip time (RTT) between a requester and a content source, the quicker retransmissions can be performed resulting in larger throughput.

Several adaptive Interest lifetime algorithms have been used in ICN literature [154, 65, 62, 231]. Yet, these works focus on throughput in wired networks and do not evaluate the number of unnecessary retransmissions due to too short Interest lifetimes. Wireless communication experiences higher RTT variability than wired communication due to varying channel conditions and MAC layer buffering. If timeouts are triggered too early, spurious retransmissions may be performed.

In this chapter, we describe two algorithms [53, 210] for wireless multi-hop communication, i.e., one that considers RTT variability and one that does not, and compare them to available algorithms from related work (see Section 2.5). We assume that requesters have dynamically established unicast faces to a content source (e.g., after a broadcast request as described in Chapter 9). By that, all segments are requested from the same content source until the path breaks and a new content

source is searched via broadcast. If a path breaks, the Interest lifetime is reset to the default value of 4 seconds to find a new content source.

While most existing works are based on simplified implementations and simulations, we have implemented all algorithms in CCNx 0.8.2 and evaluate them with NS3-DCE [16] not only in terms of content retrieval times but also message overhead. Once efficient algorithms have been identified in a single source scenario, extension to multi-homing, e.g., based on node identifiers [171, 66, 62, 40] could be implemented.

10.2 Adaptive Interest Lifetimes

In this section, we describe two newly developed algorithms to adaptively set the Interest lifetimes, namely CCNTimer and weighted moving average for CCNx (WMA). CCNTimer and WMA are illustrated in Algorithm 4. The parameters are listed in Table 10.1. CCNTimer uses the same exponential moving average for $sRTT$ and rttVAR as TCP [36] (see also Section 2.5). We measure the RTT as the time between the transmission of the first Interest in a segment and the reception of the corresponding segment. In case of timeouts, Interests may be retransmitted but the RTT start time of the corresponding segment is not changed because the segment has possibly been transmitted over some hops and may be cached in intermediate nodes. $Tout$ defines the number of timeouts for each segment. We only process RTT measurements if no timeouts occurred, i.e., no retransmissions of the same Interests. Thus, if a received segment experienced a timeout, the $sRTT$ value remains unchanged since there is no new RTT sample. Consequently, the old Interest lifetime IL is used for the next transmission.

Parameter	Meaning
$recv$	# received Data messages
Γ	threshold of received Data messages (for stability reasons)
T_{out}	# timeouts per segment
Φ	retransmission threshold (to reset algorithms)
γ	weighting factor for RTO
ϵ	multiplicative factor for IL (after timeouts)

Table 10.1: Parameter Overview for CCNTimer and WMA.

In addition to TCP and similar to TimeoutEstimator [62], $sRTT$ is weighted by a factor γ to obtain the Interest lifetime IL. This ensures that the Interest lifetime is slightly longer than the usual RTT, which avoids spurious retransmissions. While too long Interest lifetimes only have an influence in case of collisions (retransmissions), too short Interest lifetimes may already trigger retransmissions if Data messages are returned with a delay and, therefore, cause unnecessary traffic. In the worst case, too early Interest retransmissions may cause collisions with delayed Data packets.

Algorithm 4 CCNTimer / **WMA**

```
 1: initial values: IL = 4s, recv = 0, initial = true
 2:
 3: function RECEIVE_CONTENT(RTT, Tout)
 4:     recv++
 5:     if Tout == 0 then
 6:         if recv ≥ Γ then
 7:             if initial == true then
 8:                 sRTT = RTT
 9:                 rttVAR = RTT / 2                          ▷ not for WMA
10:                 initial = false
11:             else
12:                 sRTT = 0.8 × sRTT + 0.2 × RTT
13:                 rttVAR = 0.75 × rttVAR + 0.25 × abs(sRTT − RTT)   ▷ not for WMA
14:             RTO = sRTT + 4 × rttVAR                       ▷ for WMA: RTO = sRTT
15:             IL = max(4s, γ × RTO)

16: function TIMEOUT_OCCURRED(Tout)
17:     if Tout > Φ then
18:         IL = 4s
19:         initial = true
20:     else
21:         IL = max(4s, (1 + ϵ) * IL)
```

In contrast to TCP, CCNTimer does not double the Interest lifetime in case of timeouts because collisions of received and transmitted packets may happen regularly in wireless multi-hop networks, even if the traffic load is low. Longer Interest lifetimes, e.g., after burst errors, cannot avoid future collisions but only delay the detection and reaction to it. Therefore, if a retransmission is required for a segment, CCNTimer increases the Interest lifetime only slightly by the factor ϵ to account for delay variability. If Φ retransmissions are required for the same segment, CCNTimer resets the Interest lifetime to 4 seconds (default CCNx Interest lifetime), which we also set as maximum Interest lifetime. For stability reasons and to avoid high variability in the beginning of a content retrieval, we ignore the first Γ messages before processing the samples. In our evaluations, we use the following default numerical values, $\Gamma = 4$, $\gamma = 1.2$, $\epsilon = 0.2$ and $\Phi = 3$.

Besides CCNTimer, we have implemented WMA, which is a simpler version of Algorithm 4 without variance considerations, i.e., without lines 9, 13, 14. Similar to CHoPCoP [231], WMA uses an Interest lifetime that is only based on the exponential moving average of RTT values weighted by a factor γ. Since the specified γ is 6 [231], which would result in very long Interest lifetimes close or equal to 4 seconds (maximum value), WMA uses a smaller value of 1.5 to ensure that Interest lifetimes stay close to RTT values, i.e., only 50% higher. In case of timeouts, WMA increases the Interest lifetime only slightly by 20% as with CCNTimer.

10.3 Evaluation

We have implemented CCNTimer and WMA as well as TCP's RTO [36], TimeoutEstimator [62] and ICP [65] by integrating them into the *ccncat* application [162] of CCNx 0.8.2 [27]. We have also compared it to a constant Interest lifetime of 4 seconds, which is the default value in CCNx. The segment size was set to 4096 bytes and we used a pipeline size of 4. In this section, we show evaluation results in wireless multi-hop communication, which are obtained by emulations with NS3-DCE [16] using the IEEE 802.11g MAC layer and the free-space path loss model. All evaluations have been performed 100 times.

10.3.1 Implementation Details

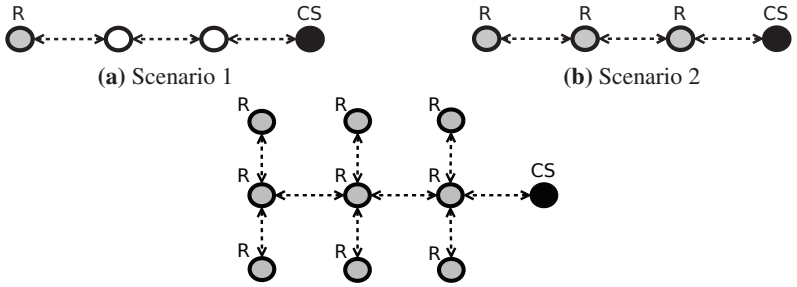
CCNTimer is similar to TimeoutEstimator [62], but we use the same weight for rttVAR as TCP and multiply the RTO with 1.2 instead of 2 to keep Interest lifetimes closer to RTT values. Since the behavior of TimeoutEstimator in case of timeouts is not specified, we assume the same behavior as for TCP, i.e., doubling the RTO. WMA is based on CHoPCoP's retransmission timer [231], i.e., ignoring the RTT variance, but we multiply the exponential moving average with a weighting factor of 1.5 instead of 6. For CCNTimer and WMA, we increase the Interest lifetime in case of timeouts by only 20% instead of doubling it as with TCP.

For CCNTimer, WMA, TCP and TimeoutEstimator, only RTT values of segments that have not experienced any timeouts are processed. Since these algorithms use an exponential moving average $sRTT$, large RTT values (due to Interest retransmissions) could distort $sRTT$ excessively and affect Interest lifetimes for a long time. For ICP we consider all measured RTT values, even for segments that experienced timeouts and required Interest retransmissions. We have also implemented an ICP variant that ignores timeouts similar to the other algorithms but the performance was significantly worse. Please recall that ICP uses a history of 20 RTT samples (see Section 2.5). If RTT samples with timeouts would be ignored, the algorithm could not adapt the Interest lifetime until the next segment is received in time. However, by considering RTT samples from segments that have experienced timeouts, higher RTT values are possible, which automatically increase the next Interest lifetime and enable a timely reception.

For all algorithms, we set the maximum Interest lifetime to 4 seconds because higher values resulted in worse performance. TCP's RTO has a lower bound of 1s and we set the lower bound of all other algorithms to 125ms (minimum in CCNx). In case of 3 consecutive timeouts of the same segment, the Interest lifetime is reset to the default value of 4 seconds for all algorithms. This enables algorithms to find another content source in case of mobility, e.g., if the content source has moved farther away.

10.3.2 Evaluation Scenarios

We evaluate all algorithms in wireless multi-hop communication with 1 to 5 hops. The network nodes are placed in a row so that only immediate neighbors see each other and every node has a unicast face configured to its neighbors.



(a) Scenario 1 **(b)** Scenario 2 **(c)** Scenario 3

Figure 10.1: Evaluation Scenarios: there are between 1 to 5 hops on the horizontal, here shown for 3 hops. The Content Source is on the rightmost node.

We evaluate the algorithms in three scenarios. Scenario 1 consists of two active nodes: one requester that requests a content object and a content source, which is 1 to 5 hops away. Figure 10.1a shows the topology for scenario 1 with 3 hops. The requester (R) is the leftmost node and the content source (CS) the rightmost node. In scenario 2, there is one requester at every node requesting a different content object. Figure 10.1b shows the topology for scenario 2 with 3 hops. Scenario 3 is similar to scenario 2, but every requester additionally receives requests for different content objects from two requesters above and below. Figure 10.1c shows the topology for scenario 3 with 3 hops.

Please note that we focus on static network topologies to analyze the algorithms in multi-hop communication with varying traffic loads. If synthetic mobility models would be used, mobility and routing would have a significant impact on the performance of content transmissions making algorithm analysis less traceable. For example, if requesters and content source are too far apart, no multi-hop communication is possible and if they are too close, direct communication (no multi-hop) could be performed. However, we would like to emphasize that human mobility does not always include continuous movements, but is often composed of long (nearly) static periods, where the same neighbors are seen, e.g., at work, at home or in the train. These static periods are followed by transient mobility with high host churn, e.g., when walking from work to the train station before entering the train. In this chapter, we do not focus on transient mobility but on static periods in-between. Whenever changing the environment, a new content source needs to be found via routing algorithms. Routing itself is out of the scope of this work and we assume that requesters have found the content source, e.g., via flooding, and configured a unicast face towards it (as described in Chapter 9). If the content source moves away, content retrieval will experience multiple timeouts for the

same segment, which cause the algorithms to reset the Interest lifetime to the initial configuration, i.e., 4 seconds, until the next content source is found.

10.3.3 One Requester - Varying Path Length

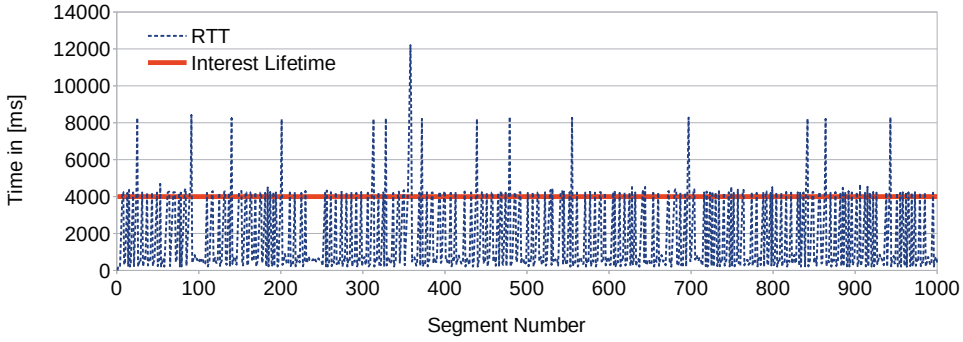
In this section, we evaluate the algorithms in scenario 1, between one requester and one content source. To better understand the algorithms, Figures 10.2 and 10.3 show the measured RTT (blue dashed lines) and the Interest lifetime (red solid lines) values of a sample run for all evaluated algorithms over 3 hops. Since the RTT is measured between the first Interest transmission and the reception of the corresponding segment, i.e., incorporating Interest retransmissions, the RTT values are slightly different for all algorithms. If collisions are detected quicker (due to shorter Interest lifetimes), retransmissions can be performed faster resulting in lower RTT values.

Figure 10.2a shows the results if the default CCNx strategy, i.e., a constant Interest lifetime of 4 seconds, is used. RTT values of 4 seconds indicate collisions because regular RTT values are much shorter and Interest retransmissions are only triggered after 4 seconds (Interest expiration). If RTT values are longer than 4 seconds, multiple timeouts of the same segments have occurred. This shows that even with large Interest lifetimes, collisions can not be avoided, but due to longer detection times, RTT values become larger.

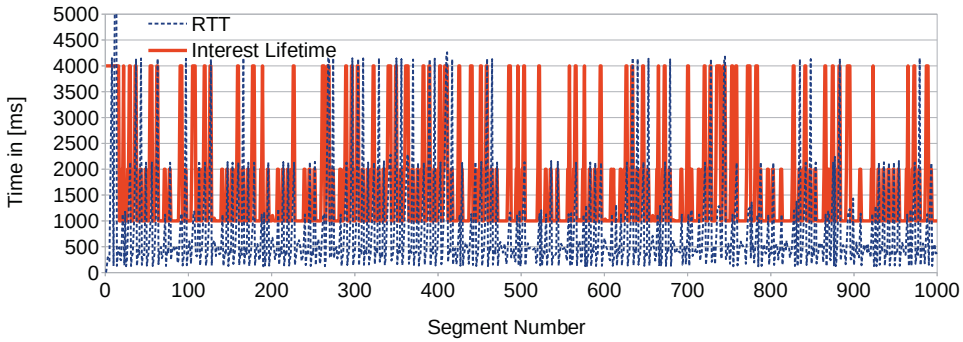
Figure 10.2b shows Interest lifetime and RTT values for TCP's RTO. Because most RTT values are below 1 second, Interest lifetimes are often set to the lower bound. However, due to high variability, some RTT values may be slightly larger than 1 second and, therefore, cause a timeout. These timeouts are visible in Figure 10.2b as RTT values slightly above 1 second, which indicates that retransmissions could be satisfied from the cache. Since Interest lifetimes are doubled in case of timeouts, these unnecessary timeouts cause Interest lifetimes to be increased to 2 seconds. As a result, collision detection takes more time and RTT values increase.

Figure 10.2c shows Interest lifetimes with TimeoutEstimator, which is similar to TCP but without lower bound, i.e., the lower bound of 125ms was never reached in any measurements. Since RTOs are multiplied by 2, the resulting Interest lifetimes are still larger than most RTT values, avoiding too early retransmissions. In addition, Interest lifetimes can adopt more diverse values because they are not always set to the lower bound as TCP's RTO. More diverse values enable TimeoutEstimator to adjust the Interest lifetimes better to current round-trip times, resulting in slightly lower RTT values compared to TCP's RTO. However, because RTOs are doubled in case of collisions, Interest lifetimes can quickly reach high values, which cause higher RTT values in case of collisions.

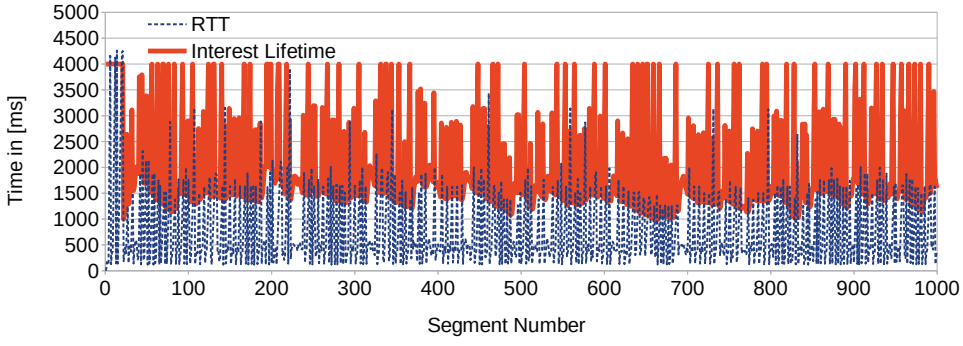
In Figure 10.3a, we show Interest lifetimes of CCNTimer. In contrast to TimeoutEstimator, the RTOs use the same weight for the RTT variance as TCP's RTO, i.e., not half the weight as for TimeoutEstimator, and the RTOs are multiplied with a smaller weight of 1.2 to account for higher RTT variability but ensure that Interest lifetimes are only slightly larger than most RTT values. In case of timeouts, the In-



(a) default 4s

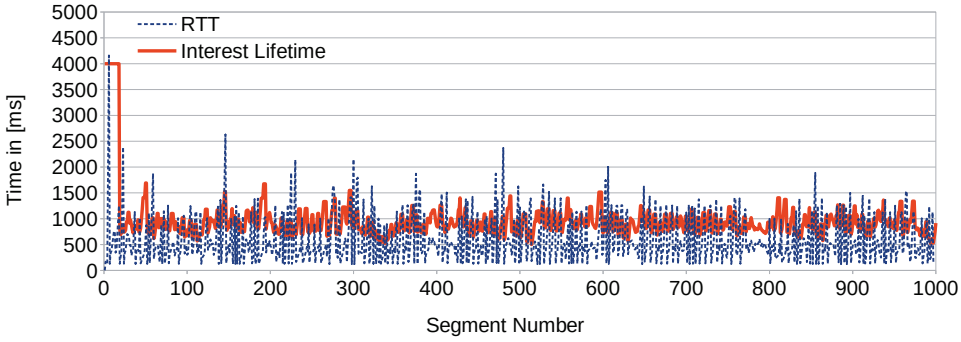


(b) TCP's RTO

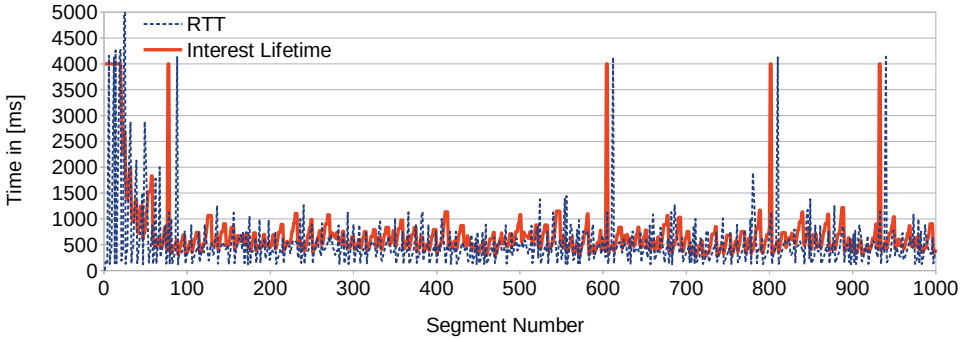


(c) TimeoutEstimator

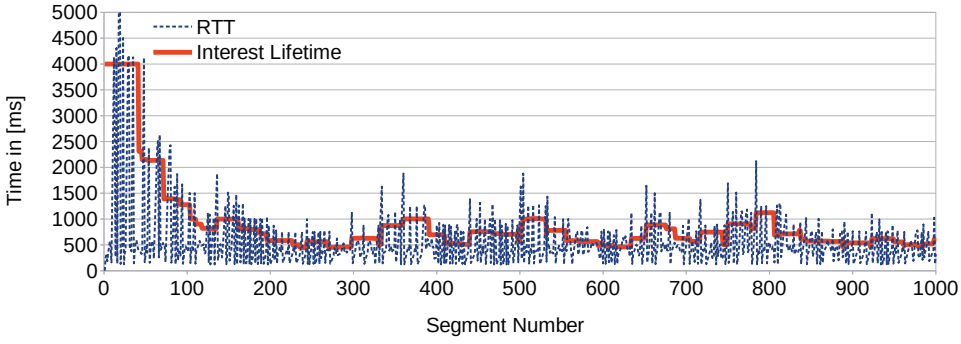
Figure 10.2: RTT and Interest Lifetime Values of the Default CCNx Strategy, TCP's RTO and TimeoutEstimator from a Sample Run over 3 Hops for the First 1000 Segments.



(a) CCNTimer



(b) WMA



(c) ICP

Figure 10.3: RTT and Interest Lifetime Values of CCNTimer, WMA and ICP from a Sample Run over 3 Hops for the First 1000 Segments.

terest lifetime is only slightly increased by 20% for every timeout and not doubled as for TCP's RTO and TimeoutEstimator. Timeouts are mostly caused by higher RTT variability and collisions. In both cases, the content might be found in caches at intermediate nodes. Therefore, CCNTimer can react quicker to timeouts and does not overcompensate them by increasing the Interest lifetime extensively. This enables CCNTimer to reduce timeout periods, which leads to lower RTT values as Figure 10.3a shows. There are a few segments that time out and cause RTT values of slightly more than 2 seconds, e.g., segment 149, 232 and 302, but the Interest lifetime does not drastically react to it and only increases slightly. As Figure 10.3a shows, this is enough to receive the next few segments without timeout.

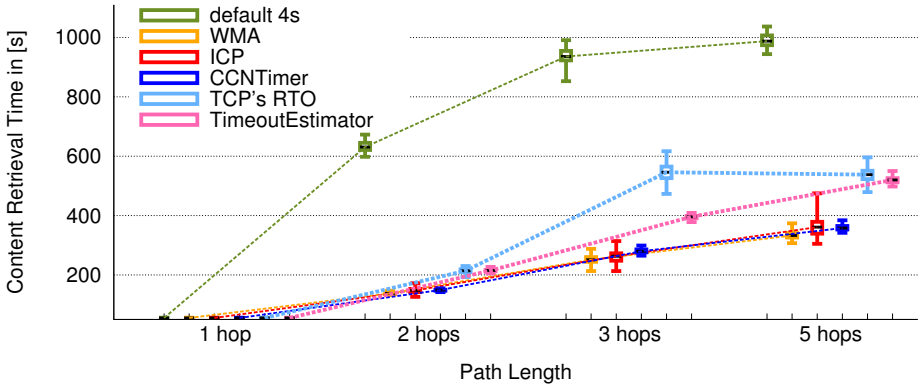
Figure 10.3b shows Interest lifetimes and RTT values for WMA. The RTT values are on a slightly lower level than with CCNTimer due to shorter Interest lifetimes, which reduce timeout periods. However, there are four peaks of Interest lifetimes and RTT values. Because Interest lifetimes may be slightly too short, i.e., RTT variance is not considered, timeouts happen more frequently and for segments 80, 606, 802 and 934 in Figure 10.3b, the Interest lifetime is reset to 4 seconds because of 3 consecutive timeouts of the same segment. If segments, which are transmitted with an Interest lifetime of 4 seconds, collide, the RTT values become considerably larger since it takes more time to detect the collision. We can see such RTT peaks at segments 90, 614, 812 and 942, i.e., after every reset of the Interest lifetime. This indicates that WMA results in many timeouts.

Finally, Figure 10.3c depicts Interest lifetimes of the ICP algorithm. The Interest lifetimes do not change as quickly as for the other algorithms because Interest lifetimes are only based on minimum and maximum RTT values of the last 20 samples. Figure 10.3c shows that the Interest lifetime changes in cycles. For small minimum RTT values, the Interest lifetime becomes slightly too small resulting in timeouts. Timeouts require retransmissions, which means that RTT values increase resulting in larger Interest lifetimes. Larger Interest lifetimes result in fewer timeouts and, thus, enable lower RTT values completing the cycle.

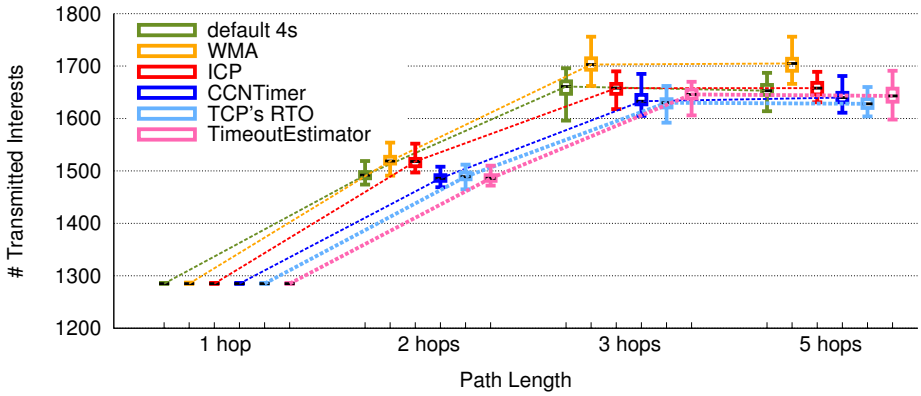
Algorithm	2 hops		3 hops		5 hops	
	Time	Interests	Time	Interests	Time	Interests
CCNTimer	*reference*		*reference*		*reference*	
default 4s	+324%	0%	+234%	+1%	+176%	+1%
TCP's RTO	+44%	0%	+95%	0%	+50%	-1%
TimeoutEst.	+44%	0%	+41%	+1%	+45%	0%
WMA	-7%	+2%	-10%	+4%	-7%	+4%
ICP	-1%	+2%	-6%	+2%	+1%	+1%

Table 10.2: Content Retrieval Times and Transmitted Interests relative to CCNTimer (Scenario 1).

To measure algorithm performance, Figure 10.4a illustrates *content retrieval times* of a 5 MB content object from a content source multiple hops away and



(a) Content Retrieval Time



(b) Transmitted Interests

Figure 10.4: Content Retrieval Times and Transmitted Interests of a Requester in Scenario 1 for Varying Path Lengths.

Figure 10.4b shows transmitted Interest messages by the requester. Table 10.2 lists the relative differences of all algorithms compared to CCNTimer in percentages, i.e., overhead in terms of content retrieval times and transmitted Interest messages.

For two hops or more, Interests need to be forwarded by intermediate nodes and may collide with other Interest or Data messages. As a result, adaptive Interest lifetimes result in shorter retrieval times than the default Interest lifetime of 4 seconds. However, with increasing path length, the performance of adaptive algorithms decreases slightly compared to the default 4 seconds due to longer content retrieval times and more collisions. CCNTimer results in notably shorter retrieval times than TCP and TimeoutEstimator without requiring more Interest transmissions because Interest lifetimes are closer to RTT values, i.e., the algorithm can react faster to collisions. However, requesters using WMA or ICP have slightly shorter retrieval times (due to shorter Interest lifetimes) and require only slightly

more Interest transmissions than CCNTimer. Although the relative difference of Interest transmissions compared to CCNTimer seems to be low, it is worth mentioning that ICP and WMA result in significantly more timeouts, e.g., 3.59 times more timeouts with ICP and 2.2 times more timeouts with WMA for 3 hops compared to CCNTimer, while the other algorithms are on the same level in terms of timeouts. This means that Interest lifetimes with ICP and WMA are only slightly too short but retransmissions do not need to be sent on the wireless medium in most of the cases but can be satisfied from the local cache, i.e., delayed Data arrival.

Thus, in low traffic scenarios, WMA and ICP perform best in terms of content retrieval times. However, if transmitted Interests and timeouts are relevant, e.g., if algorithms may also experience occasional high traffic, CCNTimer performs better.

10.3.4 Multiple Requesters - Varying Path Length

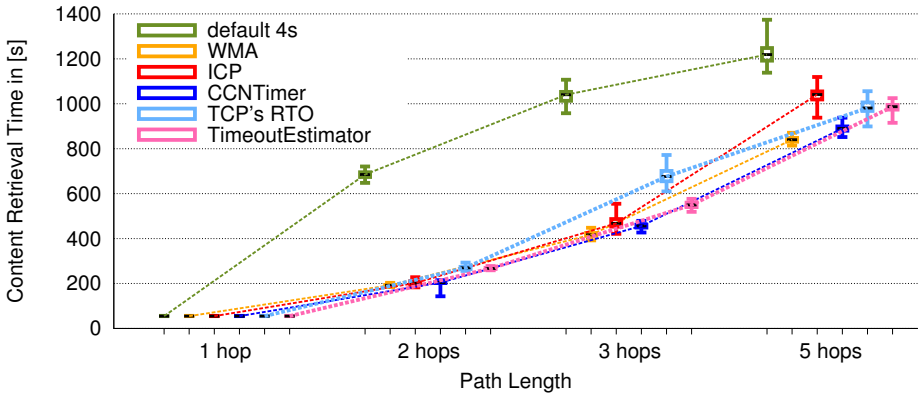
In this section, we evaluate the algorithms in scenario 2, where every node requests content from a content source. Every requester uses the same algorithm and we measure the statistics at the leftmost requester. However, the relative performance of the algorithms at other requesters is similar.

Algorithm	2 hops		3 hops		5 hops	
	Time	Interests	Time	Interests	Time	Interests
CCNTimer	*reference*		*reference*		*reference*	
default 4s	+235%	0%	+128%	-1%	+37%	-6%
TCP's RTO	+32%	0%	+48%	-1%	+10%	-1%
TimeoutEst.	+31%	0%	+21%	0%	+11%	-2%
WMA	-6%	+2%	-9%	+4%	-6%	+6%
ICP	-1%	+2%	+3%	0%	+17%	-2%

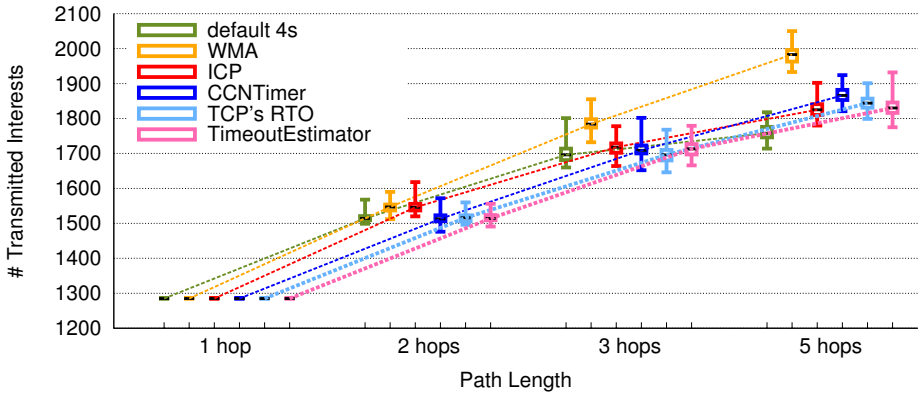
Table 10.3: Content Retrieval Times and Transmitted Interests relative to CCNTimer (Scenario 2).

Figure 10.5a shows content retrieval times of the leftmost requester for a 5 MB content object over multiple hops and Figure 10.5b shows the number of transmitted Interests at the leftmost requester. Table 10.3 lists the overhead of all algorithms compared to CCNTimer. CCNTimer performs better than TCP's RTO and TimeoutEstimator for content retrievals over 2 to 5 hops, i.e., significantly shorter content retrieval times and similar (or only slightly more) Interest transmissions. ICP performs similar to CCNTimer between 2 hops and 3 hops but it's performance degrades from 3 hops to 5 hops due to a higher collision probability. Since ICP incorporates all RTT measurements including retransmissions, RTT values can be considerably larger than with other algorithms. In case of frequent timeouts, there is always at least one large RTT value, which causes the Interest lifetime to remain at 4 seconds (maximum value).

WMA results in slightly faster content retrieval times than CCNTimer, but it is at the expense of more Interest transmissions. In fact, Figure 10.5b shows that



(a) Content Retrieval Time



(b) Transmitted Interests

Figure 10.5: Content Retrieval Times and Transmitted Interests of a Requester in Scenario 2 for Varying Path Lengths.

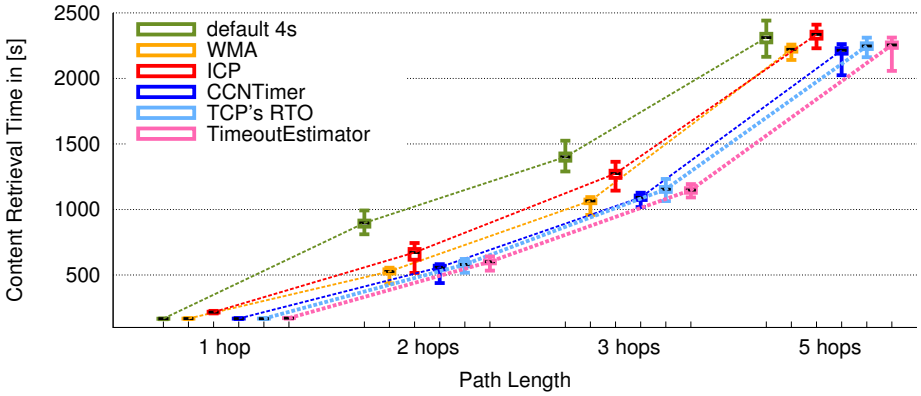
WMA results in more Interest transmissions in all scenarios and the difference to CCNTimer becomes bigger with increasing path length. Thus, CCNTimer shows the best overall performance (content retrieval times and transmitted Interest messages).

10.3.5 Multiple Streams - Varying Path Length

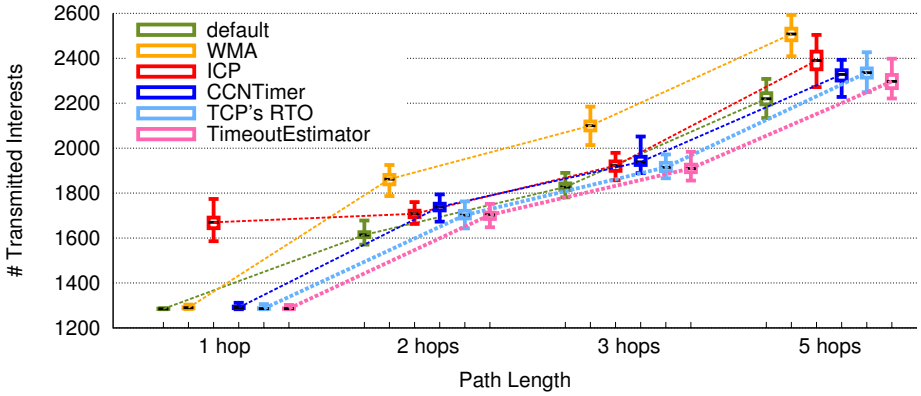
In this section, we evaluate the algorithms in scenario 3, where each requester additionally receives and forwards Interests from two requesters (above and below) as illustrated in Figure 10.1c. This results in considerably more traffic in the network. Figure 10.6a and Figure 10.6b show content retrieval times and transmitted Interests for a 5 MB content object at the leftmost requester in the middle, i.e., same requester as in previous scenarios. Table 10.4 lists the overhead of all algorithms compared to CCNTimer.

Algorithm	2 hops		3 hops		5 hops	
	Time	Interests	Time	Interests	Time	Interests
CCNTimer	*reference*		*reference*		*reference*	
default 4s	+60%	-7%	+28%	-6%	+4%	-5%
TCP's RTO	+4%	-2%	+6%	-1%	+1%	0%
TimeoutEst.	+8%	-2%	+5%	-2%	+2%	-1%
WMA	-6%	+7%	-2%	+8%	0%	+8%
ICP	+20%	-2%	+17%	-1%	+6%	+3%

Table 10.4: Content Retrieval Times and Transmitted Interests relative to CCNTimer (Scenario 3).



(a) Content Retrieval Time

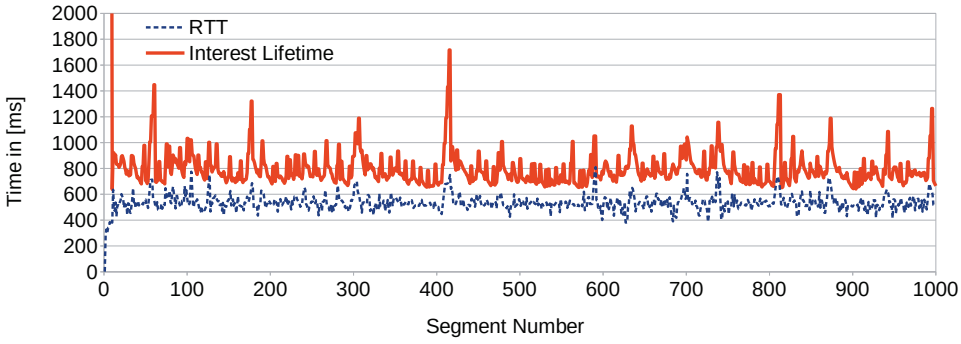


(b) Transmitted Interests

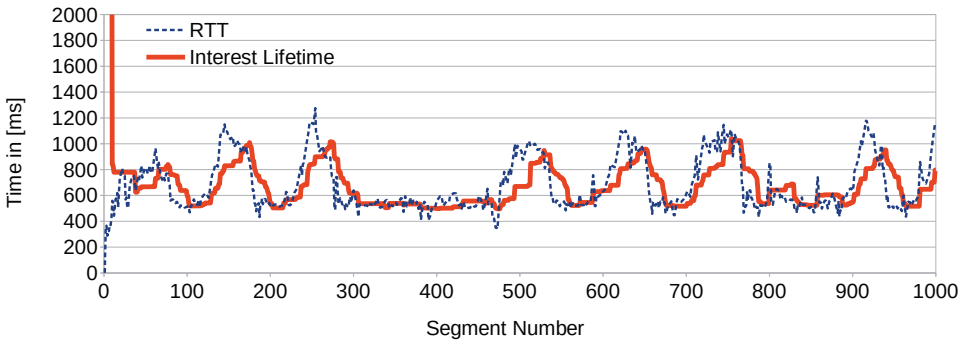
Figure 10.6: Content Retrieval Times and Transmitted Interests of a Requester in Scenario 3 for Varying Path Lengths.

CCNTimer, TCP's RTO and TimeoutEstimator perform similarly because there are many transitions to Interest lifetimes of 4 seconds, i.e., TCP's RTO and Timeout-

Estimator have slightly longer content retrieval times but also slightly fewer Interest transmissions. WMA results in shorter content retrieval times than CCNTimer only for short path lengths, but with increasing path length the performance of WMA degrades. For example, for 5 hops, WMA has equal content retrieval times as CCNTimer but notably more Interest transmissions. ICP results in significantly longer content retrieval times than CCNTimer in all scenarios because Interest lifetimes remain mostly at 4 seconds similar to scenario 2. However, in this scenario, ICP performs particularly badly over 1 hop, i.e., it requires 29% more Interests and results in 29% longer content retrieval times than CCNTimer.



(a) CCNTimer



(b) ICP

Figure 10.7: RTT and Interest Lifetime Values from a Sample Run over 1 Hop for the First 1000 Segments.

To better see differences and understand why ICP sends more Interests over 1 hop, we show Interest lifetimes and RTT values for CCNTimer and ICP over 1 hop in Figure 10.7. Recall that there are two additional requesters above and below the requester, which cause significantly more traffic and RTT variability compared to scenario 1. Figure 10.7a shows that Interest lifetimes with CCNTimer are slightly larger than RTT values but they adapt quickly to changing RTT values. For example, at segment 418 and 811 the Interest lifetime increases due to higher RTT variance. Figure 10.7b shows the Interest lifetime and RTT values for ICP.

While ICP's retransmission timer may calculate a good average estimation, the algorithm underestimates RTT values in case of high RTT variation resulting in too short Interest lifetimes. Consequently, Interest timeouts trigger retransmissions resulting in larger RTT values and Interest lifetimes, which again enable shorter RTT values without timeouts, i.e., there are cycles between low and high values. Due to these peaks in Figure 10.7b, ICP results on average in higher RTT values than CCNTimer.

Overall, we can conclude that slightly too short Interest lifetimes (WMA, ICP) do not result in any benefits in high traffic scenarios. Compared to CCNTimer, they either result in longer content retrieval times with a similar or higher amount of transmitted Interests (ICP), or in similar content retrieval times but considerably more Interest transmissions (WMA).

10.4 Caching and Multi-homing

In this section, we discuss caching and multi-homing, which require more investigations. Currently, we do not process RTT samples if Interests have been retransmitted, although the segments may be found in intermediate caches and do not need to be retrieved all the way from the content source. However, before an Interest can be retransmitted, an Interest timeout needs to be triggered, which results in imprecise RTT samples.

Even if Interests have not been retransmitted, the same content could be found in intermediate caches due to earlier downloads from other requesters. However, since NDN follows a sequential request strategy, NDN caches may contain contiguous blocks of subsequent Data segments but not random segments. Thus, algorithms could retrieve cached segments slightly faster resulting in shorter RTT and Interest lifetime values. If only part of the content could be found in the cache, e.g., only the beginning of a content object, Interest lifetimes would become too short to retrieve the remaining parts of the content from the original content source resulting in timeouts. If the same segment times out multiple times, the algorithm could reset to the initial configuration (Interest lifetime of 4s) to retrieve content and process RTT values from the original content source.

In this work, we only considered one content source because we assumed that routing algorithms enabled requesters to build unicast routes to the content source. Furthermore, all requesters retrieved different content objects. However, if all requesters would retrieve the same content object concurrently, partial content may be found in caches. To support multi-homing and differentiate RTT measurements from different responding nodes (content source or intermediate caches), existing works [171, 66, 62, 40] propose the usage of node identifiers. All described algorithms in this chapter could be extended to support multi-homing by tracking node identifiers of responding nodes. However, to enable multi-homing via node identifiers, modifications to CCNx would be required to include node identifiers of responding nodes in Data messages, i.e., either by content source or intermediate

cache. Since any intermediate node that answers an Interest from cache should, then, be able to modify the node identifier field (such that requesters can differentiate RTT measurements from different responding nodes), a node identifier field cannot be protected by the signature of the original content source making it vulnerable to attacks. In addition, node identifiers in ICN messages may have implications for message forwarding and caching efficiency. Future works that address multi-homing, should, therefore, also consider such implications and associated security challenges.

10.5 Conclusions

Adaptive algorithms for Interest lifetimes are beneficial in wireless networks to control how quickly retransmissions can be performed in case of collisions or path breaks. Fast retransmissions have a large impact on content retrieval times in wireless information-centric networks. Since the number of transmitted packets is limited by the pipeline size, new Interests can only be transmitted if Data is received in return or unsatisfied Interests time out. Thus, all evaluated algorithms resulted in considerably faster content retrieval times than the default 4 seconds in CCNx.

Evaluations in a low traffic scenario (scenario 1) have shown that CCNTimer results in significantly shorter content retrieval times than TCP's RTO and TimeoutEstimator without transmitting more Interests. In high traffic scenarios (scenario 3), CCNTimer performs similar to TimeoutEstimator and TCP's RTO regarding content retrieval times and transmitted Interests. Furthermore, we have seen that it is not required to double Interest lifetimes in case of timeouts. Timeouts are mostly caused by higher RTT variability and collisions. In both cases, the content might be found in caches of intermediate nodes. Evaluations have shown that increasing the Interest lifetime only slightly (CCNTimer) is enough to receive subsequent segments without timeouts.

It is important that Interest lifetimes are larger than RTT values. Although algorithms with slightly too short Interest lifetimes, e.g., ICP and WMA, may result in slightly faster content retrieval times than CCNTimer and TimeoutEstimator in low traffic scenarios, it is at the expense of significantly more timeouts and Interest transmissions. Consequently, ICP and WMA perform worse in high traffic scenarios, i.e., longer content retrieval times and more Interest transmissions than CCNTimer.

Strategies that are based on the history of RTT samples are much more complex than exponential moving averages because they need to maintain recent values in a buffer as well as compare and update minimum and maximum values for every new sample. Our evaluations did not show any benefits of such strategies compared to exponential moving averages. If the midrange of the samples is considered, Interest lifetimes may often be slightly too small resulting in more frequent timeouts. Then, in case of frequent timeouts, Interest lifetimes may remain at the maximum value because of large maximum RTT samples.

Chapter 11

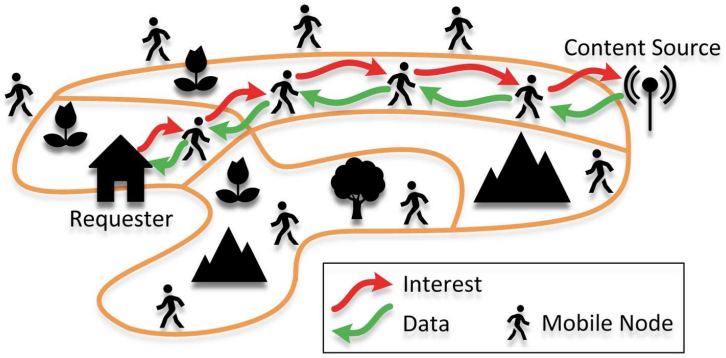
Agent-based Content Retrieval for Information-Centric Delay-Tolerant Networks

11.1 Introduction

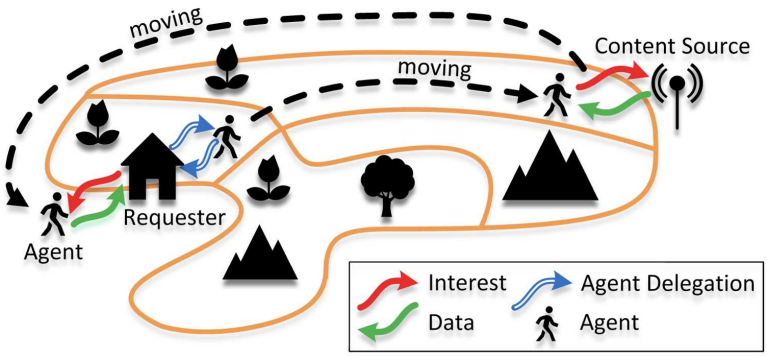
If a node never meets a content source directly, it needs to forward requests to other nodes. NDN requires symmetric Interest-Data forwarding paths because Data travels on the reverse path (of Interests) back to requesters. If there is a continuous path between requester and content source, multi-hop routing can be used as described in Chapter 9. However, in case of intermittent connectivity, Interests may either time out at intermediate nodes such that they never reach a content source or the reverse path may break due to topology changes such that Data can not reach the requester (asymmetric forwarding paths). Then, if segments are stored at different intermediate nodes (since some nodes may have move away), it may be difficult to retrieve the complete content via NDN's sequential request strategy.

In this chapter, we investigate DTN support via agent-based content retrieval (ACR) [45, 34, 49, 177]. In ACR, requesters can delegate content retrieval to mobile nodes (called agents), which will retrieve content on their behalf. Requesters can then retrieve content from agents when they meet them again. Mobility studies [99] have shown that human mobility exhibits temporal and spatial periodicity. This means that people tend to have strong location preferences in their daily mobility and meet other individuals regularly. ACR can exploit this property since agents (with different mobility patterns as requesters) can request content in locations requesters would never visit.

In NDN, routing and caching is performed on every node at the CCND and additional functionality can be provided by application modules above the CCND (see Section 2.1.1). If applications provide a certain service or content, they can register prefixes (Interest filters) at the CCND to receive requests from other nodes or applications. This means that the CCND includes only vital ICN functionality (to limit the processing complexity such that line-speed operations can be suppor-



(a) Multi-hop Communication in Dense Environments



(b) Agent-based Content Retrieval in Sparse Environments

Figure 11.1: Content Retrieval in Different Environments.

ted) and individual nodes can have extended functionality by running additional applications.

Consequently, we implement ACR as application module to be run on mobile nodes. By this, ACR can be combined with multi-hop routing (since message processing at the CCND is not modified) enabling interoperability in dense and sparse environments. For example, a requester could initially try to retrieve content via multi-hop routing if the node density is high (Figure 11.1a). However, if the node density is low, multi-hop content retrieval would not be successful. Then, requesters may delegate content retrieval to agents, which move closer to content sources, can retrieve content and deliver it back to requesters as illustrated in Figure 11.1b. Furthermore, requesters can switch back to multi-hop routing at any time to retrieve content from content sources or agents if the node density is sufficient to enable efficient multi-hop communication.

11.2 Relations to classical DTN Protocols

DTN protocols such as the Bundle protocol [179] exchange so-called Bundle packets upon node encounters. Combining the Bundle protocol with NDN is difficult due to potentially long delays between encounters that may result in the expiration of NDN Interest messages. Increasing the Interest lifetime to obtain long-lived Interests may enable the integration with existing DTN protocols but it would result in two major drawbacks:

1. Since content is organized in segments and content retrieval is pull-based, multiple Interests in segments are required to obtain the complete content. In general, a requester does not know the length of the requested content until it receives the last segment. Therefore, proactive Interest transmissions to request all segments are not possible. Assuming a very high number of segments may obviously result in many unnecessary Interest transmissions and inefficient resource utilization if the content has much fewer segments. Since all entries are valid for a long time, the increasing PIT size would drastically degrade PIT lookup performance.

2. Long-lived Interests in the PIT prevent forwarding of similar Interests because the requests are already pending. Thus, forwarding and retransmission is blocked for the entire duration of an Interest lifetime, even if the environment has changed due to mobility and a content source would be available. If pending (long-lived) Interests would be re-expressed periodically [213] before they expire, it would compromise NDN's ability to aggregate Interests in the PIT. Interests would only be forwarded to the next hop (where the PIT entry already exists) and applications would lose control when Interests are (re-)transmitted.

As evaluations in Chapter 5 and 10 have shown, short Interest lifetimes may result in higher wireless throughput because retransmissions due to packet collisions can be triggered earlier. Therefore, Interest lifetimes should be rather short and used to retransmit Interests in case of collisions. If multiple immediate retransmissions are not successful, it can be assumed that no content source is available in the current environment and the retrieval can be postponed to a later time. In most DTN approaches, nodes exchange hello beacons to learn about their neighbors and subsequently connect to each neighbor to discover and exchange content. This is not required in NDN: users can broadcast requests and only nodes that can provide the desired content (or service) will reply. The lack of a reply means that no neighbor node can (or is willing to) provide the requested content, which - in terms of content retrieval - is equivalent to the unavailability of neighboring devices.

11.3 Agent-based Content Retrieval

In ACR, requesters can delegate content retrieval to agent nodes. In most cases, the exact content size may not be known. Thus, it is impossible to transfer a sufficient and not extensive amount of Interests (for every content segment) to the agent. However, this is not required since only the task to retrieve content is transferred. Agents are implemented as application modules and can perform content retrieval independently. Later, after receiving a notification from the agent, the requester can regularly retrieve the content from the agent via multiple Interests. ACR is performed in three phases, which are described in the following subsections.

11.3.1 Phase I: Agent Delegation

Agent delegation describes the process of finding an agent and delegating content retrieval to it. The proposed agent delegation sequence is presented in Figure 11.2.

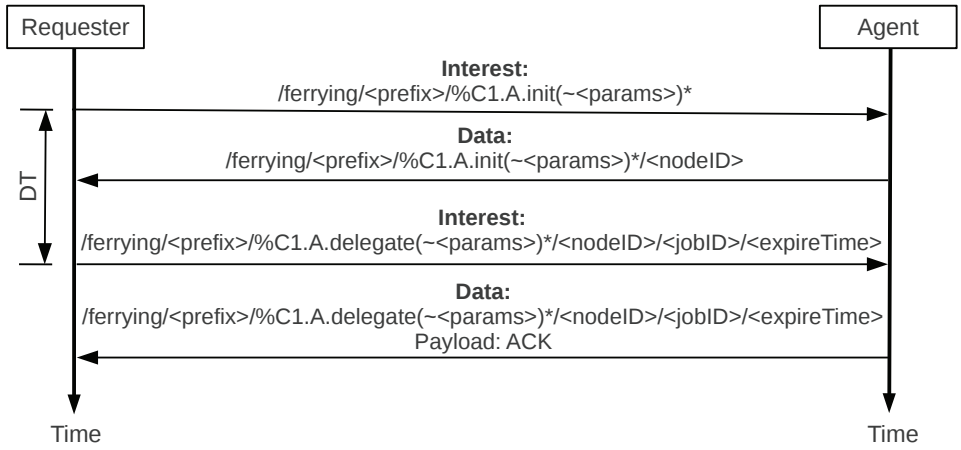


Figure 11.2: Message Sequence during Agent Delegation.

A requester broadcasts an Exploration Interest with the prefix */ferrying*, the content name <*prefix*> and optional selection parameters, e.g., coordinates where the content may be found, to its one-hop neighbors. Agents have registered Interest filters for the prefix */ferrying* at the CCND to receive Exploration Interests. If agents have sufficient resources to perform the task and agree with the optional parameters, they reply to an Exploration Interest with Exploration Data appending their nodeID, which uniquely identifies the agent, in the name. Since the Exploration Interest is broadcast (see Figure 11.3), the requester may receive multiple Exploration Data replies from agents in one-hop distance. The requester can then create an agent list by retrieving other replies from its local cache (using Exploration Interests with Exclude filters [5] containing already known nodeIDs). Exploration Data has a short lifetime of only a few seconds to avoid usage of old information from the cache. After a short delegation time (DT, in our implementation: 2s),

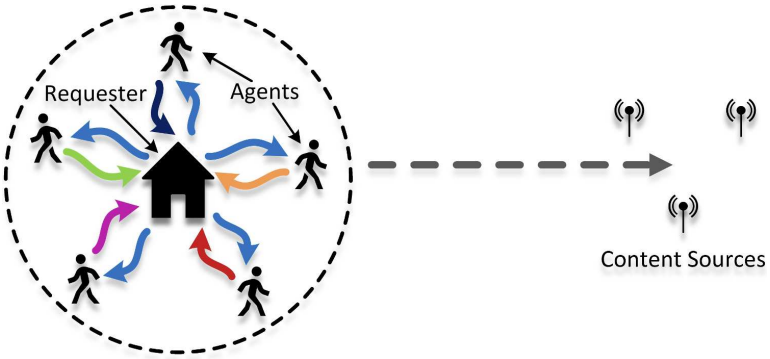


Figure 11.3: Agent Delegation: a Broadcast Exploration Interest from a Requester may retrieve Multiple Exploration Data Replies from Different Agents.

during which additional responses may be retrieved from the cache, the requester can select an agent from the agent list for delegation. Agent selection can be based on diverse criteria such as social relations or past GPS traces. For example, if a requester knows at which location or in which area desired content can be found, it can indicate this as optional parameter in Exploration Interests such that only agents who travel in this area may respond. Since most smart phones nowadays store GPS traces locally, an agent can simply examine it's locally collected GPS traces without requiring to transmit or share it with others. However, for the sake of simplicity, we randomly select agents from the agent list in this work and leave more sophisticated criteria as well as multi-hop agent selection algorithms for future work.

The requester sends a Delegation Interest to the selected agent using its nodeID, a jobID, an expiration time and optional parameters such as the notification type *push* or *pull*. The jobID is used in the notification phase (see below) and the expiration time limits the duration that an agent is looking for the content. As last step, the agent has to confirm the delegation with an acknowledgment (ACK). The last step is required to ensure that the agent has received the delegation and has not moved away, i.e., without acknowledgment a requester may wait infinitely long for content if the agent has never received the delegation. An agreement between requester and agent could be enforced by signing the exchanged Interest and Data messages with the sender's private key so that both nodes can learn the identities of each other (even if they are a pseudonyms) via the corresponding certificates. Knowing the identities may help to implement incentives and avoid denial of service attacks. Agent delegation can be performed to one or multiple redundant agents up to an agent limit (maximum number of delegated agents).

Please note that an early version of agent-based content retrieval [45], which we evaluate in Section 11.4, uses a slightly different implementation than described here. First, it uses only pull notifications, thus, there is no option to select the notification type in the Delegation Interest. Second, no acknowledgment has been transmitted after the Delegation Interest. The acknowledgment has only been introduced after investigating ACR in mobile environments.

11.3.2 Phase II: Content Retrieval

After agent delegation, the agent can find and retrieve content for the requester by periodic Interests (Interest probing, in our implementation: every 1s). Since neighbor nodes may change, static unicast faces to content sources cannot be configured and communication needs to be performed via broadcast. Broadcast requests enable implicit content discovery (see Chapter 4) as illustrated in Figure 11.4, i.e., a broadcast request can address multiple nodes at the same time but only a content source, which holds the desired content, will reply. Content retrieval can also be performed via Dynamic Unicast (see Chapter 6), where content requests are transmitted via broadcast only until a content source is found. Then, subsequent Interests are addressed via unicast to the same content source until it becomes unavailable. Although content retrieval can be performed via opportunistic one-hop or multi-hop communication, we limit the scope of ACR requests in our current implementation to one-hop and leave evaluations of multi-hop for future work.

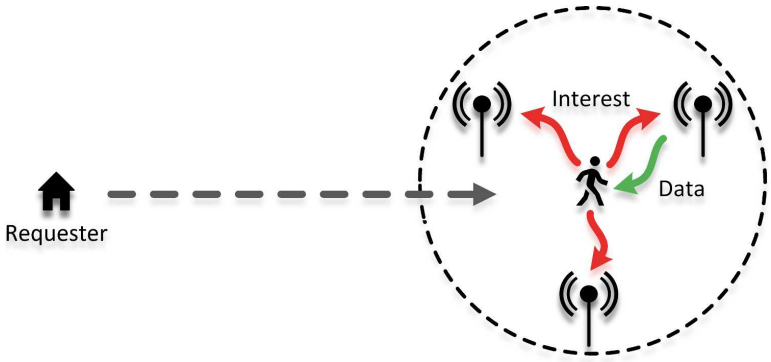


Figure 11.4: Content Retrieval by Agents: Broadcast Requests enable Implicit Content Discovery, i.e., Agents can find the Content quickly at any Neighbor Node.

Content retrieval is performed in two steps. First, the agent resolves the content version. Then, as soon as a version has been found, content retrieval can start. To persistently store the content and keep the publisher's original signatures, the agent delegates content retrieval to it's local repository, which is an independent application that retrieves and persistently stores content. Interactions between local applications and repositories (via internal faces through the CCND) follow the repository protocol [7]. Two commands can be used in Interest messages to initiate a content download via repository.

- The **Start Write (%C1.R.sw)** command asks the repository to retrieve and store a content object sequentially from segment 0 until the final segment.

- The **Checked Start Write (%C1.sw-c)** command is similar to Start Write but it requires a segment number as starting point. The repository first checks whether the segment is already available in the repository such that it does

not need to be retrieved again. Then, it starts content retrieval from the specified segment number (usually different than segment 0).

In case of intermittent connectivity and disruptions, content retrieval may not be completed at once. Then, the repository implementation re-expresses expired Interests up to five times before content retrieval is aborted. After that, content retrieval needs to be restarted manually with the Checked Start Write command. Unfortunately, the repository implementation does not have a mechanism to detect whether the complete content is stored in the repository or not. However, this information is required to resume disrupted content retrievals and retrieve missing segments. Therefore, we have extended the repository protocol by a new command such that applications can retrieve additional information from the repository.

- The **Validate Content (%C1.R.vc)** command requests information whether content retrieval has been completed (all segments stored) or whether content retrieval is still running. If the content is incomplete (and content retrieval is disrupted), repositories include the lowest missing segment number in the Data reply such that content retrievals can be resumed via Checked Start Write.

Agent applications can then send Interests in the form *<prefix>/<version>/%C1.R.vc* to repositories to request status information (complete, incomplete or disrupted at specific segment number) for ongoing content retrievals. In our current implementation, agent applications perform a Validate Content check periodically every 4 seconds (default Interest lifetime used in repositories) for ongoing content retrievals until the content is complete. When content retrieval is complete, the notification phase starts.

Note that the size of agent repositories may grow with an increasing number of delegated content retrievals. To delete content that is not used anymore, e.g., incomplete content, delivered content or content that is never retrieved by requesters, deletion mechanisms of persistent caching (see Chapter 12) may be applied.

11.3.3 Phase III: Content Notification

When an agent has retrieved the content, it can notify the requester via push or pull notifications. Then, after receiving a notification, the requester can retrieve the content from the agent. The decision which notification type to use is made by the requester during agent delegation. Both notification types are based on the assumption that agents meet requesters again after a while. However, agent delegation and content delivery can also be at different locations as long as both locations can communicate and coordinate with each other. For example, Figure 11.5 shows two access points, which are placed along a road. A requester is connected to the access points via wired link (dashed line). Then, agent delegation may be performed at one access point and content delivery may be performed at another access point further down the road, while agents can collect information in-between.

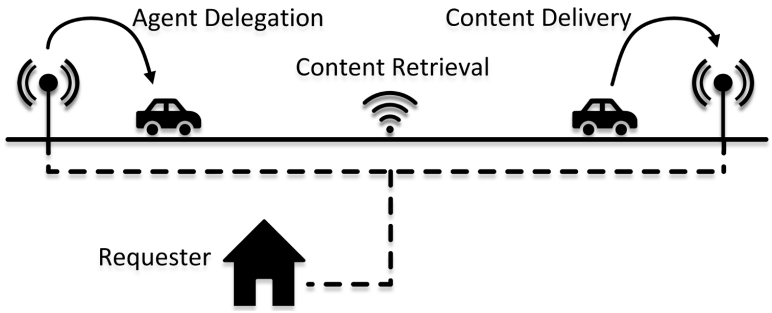


Figure 11.5: Agent Delegation and Content Delivery through Access Points at Different Locations: Requester and Access Points can coordinate each other via Wired Link (dashed line).

Alternatively, agents and requesters could exchange content with the help of home repositories [151], which are repositories running in the agents' home networks, i.e., they are continuously connected to the Internet. Then, agents can synchronize retrieved content with their home repositories (whenever they have connectivity to the Internet) such that requesters can retrieve it without meeting agents again.

Push Notification

As soon as an agent has retrieved the content, it can start the notification phase by periodically transmitting push notifications (see Figure 11.6). When the requester receives the push notification, it can start retrieving the content from the agent.

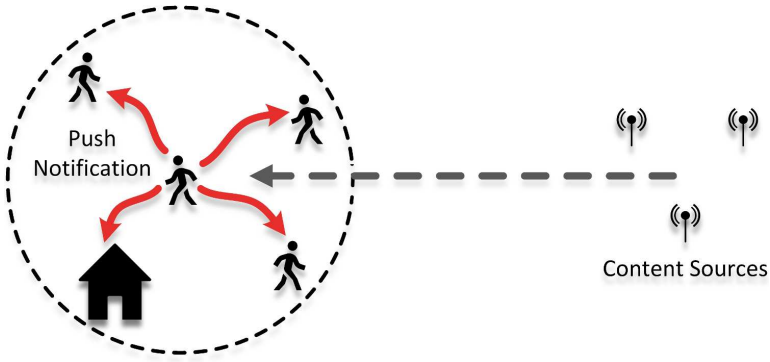
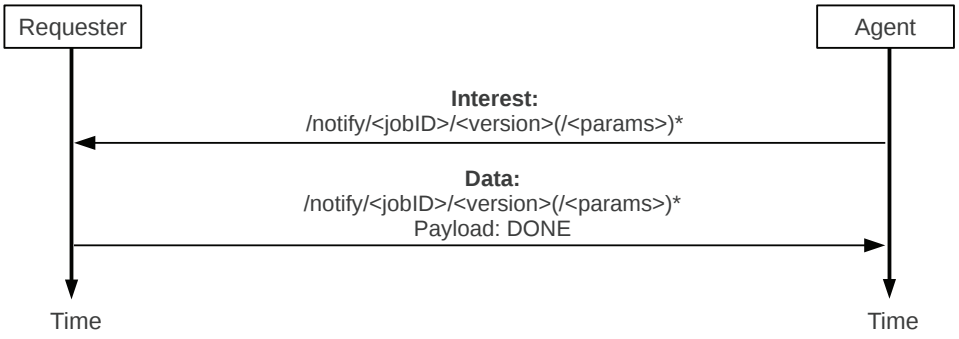
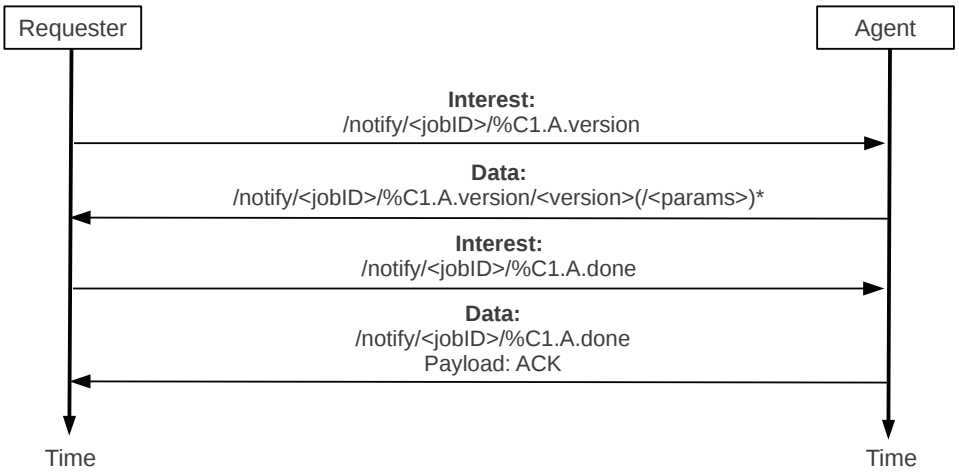


Figure 11.6: Push Notifications by Agents: After Reception, Requesters can start Content Downloads from Agents.

Figure 11.7a illustrates the message sequence for push notifications. Push notifications are Interest messages with the prefix */notify*, the *jobID* (see agent delegation) and the content version. Furthermore, agents may add additional parameters for content retrieval, e.g., their nodeIDs. In our implementation, we use IP addresses as nodeIDs but also other node identifiers, e.g., MAC addresses or descriptive names, are possible. NodeIDs are required such that requesters can create



(a) Push Notification



(b) Pull Notification

Figure 11.7: Message Sequence for Push and Pull Notification.

direct unicast faces to agents to retrieve content with a higher throughput, i.e., there are no broadcast delays and rate adaptation mechanisms are supported. Thus, in contrast to [143, 41], nodeIDs are only used to create unicast faces to neighbor nodes but they are not included into NDN messages. As soon as content retrieval has finished, the requester notifies the agent (*DONE* flag) indicating that no more notifications are required.

Pull Notification

Pull notifications are based on periodic Notification Requests transmitted by requesters followed by Notification Responses transmitted by agents if they have retrieved the content (see Figure 11.8). The message sequence is shown in Figure 11.7b. Agents that have completed content retrieval can register an Interest filter for the jobID in the CCND in order to receive Notification Requests, i.e., Interests for the jobID. Then, as soon as an agent comes into the requester's transmission range and receives a Notification Request, it can respond with a Notification Re-

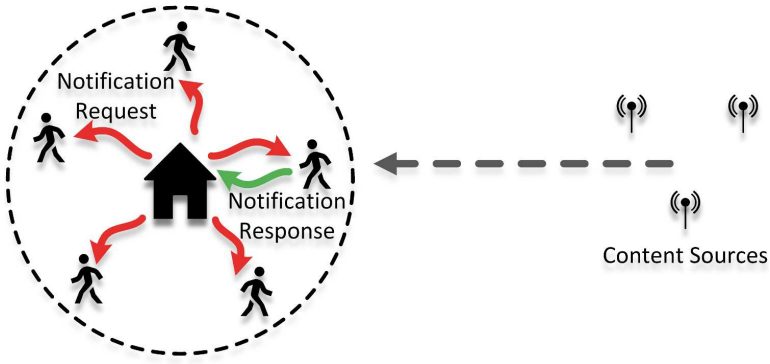


Figure 11.8: Pull Notifications by Requesters: Agents reply with Notification Responses if they have retrieved the Content Object.

sponse containing the content version and optionally the nodeID (for direct content retrieval similar to push notifications). Since multiple agents may be delegated with the same jobID for redundancy, a requester can retrieve notifications from any agent in its neighborhood with one message. After a requester has finished content retrieval from an agent, it can notify the agent (via extensible command marker [3] *%C1.A.done*) to delete the job and release the resources for other jobs.

Please note that push notifications have larger sizes than pull notifications (in our implementation push notifications are around 35 bytes larger) because they contain all information to retrieve content (e.g., nodeID, content version). Pull notifications can be short because additional information is only transmitted if requester and agent meet.

11.4 Evaluation of ACR on Smart Phones

We have implemented Agent-based Content Retrieval (ACR) in CCNx 0.7.1 and evaluated it on LG Google Nexus 4 [12] smart phones running on Android 4.2.2. This implementation included only pull notifications and was lacking the final acknowledgment in the agent delegation phase as described in Subsection 11.3.1. The evaluation topology is shown in Figure 11.9. There are four nodes: one smart phone acting as requester, two smart phones acting as agents or forwarders and one laptop acting as content source. All communication is performed via IEEE 802.11n, i.e., no cellular communication, and all nodes are within direct transmission range to each other.

For ACR, requesters delegate content retrieval to agent nodes, which retrieve the content from the content source. To avoid requesters fetching overheard content between agent and content source from their local cache, we set the AnswerOriginKind field in Interests to 0 indicating that no answer from the content store is accepted. This is only necessary in our ACR evaluations because requester and content source are within communication range. In practice, if a requester would be in communication range of the content source, it could retrieve the content dir-

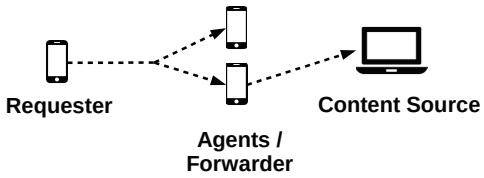


Figure 11.9: Evaluation Topology for ACR on Smart Phones.

ectly without agent. Furthermore, we assume in our scenarios that if an agent detects the content source, it has enough time to retrieve all segments subsequently without disruptions.

We also perform evaluations with *ccngetfile* [183], a standard CCNx content retrieval application, for reference purposes. To have the same conditions as with agent-based content retrieval when using ccngetfile, every Interest is transmitted via unicast on the first hop from requester to forwarder and then forwarded via broadcast on the second hop to the content source. Because the requester has only a unicast face to a forwarder, it cannot overhear broadcast communication between agent and content source. Consequently, the requester needs to retrieve content via forwarder from the content source.

11.4.1 Agent-based Content Retrieval

ACR is performed via two hops: 1) from agent to content source and 2) from requester to agent. All messages during ACR are transmitted via broadcast except for the content retrieval between requester and agent, which is unicast. The reason for this is that content sources and agents may not be known a priori. Only after the agent tells the requester its node ID (IP address), the requester can create a unicast face for direct content retrieval. In this subsection, we compare ACR with *ccngetfile* over two hops. ACR comprises all three phases including agent delegation, content retrieval and notification while ccngetfile only comprises content retrieval over two hops. In both cases, content transmission on the first hop between requester and forwarder/agent is performed via unicast and on the second hop between forwarder/agent and content source it is performed via broadcast.

Figure 11.10 shows the measured throughput of both transmissions for different content sizes. The x-axis shows the different content sizes and the y-axis shows the throughput. ACR uses a probing interval of 5s. This means that every 5 seconds the requester transmits a broadcast request to every reachable agent asking whether it has retrieved the desired content. For content transmissions of 1 MB, the throughput of ccngetfile is 20% higher than ACR and for 200 KB it is even 80% higher. Since no agent delegation and notification is required with ccngetfile, the throughput when transferring small content is higher due to lower message overhead. However, ACR results in higher throughput for content retrievals of 4 MB or larger although every segment is retrieved from the agent's repository and not from its cache as with ccngetfile. For content sizes of 10 MB and 20 MB, throughput of ACR is even 20% higher than with ccngetfile. As described above, with ccngetfile,

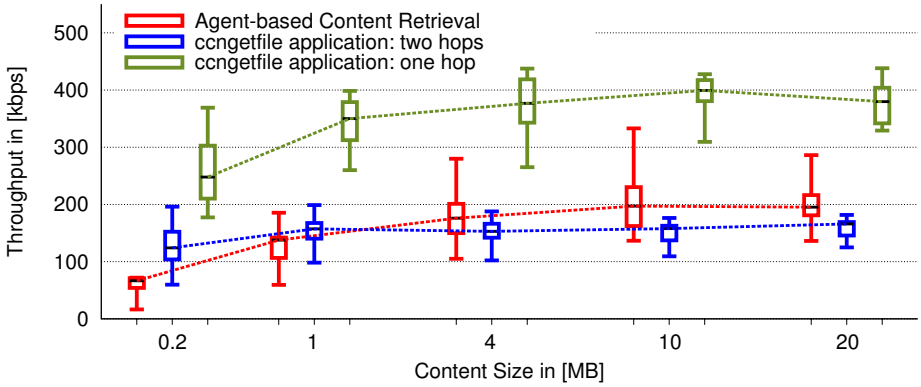


Figure 11.10: Throughput of Agent-based Content Retrieval vs. ccngetfile.

every Interest is transmitted by the requester via unicast on the first hop and then forwarded via broadcast to the content source on the second hop. Since the number of concurrently transmitted Interests is limited by the pipeline size at the requester, new Interests can only be transmitted if matching Data for previously transmitted Interests has been received. Therefore, the unicast data rate on the first hop is limited by the broadcast data rate on the second hop, which is lower than the unicast data rate. With ACR, content retrievals are performed subsequently. First, the agent retrieves the content via broadcast from any (unknown) content source and stores it in its repository. Then, the requester can retrieve the content via unicast from the agent using the full unicast data rate.

In practice, multi-hop unicast forwarding with ccngetfile is not feasible in opportunistic networks because forwarders are not known and, thus, can not be configured statically in the FIB. Only if all nodes would support Dynamic Unicast (see Chapter 9) multi-hop communication via unicast would be possible. However, communicating over many hops to reach a content source far away would reduce overall throughput. For example, Figure 11.10 shows that throughput of ccngetfile over two hops is halved compared to the throughput over one hop. Earlier studies have shown that in unicast multi-hop communication, throughput degrades with the number of n hops by $1/n$ [110] or worse [102]. In contrast to this, ACR can exploit the mobility of agent nodes to reach a content source via multiple subsequent one-hop transmissions.

11.4.2 Impact of the Probe Interval for Pull Notifications

In this subsection we evaluate pull notifications in a static scenario. A requester transmits periodic Notification Requests at different probing intervals to check whether a neighboring agent has retrieved the content (pull notifications). As soon as the requester detects the desired content on an agent, it can start downloading the content from the agent. The larger the probing interval is, the more coarse is the notification granularity. With increasing content size, the number of received Notification Requests increases, because more time is required until content retrieval

is completed and a Notification Response can be transmitted. An appropriate value for the probe interval may, therefore, also depend on the requested content size. However, it is currently not possible in CCNx to know the content size until receiving the final segment.

We explore the impact of different probing interval values on the number of transmitted Notification Requests and the content retrieval time. The content retrieval time is measured from the transmission of the first Exploration Interest at the requester until the requester has received the complete content from the agent. Therefore, the content retrieval time also depends on the time until an agent can start retrieving content from a content source. In practice, depending on the mobility and connectivity of nodes, there may be a delay until an agent meets a content source, retrieves the content and meets the requester again to reply to a Notification Request. For simplicity, we ignore this delay in this evaluation and assume that an agent can instantly reach a content source and reply to Notification Requests as soon as the download has finished.

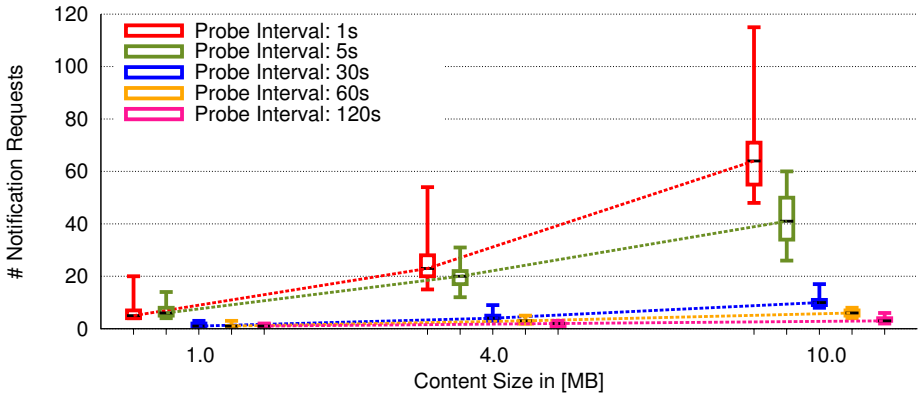


Figure 11.11: Transmitted Notification Requests for Different Probe Intervals.

	Probe Intervals		
File Sizes	**30s**	**60s**	**120s**
4 MB	-80%	-85%	-90%
10 MB	-76%	-85%	-93%

Table 11.1: Transmitted Notification Requests in relation to a Probe Interval of 5s.

Figure 11.11 shows the number of transmitted Notification Requests when using a probe interval of 1s, 5s, 30s, 60s and 120s during content retrievals of 1 MB, 4 MB and 10 MB. As expected, the shorter the probe interval is, the more Notification Requests need to be transmitted. In Table 11.1 we list the differences of transmitted Notification Requests in percent when using a probing interval of 30s, 60s or 120s instead of 5s. During the retrieval of a 10 MB content object, 76% fewer Notification Requests are required with a probe interval of 30s and 85% fewer requests with a probe interval of 60s.

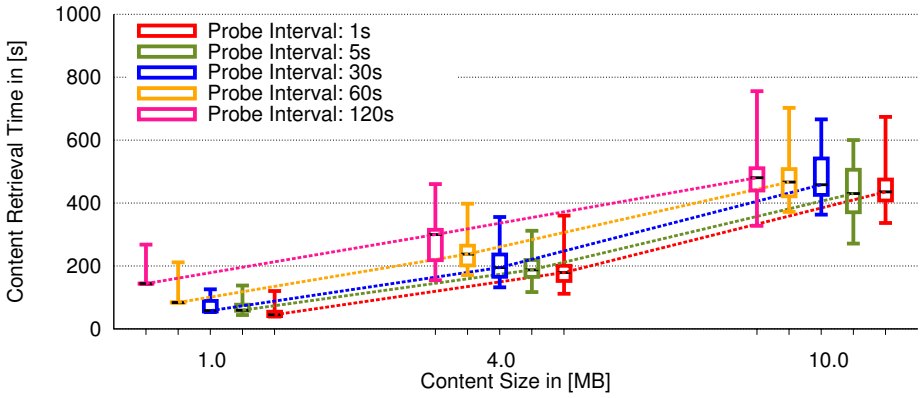


Figure 11.12: Content Retrieval Times for Different Probe Intervals.

	Probe Intervals		
File Sizes	**30s**	**60s**	**120s**
4 MB	+4%	+27%	+60%
10 MB	+6%	+9%	+12%

Table 11.2: Content Retrieval Times in relation to a Probe Interval of 5s.

Figure 11.12 shows content retrieval times when using different probe intervals. The figure shows that content retrieval times increase with increasing content size but the differences between different probe intervals decrease with increasing content size. In Table 11.2, content retrieval times are compared to a probe interval of 5s. For content retrievals of 4 MB, only 4% more time is needed with a probe interval of 30s and 27% more time with a probe interval of 60s instead of 5s. For content retrievals of 10 MB or larger, the median content retrieval times are similar because the content retrieval times are much larger than the differences in the probe interval. Therefore, in this scenario, a good tradeoff value for the probe interval seems to be 30s, because compared to a probe interval of 5s, Notification Requests decrease by 76 - 80% but content retrieval times increase only by 4 - 6%. However, probe intervals may also depend on the mobility of the nodes. For example, in case of short contact times to agents, shorter probe intervals may be beneficial.

11.4.3 Conclusions

In delay-tolerant networks, Interest forwarding over multiple hops may not be possible due to intermittent connectivity between nodes. Agent-based Content Retrieval (ACR) enables requesters to delegate content retrieval to agent nodes, which can find the content on behalf of requesters. Since available agents and content sources are not known a priori, communication needs to be performed via broadcast. If a requester detects an agent that has retrieved the complete content, it can

retrieve the content from the agent via unicast. We have implemented and evaluated ACR on Android smart phones. Evaluations have shown that the overhead for agent delegation and notification is only measurable for very small content objects. For content objects larger than 4 MB, ACR may even result in 20% higher throughput than multi-hop forwarding, although the content is stored at intermediate nodes on secondary storage and not in the cache. Because the maximum number of concurrently transmitted Interests is limited by the pipeline size, content retrieval times during multi-hop forwarding are limited by the slowest link. With ACR, content is transmitted subsequently over both hops and, thus, every link can reach its maximum capacity.

While ad hoc networking is supported on wireless mesh nodes, it is still an unresolved issue on Android devices [1]. Although Android would be capable of supporting ad-hoc networks, there is no API from the kernel enabling developers to setup and configure ad-hoc communication. Thus, to enable ad-hoc networking, Android devices need to be rooted. However, a requirement to root devices before using them would be a major disadvantage preventing widespread protocol usage because users may lose warranty when rooting their devices. To enable information-centric device-to-device communication among unrooted Android devices, workarounds are required. Wifi-Direct [26] enables direct file exchange between devices but it requires manual discovery and pairing procedures. Wifi-Opp [194] uses the tethering mode of smart phones enabling some mobile nodes to opportunistically turn into access points. Other devices can scan the environment for usable access points and associate with those for some time to exchange messages. Unfortunately, when using Wifi-Opp, nearby users may miss communication opportunities among themselves if they are associated to different access points. However, the upcoming LTE Direct [21], which is part of Release 12 of the 3GPP standard, features energy efficient proximity-based service and content discovery based on application-specific expressions. Thus, LTE Direct may enable requesters to find suitable agents or enable agents to notify requesters in an efficient way. Then, as soon as a contact is detected, content retrieval could be performed via ad hoc communication (if supported), Wifi-Direct [26] or Wifi-Opp [194].

11.5 Evaluation of ACR via Emulation

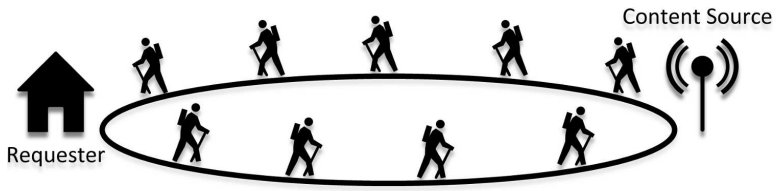
We have implemented agent-based content retrieval (ACR) and Dynamic Unicast (DU, see Chapter 9) in CCNx 0.8.2 [27] and compared it to multi-hop broadcast communication (using unmodified CCNx as reference since dynamically created unicast paths may have only a limited lifetime in mobile scenarios). The evaluation has been performed with NS3-DCE [16] on a Linux cluster [200]. By that, we deploy the same source code on simulated nodes that would run on real mobile devices. Although this evaluation method introduces limitations in terms of network size and simulation time, we believe that it increases accuracy and practical relevance of our evaluations.

11.5.1 Scenarios and Configuration

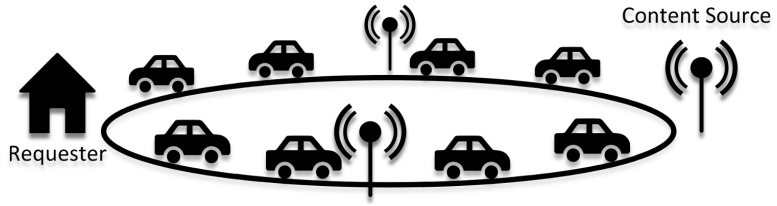
The evaluation parameters are listed in Table 11.3. Every node has an IEEE 802.11g wireless interface and we use a log distance propagation loss model. With the selected parameters, the transmission range is approximately 130m (outdoor scenario). The data rate is adapted automatically based on the distance, i.e., the signal-to-noise ratio (SNR).

Parameter	**Value**
Wireless Standard	IEEE 802.11g, 2.4 GHz
Modulation	ERP-OFDM, min. data rate: 6 Mbps max. data rate: 54 Mbps
Propagation Loss Model	Log Distance with Exponent: 3.0 Reference loss: 40.0 dB
Energy Detection Threshold	-86.0 dBm
CCA Model Threshold	-90.0 dBm
Mobility	circular mobility regular: 1.0 - 1.4m/s, node pause: 0-30s slow: 0.7 - 1.0m/s, node pause: 0-1200s fast: 10.0 - 14.0m/s, node pause: 0-30s
Circle Radius	250m (circumference: 1570.8m) 375m (circumference: 2356.19m) 500m (circumference: 3141.59m)
Nodes	1 static requester, 1-3 static sources mobile nodes: 5-100 agent limit: 1, 5, 10 agent delegation: every 10s up to limit
File Sizes	0.5 MB, 1 MB, 5 MB, 10 MB, 20 MB segment size: 4096 bytes

Table 11.3: Evaluation Parameters for ACR in Mobile Scenarios.



(a) Hiker Scenario



(b) Vehicular Scenario

Figure 11.13: Evaluation Topologies for ACR in Mobile Scenarios.

Human mobility follows cycles among different points of interest, e.g., work and home, by traveling on urban roads or using public transportation systems. In our evaluations, we consider circular mobility, where (NDN capable) nodes move with different velocities on a circle. Figure 11.13 shows the evaluation topologies for a hiker and a vehicular scenario. Mobile nodes are randomly distributed on the circle and move with velocities within a specified range depending on the scenarios described below. Node pauses specify individual occasional breaks (randomly selected within the specified intervals in Table 11.3), where nodes do not move. Although mobile nodes return to the requester in our evaluations, this is not necessarily required as described in Subsection 11.3.3. For ACR, every mobile node is an agent. Requesters can delegate content retrieval to mobile nodes (agents), at best effort every 10s until the agent limit, i.e., the maximum number of delegated agents, has been reached. If there is no agent available due to low node density, agent delegation is postponed by 5 seconds.

Since end-to-end paths (for multi-hop communication) can be disrupted, we use a content retrieval application (see Chapter 5) that persistently stores received segments at requesters. Then, even in case of long disruptions (when cached content may be deleted), content downloads can always be resumed by requesters from where they were stopped. Different from agents, requesters do not need to provide persistently stored content to others but can store it privately (not in the repository). Still, received Data messages are stored temporarily in caches of intermediate nodes enabling quick retransmissions.

Every configuration has been evaluated in 100 different simulation runs.

Hiker scenario

In the hiker scenario (Figure 11.13a), we place a static requester and a static content source on opposite sides of the circle to ensure multi-hop communication. Consider a round trip hiking trail to the top of a mountain. At the top, there is a content source, e.g., a solar powered sensor node that is gathering data (e.g., weather data, web cam snapshot etc.). At the start of the trail there is the tourist office, which is interested in the sensor data. Since there is no direct connection between tourist office and sensor, multi-hop routing or ACR needs to be applied. Hikers travel with regular pedestrian speeds of 1.0 - 1.4m/s on the trail and make a short break from time to time to enjoy the view or take a picture, i.e., node pause times of 0-30s. In addition, there are lazy hikers, who travel with slower speeds of 0.7 - 1.0m/s and make longer breaks between 0s and 1200s.

Vehicular Scenario

The vehicular scenario (Figure 11.13b) is similar to the hiker scenario but nodes move with vehicular speeds of 10.0 - 14.0m/s resulting in significantly shorter contact times between nodes. Therefore, two additional redundant content sources are placed on the circle, e.g., access points that are connected to the Internet, such that requester and content sources are in equidistance (1/4 circumference) to each other. Then, if content retrieval from one content source is disrupted, it can be resumed from another content source. Consider for example travelers on a safari in a wildlife park. While moving in their cars, travelers may collect sensor information from their surroundings (e.g., images from animal surveillance cameras). From time to time, travelers make short stops to watch animals more closely and they can deliver collected information at the exit when leaving the park. Other mobility examples may include public transportation systems or mail delivery services in urban areas, where users follow a specific route and then return back.

11.5.2 Push vs. Pull Notifications

We first evaluate the notification types for ACR with one-hop broadcast in the hiker scenario. Figure 11.14 shows the notification messages (y-axis) transmitted via push and pull notifications when all nodes move with regular speeds (1.0 - 1.4m/s) on a circle with radius 250m. We evaluate the performance for different numbers of nodes (x-axis) in the network resulting in different node densities.

Figure 11.14 shows that the number of pull notifications stays approximately constant independent of the number of delegated agents (agent limit) because one pull request can retrieve content from any agent node in the vicinity. However, more push notifications are required if the number of delegated agents increases because each agent transmits them individually. Furthermore, the number of push notifications increases with more mobile nodes (agents) in the network because agents can be delegated faster, i.e., agents can be delegated shortly after each other such that they start their notification phases approximately at the same time. If

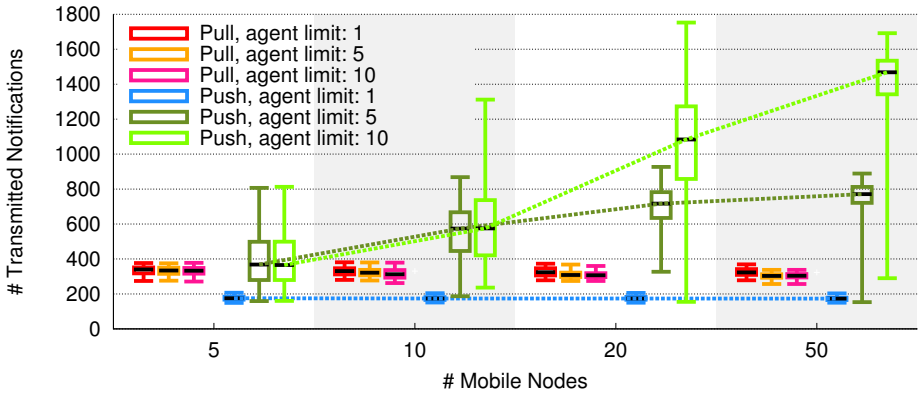


Figure 11.14: Number of Push and Pull Notification Messages for a Varying Number of Mobile Nodes.

fewer mobile nodes are available, agent delegation takes more time, i.e., some agents may have already returned the content to the requester before other agents have retrieved the content and started their notification phases.

For 10 delegated agents out of 50 mobile nodes, push notifications result on average in 3.8 times more notification messages than pull notifications. However, if there is only one delegated agent, pull notifications result in 87% more messages than push notifications. This is because we start pull notification requests immediately after delegating the first agent (no assumptions when content is retrieved), while push notifications start only after the agent has retrieved the content. Thus, pull notifications are transmitted for a longer time. However, optimizations for pull notifications are possible by estimating the time for agents to retrieve content and return back. As a rule of thumb, we can say that pull notifications should be used if content retrieval is delegated to more than one agent (fewer notifications) and push notifications are more efficient for delegations to one agent (since notifications start only after the agent has retrieved the content).

Please recall that pull notifications are smaller because additional information is only transmitted if requester and agent meet while push notifications contain all information to retrieve content (see Subsection 11.3.3). When considering the sizes of transmitted messages for one agent, pull notifications result in only 25% (or 5.5 KB) more traffic (not optimized case). Thus, if notification messages need to be transmitted periodically, pull notifications are favorable in terms of message size compared to push notifications. In the remainder of this chapter, we only use pull notifications.

11.5.3 Agent-based vs. Multi-hop Content Retrieval

We compare ACR with one-hop broadcast requests against multi-hop communication (broadcast and DU with SFF or PFF), on a circular topology with a radius of 250m. Figure 11.15 shows content retrieval times for a 1 MB file (y-axis) for dif-

ferent numbers of nodes in the network (x-axis). In this scenario, 75% of the nodes move with slow speeds and make long breaks and only 25% of the nodes move with regular speeds (hiker scenario, cf. Table 11.3). The horizontal area between the dotted lines denotes the min./max. traveling times of an agent around the circle (regular speed, no pause time).

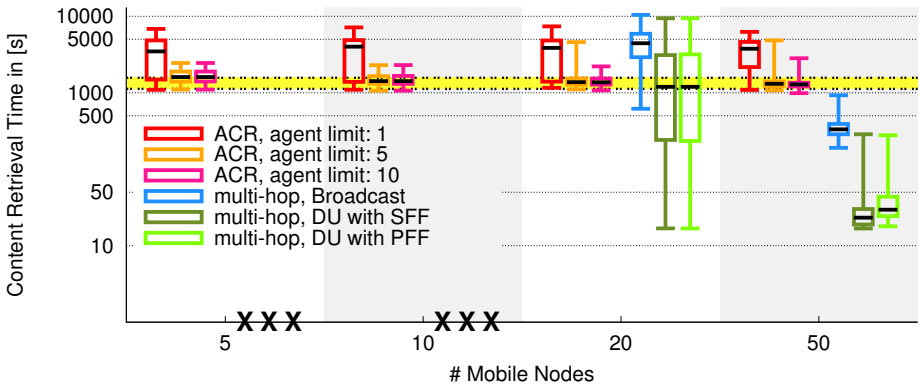


Figure 11.15: Content Retrieval Times of a Requester on a Circle Topology with 250m Radius, 75% Slow Nodes and 25% Regular Nodes.

Figure 11.15 illustrates that ACR retrieval times mostly depend on the nodes' mobility characteristics while multi-hop communication depends on the number of nodes (density) in the network. If content retrieval is delegated to only one agent, there is a high risk that it is a slow node (since most nodes are slow in this scenario) resulting in long content retrieval times. However, by delegating content retrieval to multiple redundant agents, i.e., in this scenario 5 agents are enough, the impact of slow nodes becomes negligible. Furthermore, we can observe that ACR is successful with any number of nodes while multi-hop communication is only possible for 20 and 50 nodes.

Scenarios with 20 nodes correspond to an average distance of 78.5m between nodes, which we define as *intermediate node density*, and scenarios with 50 nodes correspond to an average distance of 31.4m between nodes, which we define as *high node density*. Although communication with both node densities is expected to work well (transmission range of 130m), this is not the case as Figure 11.15 shows. For 20 nodes, ACR with 5 agents results in 59% (median) shorter content retrieval times than multi-hop broadcast and requires only 13% (median) more time than DU with SFF or PFF. However, multi-hop communication experiences a large variability for this node density. If nodes are favorably clustered between requester and content source (such that multi-hop communication is possible), multi-hop communication performs better than ACR while it performs worse if this is not the case, e.g., for long disruption periods. For 50 nodes, the node density is high enough such that multi-hop communication is always faster than ACR. In this case, multi-hop broadcast results in 2.9 times faster transmission than ACR

(with 5 agents) and multi-hop DU in even 44 times (PFF) or 55 times (SFF) faster transmission than ACR.

Furthermore, we can observe that multi-hop DU (with SFF or PFF) performs significantly better than multi-hop broadcast. This is due to two main reasons. First, since broadcast requests are addressed to all nodes of a node's vicinity, broadcast transmissions need to be delayed to enable duplicate suppression. In multi-hop communication, these delays have a significant impact on throughput because they are added at every hop. In our evaluations, we used the default broadcast forwarding delay (CCNx data pause) of 10ms, and larger values resulted in significantly worse performance. Only broadcast communication requires these delays since unicast requests address nodes directly. Second, the data rate for unicast transmissions can be adapted dynamically based on the distance between two nodes, i.e., based on the signal-to-noise ratio (SNR). For broadcast communication, however, the data rate can not be adapted and is usually set to the lowest supported rate (see Table 11.3). Although only a few nodes receive content via unicast, multi-hop DU can still exploit caching, i.e., if a multi-hop path breaks, Interests do not need to be retransmitted over the entire path (in most of the cases) since Data can be retrieved from a mobile node's cache, i.e., where the path broke.

Figure 11.15 shows that ACR can be combined with multi-hop DU, e.g., a requester could initially try to retrieve content via multi-hop communication and only switch to ACR if nothing can be received. In very sparse or very dense environments, the combination may be straightforward. For example, for 5 or 10 nodes in Figure 11.15, multi-hop DU does not work and a requester could switch to ACR after a few expired content requests. Similarly, for 50 nodes a requester could directly retrieve the content via multi-hop DU such that ACR is not required. However, in dynamic and time-varying environments, i.e., neither permanently dense nor sparse, the combination becomes more complex and requires further investigation. In particular, it needs to be explored how long a requester should try to find an alternative path in case of a disruption before content retrieval is delegated to an agent.

11.5.4 Agent-based vs. Multi-hop Content Retrieval for Higher Node Densities

In this subsection, we compare ACR with one-hop broadcast requests to multi-hop communication for increasing path lengths with intermediate and high node densities. All nodes are moving with regular speeds of 1.0 - 1.4m/s on a circle (hiker scenario). The path length indicates the minimum number of required hops between requester and content source, i.e., 7 hops for a motion radius of 250m, 10 hops for a motion radius of 375m and 13 hops for a motion radius of 500m. To keep the same node densities, we increase the number of mobile nodes for increasing motion radii as listed in Table 11.4.

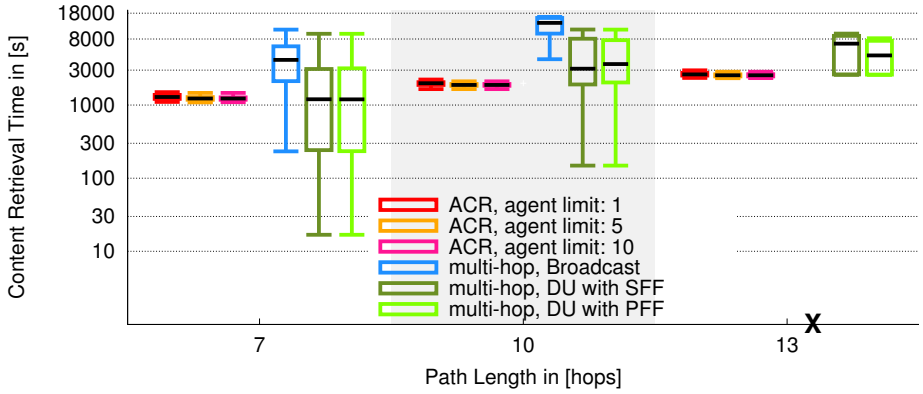
	Motion Radii		
Density	**250m**	**375m**	**500m**
intermediate	20	30	40
high	50	75	100

Table 11.4: Number of Mobile Nodes to maintain (intermediate or high) Node Densities for increasing Motion Radii.

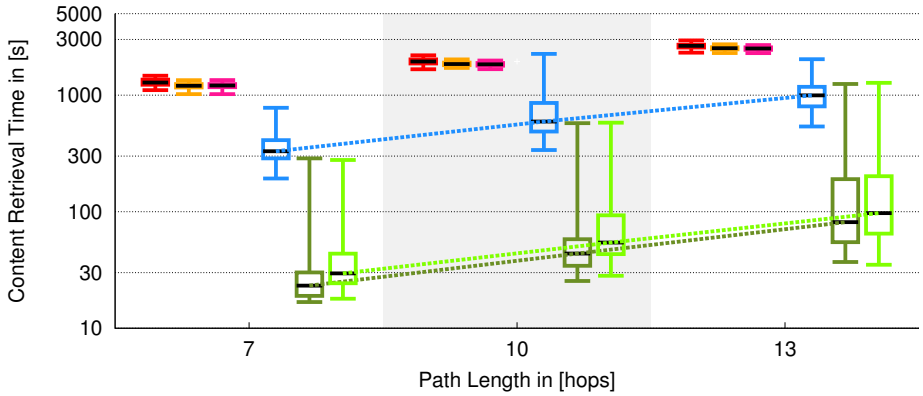
Content Retrieval Times

Figure 11.16a shows content retrieval times for a 1 MB content object (y-axis) over varying path lengths (x-axis) in case of an intermediate node density. For 7 hops, ACR performs better than multi-hop broadcast and similar to multi-hop DU (median values) as seen in the previous subsection. However, with increasing path length, multi-hop communication becomes significantly worse. For 13 hops, multi-hop content retrieval via broadcast can not be completed within 6 hours (our maximum simulation time), i.e., in 77% of the simulation runs requesters retrieve up to 11% of the content, while in 23% of the runs they do not retrieve a single segment. The performance is only slightly better with multi-hop DU because continuous end-to-end paths are only available for a short time, i.e., only in 5% of the runs requesters could retrieve the complete content. The PFF strategy results in slightly shorter content retrieval times than SFF for long path lengths due to better path redundancy. However, for intermediate node densities, multi-hop communication with more than 7 hops is barely successful or requires a lot of time. Thus, ACR should be preferred for long path lengths in such scenarios.

Figure 11.16b shows content retrieval times for the high node density. Even for long path lengths, multi-hop communication is always successful and significantly faster than ACR. This is mainly due to the fact that end-to-end path availability is more stable and agents need to travel the entire path before delivering content. We can confirm path stability by observing the number of path breaks and resume operations. For 13 hops with high node density (Figure 11.16b), approximately 5 resume operations were necessary on average with broadcast, while for the intermediate node density (Figure 11.16a) already 25 resume operations were required for 7 hops and more than 100 resume operations for 10 hops (a 1 MB file has 251 segments). Strategies to combine ACR with multi-hop DU should, therefore, not only depend on mobility patterns, e.g., node velocity and density, but also on the number of disruptions in relation to the size of already received partial files.



(a) Intermediate Node Density

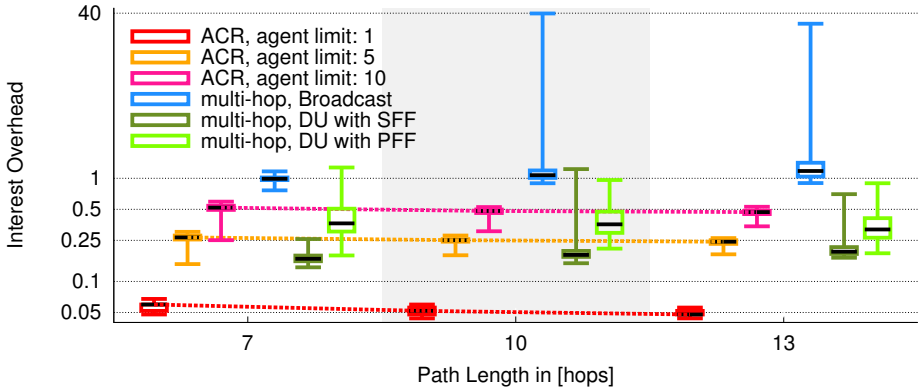


(b) High Node Density

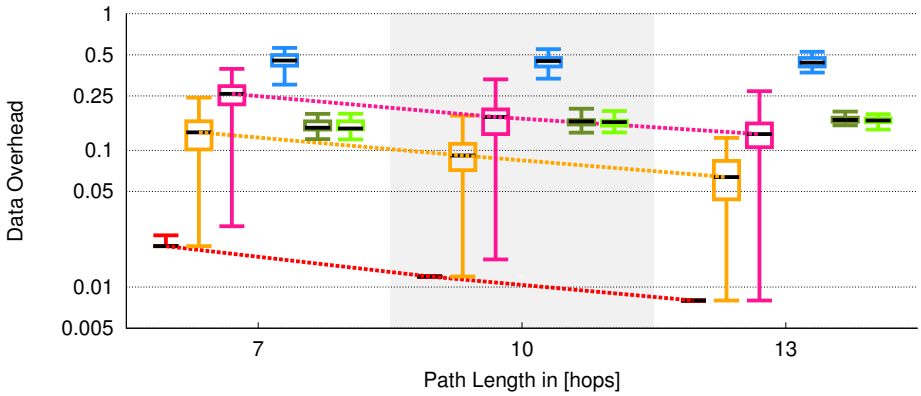
Figure 11.16: Content Retrieval Times of a Requester for Varying Path Lengths and Different Node Densities.

Transmitted Messages

Multi-hop DU establishes a path between requester and content source such that only nodes on the path receive and forward messages, while for multi-hop broadcast all nodes receive messages and decide individually whether they forward them or not. In contrast, for ACR only delegated agents retrieve content and deliver it to requesters. To compare all three schemes, we evaluate the Interest and Data overhead as defined in Section 3.3. We have evaluated the message overhead separately for content sources, requesters and mobile forwarder nodes (agents). Every configuration has been evaluated in 100 different runs and the boxplots show the message overhead of all simulation runs. Since multi-hop content retrieval does not always complete for intermediate node densities within the simulation time of 6 hours, we only show results for high node densities in the remainder of this chapter.



(a) Interest Overhead



(b) Data Overhead

Figure 11.17: Interest and Data Overhead of Mobile Nodes for a High Node Density and Varying Path Lengths.

Figure 11.17a shows the Interest overhead (y-axis) for different path lengths (x-axis). ACR with only 1 delegated agent results in the lowest Interest overhead because only 1 agent needs to send Interests to probe for the content while for multi-hop communication, Interests are forwarded by multiple nodes. With an increasing number of delegated agents, the number of transmitted Interests increases accordingly because agents probe for content independently. However, even for 10 delegated agents, mobile nodes send on average 50% fewer Interests than with multi-hop broadcast.

Multi-hop DU results in significantly fewer Interest transmissions than multi-hop broadcast because Interests are only transmitted on established paths and not flooded. Similarly, DU with SFF results in fewer Interest transmissions than DU with PFF because Interests are only forwarded over a single path and not multiple paths. Up to 4 agent delegations, ACR results in fewer Interest transmissions than

multi-hop DU with SFF, but ACR requires more Interests than multi-hop DU with SFF for 5 agent delegations or more.

While Interest messages are rather small (around 50 bytes), Data messages have a bigger impact on network traffic because they are significantly larger (around 4500 bytes). Figure 11.17b illustrates the Data overhead (y-axis) for different path lengths (x-axis). For ACR with 1 delegated agent, the Data overhead is negligible because only 1 agent needs to deliver the content to the requester. The Data overhead of multi-hop broadcast is high (mobile nodes transmit on average 50% of all Data messages), but the Data overhead can be reduced by a factor of 3 when using multi-hop DU (independent of the strategy) instead of multi-hop broadcast.

The Data overhead of ACR with 5 and 10 delegated agents is rather high in this scenario due to the high node density. Because agents can be delegated quickly after each other, they arrive at the content source approximately at the same time. Broadcast requests from some agents may, therefore, be satisfied by the content source or other agents, which have requested the content already. This illustrates the importance for efficient agent delegation to a minimum number of agents depending on environmental conditions and application requirements. In fact, there is a tradeoff between content retrieval time, i.e., how fast content can be retrieved in an arbitrary environment, and redundant message transmissions. This tradeoff is inherent to any existing DTN routing approach [139] such as Epidemic Routing [203] or Spray-and-Wait [186]. However, even for 5 delegated agents, the Data overhead of ACR is similar or lower (for an increasing path length) than for DU with SFF. The Data overhead decreases with ACR for an increasing path length since more nodes are required to maintain a high node density, i.e., only delegated agents transmit messages, while the Data overhead stays rather constant (or increases slightly due to retransmissions) with multi-hop DU.

11.5.5 Agent-based vs. Multi-hop Content Retrieval for Multiple Content Sources and Varying Content Sizes

Contact times between nodes decrease for faster node velocities. In this section, we evaluate the vehicular scenario with three redundant content sources and mobile nodes that move with velocities of 10.0 - 14.0m/s on a circle. We only consider high node densities and increase the number of mobile nodes with increasing motion radii as shown in Table 11.16. Due to short contact times to content sources, we evaluate ACR with one-hop DU requests (in contrast to ACR with one-hop broadcast requests in previous subsections) and compare it to multi-hop communication. If an agent cannot complete the content retrieval from one content source, it can resume it from the next content source.

Content Retrieval Times

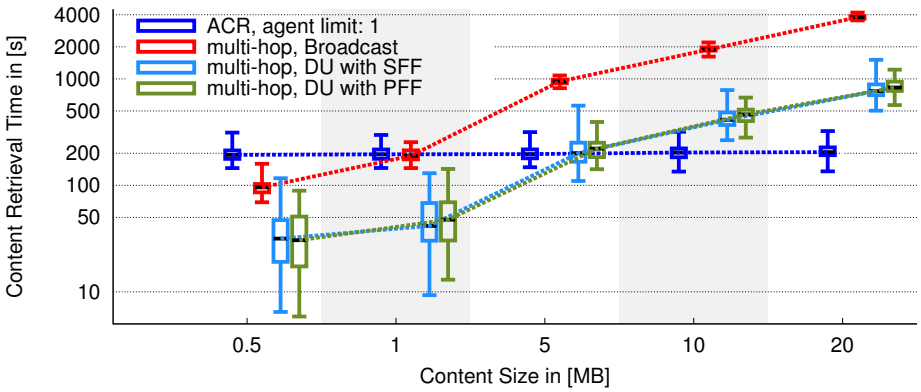
Figure 11.18a shows content retrieval times (y-axis) for different content sizes (x-axis) and a motion radius of 250m, i.e., at least 4 hops from the requester to the

next content source. The content retrieval time with ACR is rather constant independent of the content size as long as it can be retrieved during the short contact time between requester and agent. Although contact times between nodes are short, i.e., a requester sees the same next hop for only 9 - 13 seconds (and the complete path is valid for a much shorter time), multi-hop DU performs significantly better than multi-hop broadcast. ACR is faster than multi-hop DU for larger content objects, i.e., 5 MB and more, while multi-hop DU results in shorter content retrieval times for smaller content objects (1 MB and less). Evaluations with slightly larger motion radii, e.g., 375m (5 hops) in Figure 11.18b, look similar, but multi-hop DU requires 2 - 4 times longer for 5 hops compared to 4 hops while multi-hop broadcast requires only 38% more time for 5 hops compared to 4 hops. For large motion radii, e.g., 500m (7 hops) in Figure 11.18c, multi-hop DU performs similar to multi-hop broadcast because paths expire quickly, i.e., only 8 - 10 messages are transmitted on average before a path expires. Consequently, DU with PFF performs slightly better than DU with SFF (lower maximum values due to path redundancy). However, in such scenarios, ACR performs generally better than multi-hop communication (except for very small content objects below 1 MB).

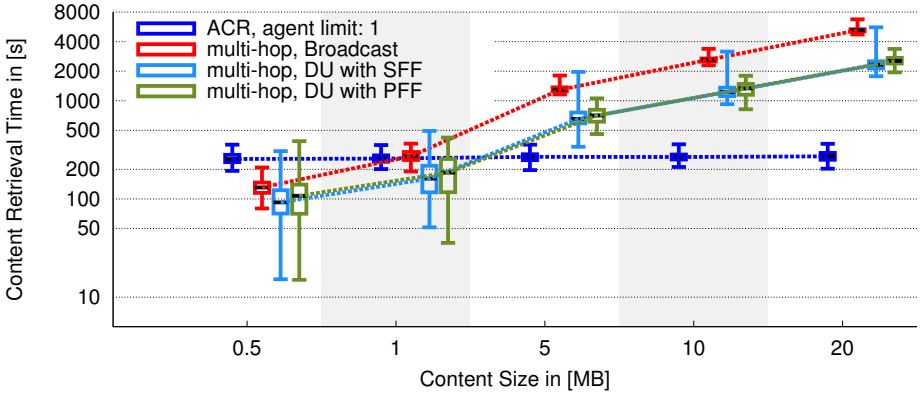
Transmitted Messages

Figure 11.19a shows the average Data overhead of all three content sources for a motion radius of 250m. While ACR results in (almost) perfect efficiency, i.e., a content source has a Data overhead of only 33%, multi-hop broadcast communication is very inefficient because every content source sends more than 90% of all Data messages. With multi-hop DU almost perfect efficiency can be obtained, i.e., only 2 - 3% more Data messages are transmitted by content sources compared to ACR. Consequently, the number of Data messages forwarded by mobile nodes in the network is also significantly lower with multi-hop DU compared to multi-hop broadcast.

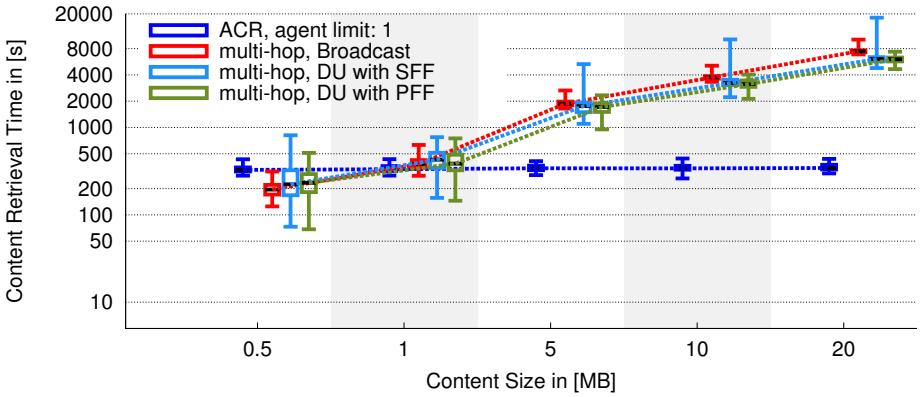
Figures 11.19b and 11.19c show the same evaluations for larger motion radii of 375m and 500m. Compared to a motion radius of 250m, broadcast Data transmissions by content sources decrease by 17% (375m) and by 24% (500m) because of longer path lengths. Hence, the probability that some paths break and not all Interests are forwarded to all content sources increases. However, for DU with SFF and PFF the situation is different. Path breaks result in more Interest transmissions via broadcast, which address all content sources (since broadcast is the fallback strategy). For DU with SFF and PFF, Data transmissions by content sources increase by 6% for a motion radius of 375m instead of 250m and even by 17% (SFF) or 12% (PFF) for a motion radius of 500m instead of 250m. For long paths, PFF requires slightly fewer Data transmissions than SFF due to a higher path redundancy (fewer fallbacks to broadcast). Furthermore, although content retrieval with multi-hop DU requires approximately the same time than multi-hop broadcast (cf. Figure 11.18c), Figure 11.19c shows that multi-hop DU still results in fewer Data transmissions than multi-hop broadcast. This indicates that multi-hop DU experi-



(a) Motion Radius of 250m

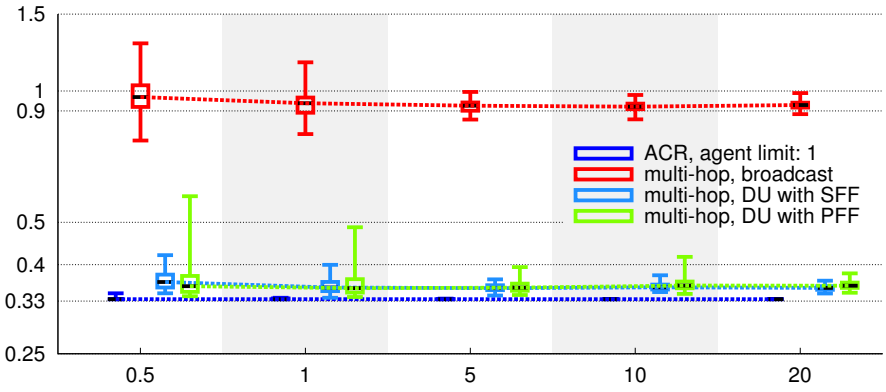


(b) Motion Radius of 375m

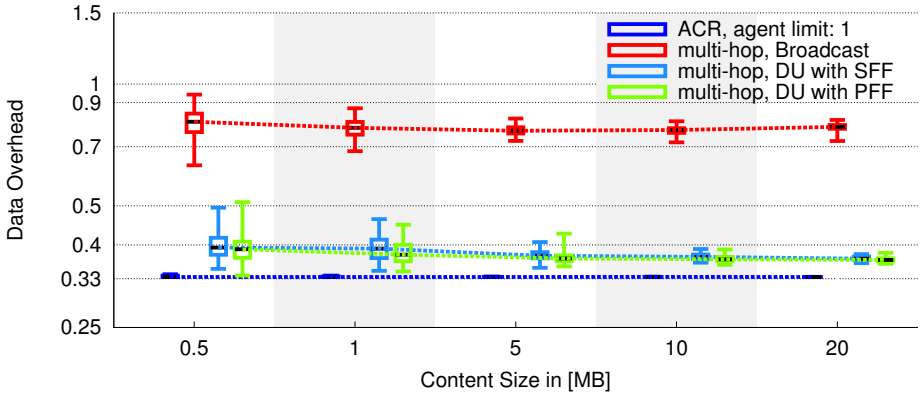


(c) Motion Radius of 500m

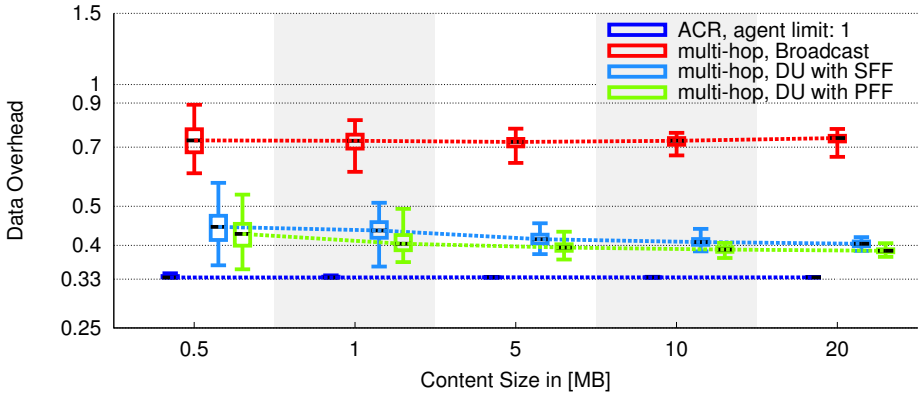
Figure 11.18: Content Retrieval Times of a Requester for Different Content Sizes and Motion Radii.



(a) Motion Radius of 250m



(b) Motion Radius of 375m



(c) Motion Radius of 500m

Figure 11.19: Data Overhead of Content Sources for Different Content Sizes and Motion Radii.

ences many timeout periods, where Interests are not forwarded due to path breaks and can not be retransmitted until they expire (the default Interest lifetime is 4 seconds). Thus, to quickly react to path breaks and increase throughput of multi-hop DU, adaptive Interest lifetimes (see Chapter 10) based on measured round-trip times (RTTs) may be beneficial.

For ACR, the path length has no significant impact on the number of Data transmissions by content sources but the contact time to content sources does. Therefore, we have also evaluated the overhead of ACR with one-hop DU when retrieving large files from multiple content sources, i.e., resume operations. However, the overhead is negligible, e.g., for a download of 60 MB (15'000 segments) at a mobile speed of 14m/s from 3 content sources only 5 - 8 more segments need to be transmitted by content sources (because the agent is out of range for reception) and agents send 10 - 15 more Interest messages via unicast (because disruptions can not be detected instantaneously).

11.6 Lessons Learned

In this section, we give an overview of our lessons learned and discuss open issues.

11.6.1 Push vs. Pull Notification

After an agent has completed content retrieval, it needs to notify the requester. We have observed that pull notifications are more efficient for multiple delegated agents since one message can address multiple agents at the same time. In addition, pull notifications are rather small, i.e., they request additional information only when a suitable agent is in range, such that they result in less network traffic than push notifications. On the downside, a requester needs to transmit pull notifications without knowing whether an agent has already retrieved the content, while push notifications are only transmitted by agents after finding the content. Thus, for pull notifications, a requester needs to estimate when to start requesting notifications (or just start after delegating the first agent).

11.6.2 Agent Selection and Delegation

When delegating agents, there is a tradeoff between content retrieval time and message overhead. Thus, it is important to keep the number of delegated agents at the lowest possible level that still enables reasonably fast content retrieval (depending on application requirements). In our current implementation, all agents are delegated in short time intervals to quickly retrieve content even in the presence of many slow nodes. An alternative (more conservative) strategy could be to delegate content retrieval to more agents only after a much larger time interval, e.g., based on estimations when an agent might return (past experiences).

Furthermore, the quality of agent selection can be improved with additional information. Currently, we select agents in a requester's neighborhood randomly.

Other options for more efficient agent selection may be based on social criteria, e.g., social interactions [145] or reputation [233], past experiences, e.g., overheard content [43] or past GPS traces [109], or based on hybrid approaches [113, 185], where potential agents are discovered from a central server (when connected to the Internet) for later usage when disconnected from the Internet. Alternatively, LTE Direct [21] or Wi-Fi Aware Networking [35] may also be promising technologies to enable agent discovery or content notification in an energy-efficient way if requesters and agents are in proximity.

11.6.3 Impact of Path Length

In dense environments, multi-hop DU results in faster content retrieval times than ACR. Yet, if only a few agents are delegated, ACR has a lower message overhead because only selected agents need to transmit messages while for multi-hop communication every node on the path forwards messages. For ACR, the message overhead decreases with increasing path length while it stays constant (or slightly increases due to retransmissions) for multi-hop DU.

In general, multi-hop DU performs better than broadcast communication for long path lengths because no broadcast delays (for duplicate suppression) are required. Hence, breaking of symmetric Interest-Data forwarding paths is no issue because Data can be returned quickly (within milliseconds), i.e, the topology has not changed much.

11.6.4 Impact of Node Velocity

DU with SFF is more efficient than DU with PFF for pedestrian mobility with respect to content retrieval times and message transmissions. If nodes move with vehicular velocities, neighboring nodes may see each other only for a short time. Thus, for long path lengths, multi-hop DU results in similar content retrieval times than multi-hop broadcast due to frequent path breaks (timeouts and fallbacks to broadcast). In particular, DU with PFF performs slightly better than DU with SFF for long paths due to higher path redundancy.

Yet, ACR performs generally better than multi-hop DU with either forwarding strategy in high speed scenarios. To retrieve content in case of fast node velocities, it is crucial to detect available content sources quickly. In our implementation, agents periodically probed for content (Interest probing) at a fixed rate of one Interest per second. However, the agent application has full control when to transmit Interests and can adapt Interest transmission based on arbitrary requirements such as node velocity (e.g., fewer Interests at lower speed), location or past experience to minimize the Interest overhead depending on scenario and application requirements. In general, more frequent Interest transmissions may result in a higher message overhead and, thus, in a higher energy consumption but may also detect a content source more quickly (tradeoff between message overhead and content retrieval time).

To increase throughput during short contact times, ACR can be combined with one-hop DU, which addresses Interests (after an initial broadcast) at a higher rate to the same content source until it becomes unavailable. We have seen that the overhead of one-hop DU for resume operations from multiple content sources is negligible.

11.6.5 Combination of ACR and DU

ACR and multi-hop DU perform differently under similar network conditions. In general, multi-hop DU performs better for content retrieval in dense environments and for small content objects while ACR performs better in sparse environments, for fast node velocities and for large content objects. Therefore, ACR and multi-hop DU can complement each other perfectly. In dense or sparse environments, the combination may be straightforward, i.e., a requester can try to retrieve content via multi-hop and delegate it to an agent only if multi-hop content retrieval is not successful. However, in intermediate (neither dense nor sparse) environments, the combination is more complex and requires further investigations. In particular, it needs to be explored how quickly a requester should delegate content retrieval to an agent after a disruption. If it is delegated too early and connectivity would be re-gained quickly, there may be redundant message transmissions. However, if a requester waits a long time before delegating content retrieval to an agent, the content retrieval time increases accordingly.

In general, there may be different scenarios and application requirements, which can influence the decision whether to use multi-hop routing (e.g., DU) or delay-tolerant communication (e.g., ACR). For example, quick content retrieval times may be most important for some applications but message efficiency (which translates to energy consumption) may be more important for other applications. Hence, depending on application requirements, delay-tolerant communication may enable energy-efficient communication [204] without degrading network performance.

11.6.6 Security

Delegating content retrieval to other nodes introduces various attack options, which have not been analyzed in this work. For example, an agent should not retrieve malicious or illegal content for other users. To mitigate this threat, trust and reputation models may be established. Then, agents and requesters could exchange their identities and sign messages during agent delegations.

Furthermore, the retrieved content should not be too large and fill the agent's complete memory. Therefore, requesters and agents can negotiate a maximum content size during agent delegation. If the content is larger than an agent can handle (e.g., due to impending resource exhaustion resulting from hardware limitations or due to an already high task load), a requester needs to delegate content retrieval to multiple agents, which request different parts of the content. An approach to do this may be based on RC-NDN (see Chapter 7), where content sources use Rap-

tor codes to create Data encodings. Without requiring coordination, agents can retrieve a certain number of Data encodings and deliver it to the requester, where original content can be recovered after decoding.

11.7 Conclusions

We have described agent-based content retrieval (ACR) and showed that delay-tolerance in information-centric networks (ICN) can be supported without modifications to ICN message processing. This enables seamless operation in well-connected and disruptive networks. Furthermore, we have shown that mobile ICN communication does not require all messages to be transmitted via broadcast. Dynamic Unicast (DU) has resulted in faster transmission times than broadcast for slow and high node velocities (up to a certain path length). Symmetric Interest-Data forwarding paths have not been identified as limitation because Data messages are returned within milliseconds, i..e, the topology has not changed much.

We have seen that node mobility is not necessarily a disadvantage for wireless communication and ICN provides the means to exploit it. While multi-hop communication is faster with high node densities, ACR is superior in low and intermediate node densities where multi-hop communication does not work or results in frequent disruptions. Furthermore, ACR is beneficial for large content sizes and works well even under high mobility, where it can be combined with one-hop DU. Although, in our scenarios, agents had to return to the requester to deliver content, agent delegation and content delivery can also be performed at different locations as long as both locations can communicate and coordinate with each other.

With our approach, multi-hop DU and ACR can be combined. A requester could initially try to retrieve content via multi-hop communication and only delegate retrieval to agents if nothing has been found. Because all messages are stored in the same ICN message format, requesters could also retrieve (via multiple hops) content from agents, which were delegated by other requesters. However, concrete mechanisms to combine multi-hop DU with ACR under dynamic (time varying) network conditions are still subject for more investigations.

Chapter 12

Persistent Caching

12.1 Introduction

In NDN, every received Data message is stored in the content store before it is further forwarded. The basic idea of the content store is to keep received Data as long as possible in buffers to satisfy similar requests (retransmissions or requests from other nodes). However, since content stores need to operate at line-speed, current memory technologies impose limitations. Fast memory is expensive, power hungry and only available in small capacities [156]. Furthermore, content stores are implemented in volatile storage, which is cleared, i.e., data loss, in case of power outages.

Delay-tolerant networks are characterized by disruptions and the lack of instantaneous end-to-end paths. In such scenarios, short-term volatile caching may not be enough and content needs to be stored persistently (at the expense of slightly slower access times). In Chapter 5, we have presented a content retrieval application that enables resume operations in case of long disruptions from content sources. Requesters that run the retrieval application store requested partial content persistently without providing it to others. Yet, to enable persistent caching, content needs to be made available publicly with original publisher signatures such that requesters can verify its authenticity. In Chapter 11, we have described agent-based content retrieval, where agents store retrieved Data messages in repositories such that requesters can retrieve them at a later time enabling delay-tolerant communication. Furthermore, to increase communication opportunities in (sparse) DTNs, agent nodes could also be statically deployed in wireless networks such that mobile users, which meet agent nodes at different times, could use them as relay for delay-tolerant Data exchange among themselves similar to DTN throwboxes [232].

However, with increasing number of delegated content retrievals, an agent's repository size may grow with time. Since storage space is limited, it is crucial to delete non-essential, incomplete or outdated content to free space for new content. In this chapter, we present a persistent caching concept that maintains popular content in repositories [47, 46, 95] and deletes unpopular content if memory space is required. Our work is orthogonal to existing ICN research on caching (see Subsec-

tion 2.8), because it can be combined with fast (short-term) caching. Hence, our persistent caching concept is not limited to delay-tolerant networking via agents but can also be used for custodian-based information sharing [118] or network caches that enable high content availability and performance similar to content distribution networks (CDNs) by dynamically storing content in regions of high demand. While received and forwarded content will automatically be stored in the content store (short-term cache), persistent caching can be used to store only a subset of it for a longer time. For example, real-time audio streams from phone conferences may be stored in the cache, but it may not be required to keep them for a long time. In contrast, large static files, such as multimedia files or pictures may be valid for a longer time and can be cached at persistent storage closer to requesters.

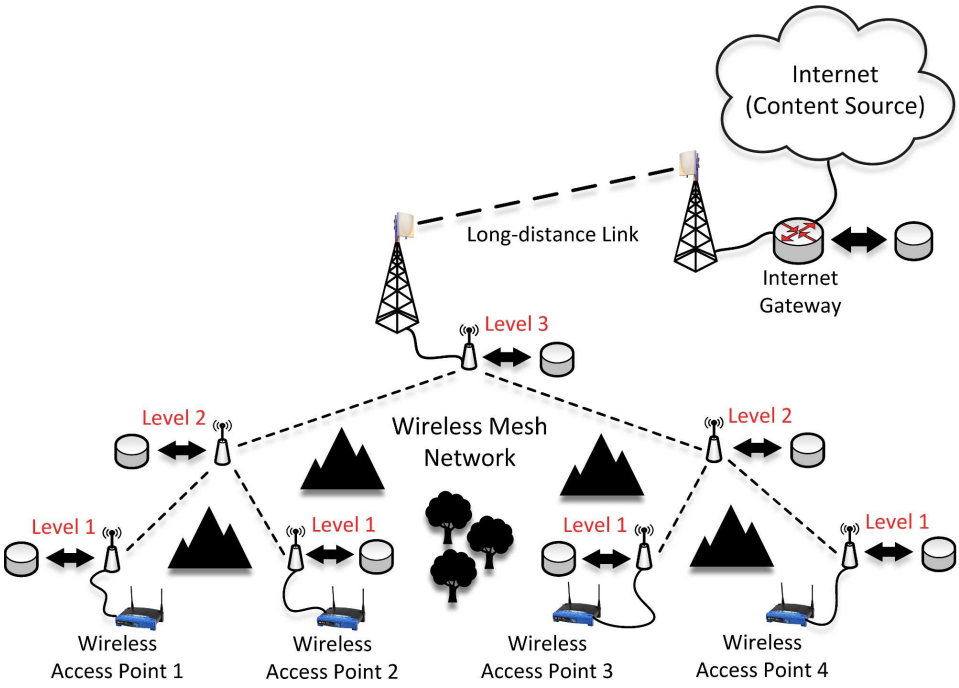


Figure 12.1: Hierarchical Caching in a Wireless Mesh Network.

Consider a wireless mesh network deployed in a mountain area that provides Internet access to users in remote valleys as illustrated in Figure 12.1. User requests are forwarded in a hierarchical way from wireless access points to wireless mesh nodes (at level 1, 2 and 3) and via directional antennas over a long-distance link to an Internet gateway. The Internet gateway may be connected with multiple wireless mesh networks and provides connectivity to content sources in the Internet. Persistent caching may be deployed alongside wireless mesh nodes and Internet gateways to store a subset of content forwarded through these nodes. Several studies of social networking services have shown that network traffic characteristics have significant homophily and locality components (see Subsection 2.8). This means that people geographically close to each other have similar interests and trends to

access content. Thus, by storing very popular content of the day, e.g., electronic newspapers or popular videos, at persistent caches at the edge of a network (e.g., at level 1), the network performance can be improved. While many requests for popular content may be satisfied already at edge routers, requests for less popular content may be forwarded further to the next wireless mesh node, which may hold a cached copy of the content. Thus, only unpopular content, for which caching would not yield any benefits, would need to be retrieved all the way from the content source in the Internet. Hence, persistent caching may increase the capacity of a wireless network since only new (non-redundant) content needs to be transmitted over potential bottlenecks such as the long-distance link in Figure 12.1.

12.2 Design and Implementation of Persistent Caching

Persistent storage in NDN is provided by repositories. In CCNx 0.8.2, repositories store all content in the *repofile* and maintain references to the content in a B-tree. Content sources publish content in repositories [184] to make it available to other nodes. It is also possible to synchronize collections among repositories with a synchronization protocol [9]. However, there is no way of deleting content from a repository besides resetting the entire repository, which results in the deletion of all stored content. Hence, to use repositories for caching, content deletion needs to be introduced in an automatic way, e.g., based on popularity. Yet, we do not maintain popularity counters for two reasons. First, popularity counters would need aging mechanisms, introducing significant additional complexity [164]. For example, content that has been requested extensively one year ago may be less popular in the near future than content that has been frequently requested in the last hours, although the absolute number of requests would be lower. Aging timers would require last access timestamps in case multiple objects have expired (to know what to delete first). We prefer simplicity over complex solutions to minimize the processing overhead in content routers. Second, content requests that reach the repository (persistent cache) would not reflect the effective number of requests due to regular caching in the content store. In case of multiple concurrent requests, only the first request would be forwarded to the persistent cache while the others may be satisfied from the content store until the content is replaced.

Therefore, in this work, we maintain access information and delete content that has not been requested recently. There are two main differences to LRU strategies. First, deletion operations are performed based on content and not individual chunk popularities. Second, multiple content objects may be deleted at the same time to free space if a certain storage utilization threshold is reached because deletions in the filesystem take more time than in main memory (ms vs. sec).

12.2.1 Data Structures

Figure 12.2 illustrates data structures required to enable content deletion for persistent storage. The *delete_queue* maintains a *queue_element* for every content object in the *repofile*. The basic idea is to move requested content to the tail (bottom) of the queue such that unpopular content can be found at the head (top) of the queue. Therefore, if content needs to be deleted, it can be removed from the head.

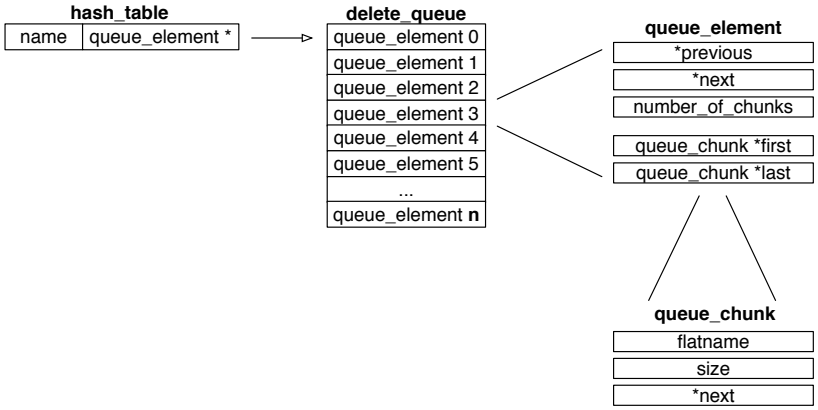


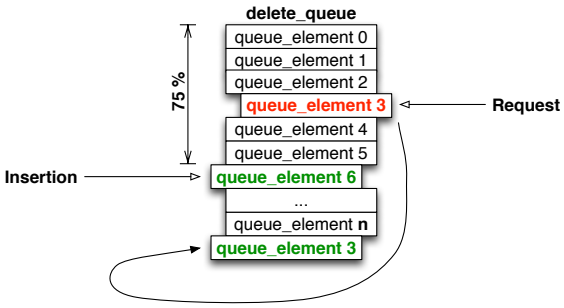
Figure 12.2: Additional Data Structures for Persistent Storage.

Figure 12.2 shows that the *delete_queue* is implemented as doubly linked list, on which every *queue_element* has a pointer to the previous and next *queue_element*. In addition, every *queue_element* has a pointer to another linked list of *queue_chunks*, i.e., the individual chunks of the content. Besides a pointer to the next element, we also maintain a pointer to the last chunk in the list to avoid long list traversals when including chunks of large content. The *queue_chunk* contains the flat name of a chunk to find the content (and its reference in the *repofile*) in the B-tree. When we need to find a *queue_element* quickly, we use a *hash table* to get its reference in the *delete_queue* based on a lookup of the base name, i.e., content name without chunk numbers.

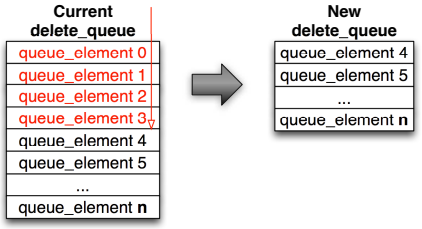
In contrast to related work on NDN caching, we keep content based on object granularity and do not make individual caching decisions for every chunk. Because content in NDN is requested sequentially based on the pipeline size, high variability in chunk downloads would degrade overall download performance. More information on chunk-based vs. object-based caching can be found in Section 12.4.

12.2.2 Processing

In this subsection, we describe processing procedures in the *delete_queue*. In particular, this comprises 1) Inclusion of New Content, 2) Queue Updates and 3) Deletion Operations.



(a) Insertion and Update



(b) Deletion

Figure 12.3: Delete Queue Processing.

Inclusion of New Content

Content information is stored based on object granularity. When a chunk is received, the content name, i.e., base name without chunk number, can be extracted. Based on a hash table lookup, *delete_queue* entries of existing content can be found quickly. In this case, only a new *queue_chunk* needs to be added to the existing *queue_element*. If it is new content, a new *queue_element* is included into the *delete_queue*. As Figure 12.3a shows, we include new *queue_elements* in the middle of the lower half, i.e., at 75% of the queue. If content would be included in the upper 50% of the delete queue, new popular content could be deleted almost instantly, e.g., if the inclusion is just before a content deletion, since up to 50% of the repofile is deleted during a deletion operation (see below). In addition, new content is not appended to the tail such that unpopular content can quickly reach the head of the *delete_queue*, while popular content can go to the tail.

Queue Update

Figure 12.3a illustrates also update operations on the *delete_queue*. Every time content is requested, the corresponding element in the *delete_queue* is pushed to the tail of the queue. This push operation can be performed for every requested chunk, every n-th chunk or only the first chunk. If every (or every n-th) chunk would be processed, there would be a tendency of larger files at the tail of the queue, since they have more chunks and, thus, more pushing operations. Therefore, we decided

to consider only the first chunk of a content object as trigger for pushing operations (see Section 12.4 for more information).

Deletion Operation

A deletion operation is initiated if the *repofile* has reached a certain size, i.e., the *repofile threshold*. Then, a deletion operation is performed by deleting a configurable percentage of the *repofile*, i.e., the *deletion ratio*. A deletion operation is initiated after a file inclusion, if the *repofile threshold* has been exceeded. In CCNx, file sizes are only known when the final chunk has been received with the *final bit* set. Therefore, the *repofile threshold* is a soft threshold and the *repofile* size can become slightly larger than the threshold depending on the size of included files, i.e., we do not perform deletion operations during file inclusions but rather afterwards.

Figure 12.3b shows modifications on the *delete_queue* due to deletion operations. A deletion operation is performed by the following steps.

1. Prevent the repository daemon from accepting new content while the deletion operation is being performed. If new content arrives during local deletion operations, it will only be stored temporarily in the content store. However, if persistent caching is used as network cache, other repositories on the path to the requester will store it persistently.

2. Start at the head of the *delete_queue* and iterate through the elements until the *deletion ratio* is reached. All content entries up to this point will be deleted (red part in Figure 12.3b) and the lower part becomes the new *delete_queue*.

3. Every *queue_element* contains multiple *queue_chunks*. The *queue_chunks* of all deleted content objects need to be sorted based on their position in the *repofile* such that the repofile can be sequentially processed (see next step) and every B-tree entry needs to be processed only once. In our current implementation, we use the $\mathcal{O}(n \log n)$ merge-sort algorithm for sorting.

4. All content from the *repofile* (except deleted chunks) are copied to a new *repofile*. This is required because file systems do not support selective deletions inside files. Due to deletions, content is copied to other positions in the new *repofile*, thus, reference values in the B-tree need to be updated.

5. All B-tree entries of deleted content are removed.

6. Instruct the repository daemon to start accepting new content again.

To limit service interruptions of network caches due to deletions, a (read-only) repository can be started to provide content from the old *repofile*. Otherwise, Interests may just be forwarded and satisfied by persistent caching at the next router level. Thus, only in the worst case Interests would be forwarded all the way to the content source.

12.3 Evaluation

Persistent caching has been implemented by extending the repository implementation in CCNx 0.8.2[1], and extensive evaluations have been performed in different scenarios on physical servers of a Linux cluster [200]. Evaluations have been performed on Intel Xeon E5-2665 and Intel Xeon E5-2650-v2 processors, i.e., each evaluation ran on a single core with 25 GB of RAM and enough disk space to store the repofile.

12.3.1 Scenarios

Figure 12.4 shows our evaluation topology. We evaluate the performance of persistent caching at an edge router, e.g., at an access point or Internet gateway, that continuously receives requests from the network according to YouTube and web server traffic models. The edge router is connected to a local repository, which is responsible for persistent caching. Independent of the network topology, an edge router has a *downstream face* from which file requests are received and content is returned and an *upstream face* where received Interests are forwarded and new content is received, i.e., file inclusions at the persistent cache of the edge router. The evaluation parameters are listed in Table 12.1.

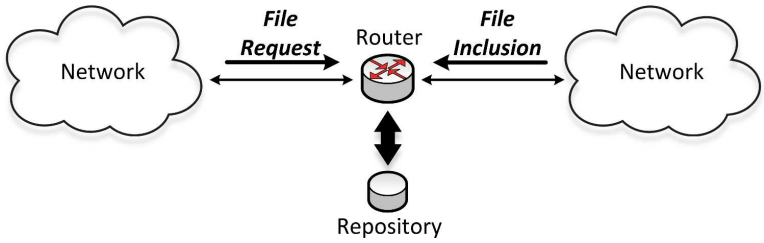


Figure 12.4: Network Scenario for Persistent Caching.

Similar to existing ICN literature [170], we assume that content popularity follows a Zipf distribution. We use 20 popularity classes and perform evaluations with parameters α set to 1 and 2. A parameter of $\alpha = 1$ is considered realistic for web server traffic and $\alpha = 2$ is used for YouTube traffic [170]. Several studies have shown [70, 229, 196] that most files are unpopular and only a few files are very popular. Therefore, we map the number of files in all popularity classes to a Zipf distribution $\alpha = 1$ with inverted classes, i.e., most files are included in class 19 and fewest files in class 0.

The file sizes in each popularity class vary as well. Based on existing YouTube models [28], we set the file size distribution for our YouTube scenario to a gamma distribution with $\alpha = 1.8$ and $\beta = 5500$. Our YouTube files are between 500 KB and 100 MB, while most files are between 2 and 10 MB (9.9 MB mean).

[1] Although CCNx 1.0 has been made available, our caching concept is still valid as the CCNx 1.0 Tutorial [141] confirms.

Parameter	YouTube	Web server
Requests	every 5s	
Request Popularity	Zipf distribution with $\alpha = 2$ $\alpha = 1$	
File distribution per popularity class	Zipf distribution, $\alpha = 1$ mapped to inverse classes	
New Files	every 10s	
File sizes per popularity class	Gamma distribution, $\alpha = 1.8$, $\beta = 5500$ min. 500 KB max. 100 MB	Gamma distribution, $\alpha = 1.8$, $\beta = 1200$ min. 50 KB max. 50 MB
Repofile thresholds	2 GB, 4 GB, 8 GB	8 GB, 12 GB, 16 GB
Deletion ratios	50%, 25%	
Effective duration	86400s (1 day)	

Table 12.1: Evaluation Parameters for Persistent Caching.

The file sizes for web server traffic are considerably smaller [222]. File sizes have increased in the last years and we believe that file sizes will increase even more in future information-centric networks. Transmitted ICN packets need to have a certain minimum size to be efficient, e.g., chunk size of 4096 bytes or more, to avoid too large overhead for content headers including names and signatures. Therefore, we believe that for future ICN traffic, many small files may be aggregated to larger data packets or ICN would only be applied to large static files, e.g., pictures or embedded videos, and not small text files that may change frequently. Therefore, we use a gamma distribution with $\alpha = 1.8$ and $\beta = 1200$ for the web server scenario. Our web server files are between 50 KB and 50 MB, however, most files are between 750 KB and 1250 KB (2 MB mean).

In our scenarios, requests are performed periodically, i.e., 1 new content request every 5 seconds. The requested content of a popularity class (Zipf distribution) is selected randomly among all created content objects in that popularity class. To simulate a dynamically growing file catalog and to evaluate the performance of persistent caching with regular deletion operations, we create and request new files every 10s. These files are included into the repository, i.e., file inclusions, as mentioned above.

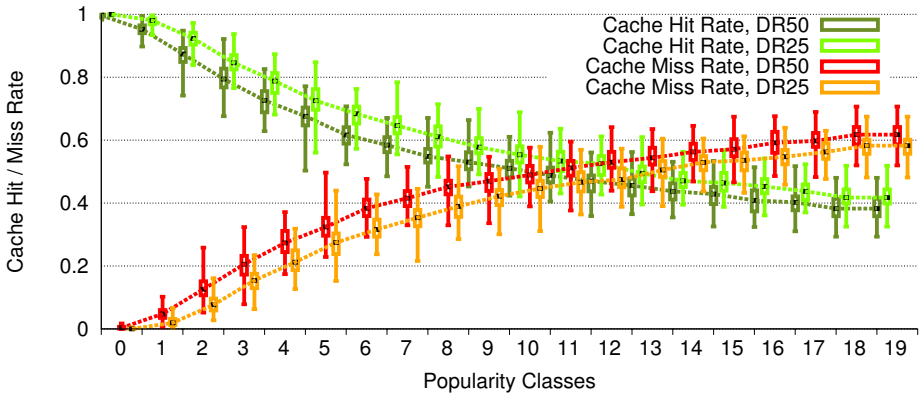
For every scenario, we evaluate various *repofile thresholds* and *deletion ratios* of 50% (DR50) and 25% (DR25) of the *repofile*. We measure the performance of persistent caching during operation, i.e., the repository is filled initially with content and we collect statistics after a first deletion operation has been performed. The effective evaluation starts after the first deletion and lasts 86400 seconds (1 day). Thus, in one day we create approximately 85.54 GB of data in the YouTube scenario and 18.67 GB of data in the web server scenario. Every configuration has been evaluated and repeated 100 times on physical servers [200] that run a NDN router with persistent caching.

12.3.2 Cache Hit and Miss Rates

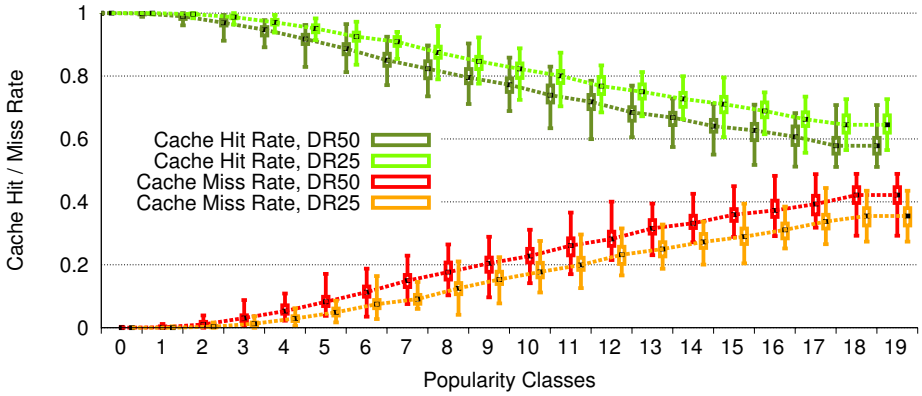
In this subsection, we evaluate the cache hit rates of our repository implementation in the YouTube and web server scenario. Figures 12.5 and 12.6 show the cache hit and miss rates for all popularity classes in different configurations. The y-axis shows the cache hit/miss rates and the x-axis indicates the popularity class. Figure 12.5 is obtained for our web server scenario, i.e., requests with Zipf distribution $\alpha = 1$, and Figure 12.6 shows the YouTube scenario with Zipf distribution $\alpha = 2$.

Figure 12.5a shows the cache hit and miss rates in the web server scenario with a repofile threshold of 2 GB. The dark green boxplots show the cache hit rates for a deletion ratio of 50% (DR50) and the light green boxplots for a deletion ratio of 25% (DR25). The overall cache hit rate of DR25 is slightly higher, i.e., 81%, compared to 77.5% with DR50. For high popularity classes, such as classes 0 and 1, the difference between DR50 and DR25 is smaller because files from these classes are barely deleted in both cases. However, for classes 3 - 17, the difference between DR25 and DR50 is larger by up to 6.3% because these files are kept more likely with DR25, while they are deleted with DR50. The red boxplots show the cache miss rates for DR50 and the orange boxplots for DR25. For DR50, cache hit rates are higher than miss rates up to files from popularity class 11, while for DR25, cache hit rates are better for more file classes, i.e., up to class 13. Even for the most unpopular content in class 19, DR50 results in a slightly higher cache miss rate of 61.5% compared to 58.1% with DR25. Therefore, freeing space too aggressively, i.e., DR50, has a noticeable impact on cache hit rates in the web server scenario.

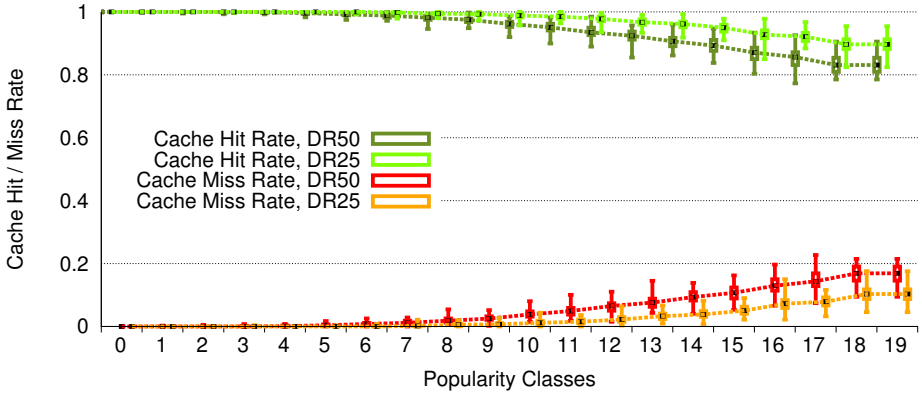
Figure 12.6a shows the cache hit and miss rates for our YouTube scenario with a repofile threshold of 8 GB. Because file sizes are larger compared to the web server scenario, we use larger repofile thresholds for YouTube scenarios than for web server scenarios. With DR50, the overall cache hit rate is 95.3% and with DR25 it is 96.9%. The relative differences between DR50 and DR25 are smaller compared to the web server scenario. This is due to the fact that the probability for requests in most popular files in classes 0 and 1 are larger for a Zipf distribution with $\alpha = 2$ (YouTube) than $\alpha = 1$ (web server), i.e., 62.7% and 15.7% instead of 27.8% and 13.9%. Therefore, more than 78% of all requests in the YouTube scenarios request content from popularity classes 0 and 1. Since our implementation keeps the most popular files, there is no difference for DR50 and DR25 for most of the requests. However, for class numbers larger than 2, DR25 results in 4.9% to 12% higher cache hit rates than DR50. We notice a large variability in performance for popularity class numbers larger than 3 in Figure 12.6a. The variability is much larger than for the web server scenario in Figure 12.5a. This can be explained by two reasons. First, the request frequency of class numbers larger than 3 is higher with Zipf distribution $\alpha = 1$ compared to $\alpha = 2$ due to larger request probabilities. Second, the file ranges that we selected in the YouTube scenario are larger than for the web server scenario resulting in higher variability. When considering the average values, DR25 results in higher cache hit rates than miss rates up to popularity class 17, while for DR50 cache miss rates become higher already



(a) Repofile Threshold of 2 GB

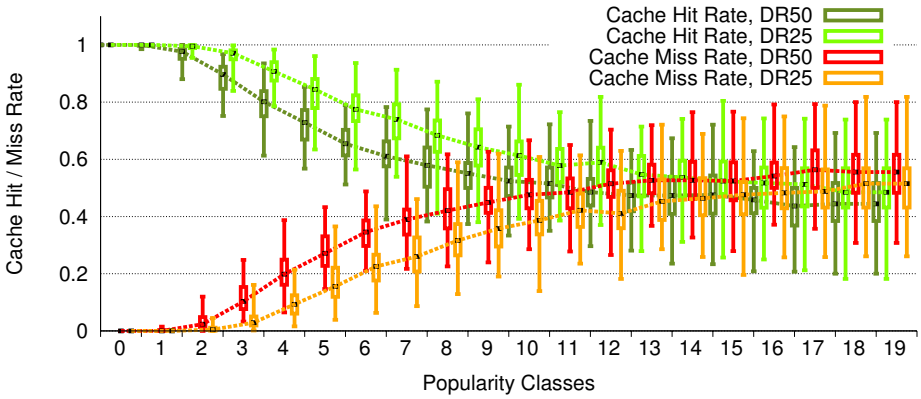


(b) Repofile Threshold of 4 GB

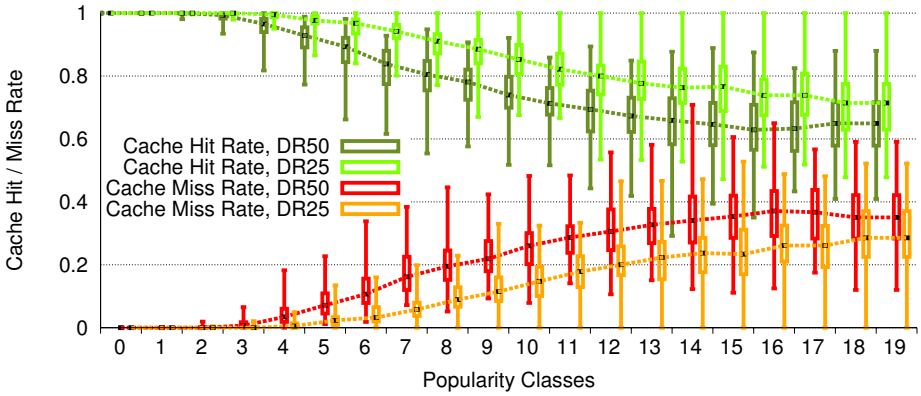


(c) Repofile Threshold of 8 GB

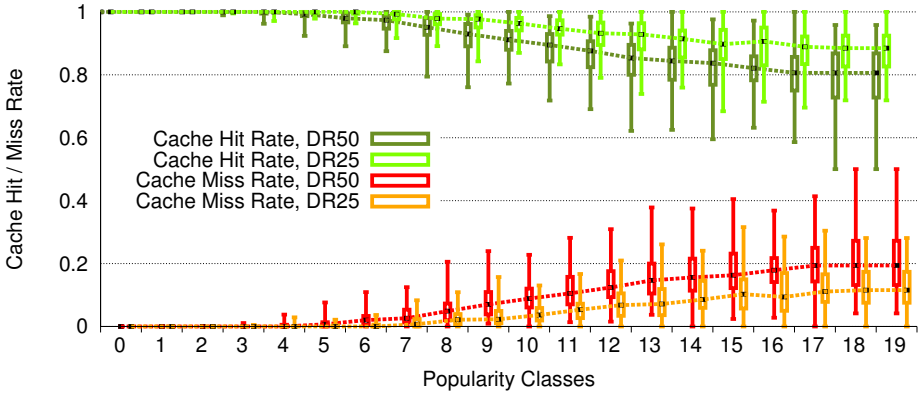
Figure 12.5: Hit and Miss Rates for Web Server Scenarios.



(a) Repofile Threshold of 8 GB



(b) Repofile Threshold of 12 GB



(c) Repofile Threshold of 16 GB

Figure 12.6: Hit and Miss Rates for YouTube Scenarios.

at popularity class 11.

Figure 12.5b and Figure 12.6b show the cache hit and miss rates for the web server scenario with a repofile threshold of 4 GB and the YouTube scenario with a repofile threshold of 12 GB. Similarly as above, DR25 results in superior performance compared to DR50. In the web server scenario, DR25 results in an overall cache hit ratio of 93%, while for DR50 it is 90.7%. In the YouTube scenario, DR25 results in an overall cache hit rate of 99% and with DR50 it still reaches 98.1%. Although these values are much higher than in the web server scenario, cache miss rates for popularity class numbers larger than 12 may become larger than cache hit rates in the YouTube scenario (worst case) due to a large variability.

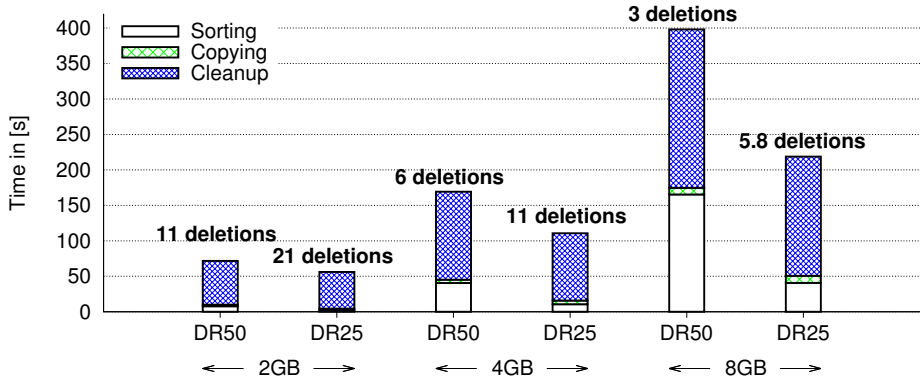
Figure 12.5c and Figure 12.6c show that if we further increase the repofile threshold to 8 GB in the web server scenario and 16 GB in the YouTube scenario, the average cache hit rates do not go below 80%. Even in the worst case, cache hit rates are always higher than miss rates in the YouTube scenario.

12.3.3 Deletion Times

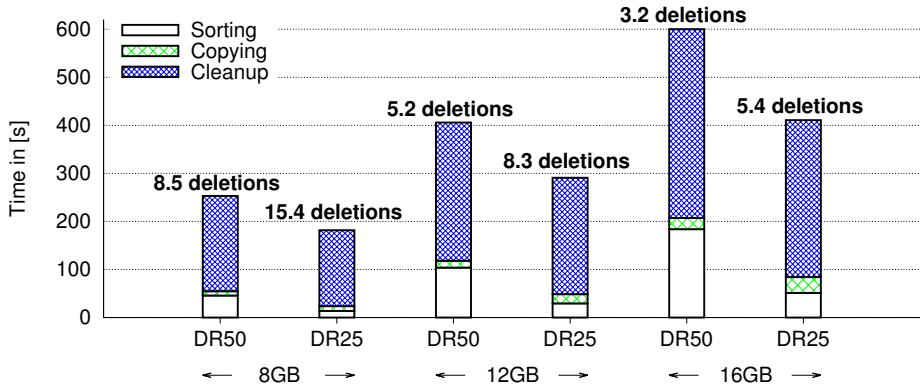
In this subsection, we evaluate the processing times to perform deletion operations in the web server and YouTube scenario. Figure 12.7 illustrates the overall processing times for deletions, i.e., deletion times, and the number of deletion operations in each evaluated scenario. The deletion time is split into subparts for sorting chunks of deleted content, copying files from the repofile and cleanup of the B-tree. The number on top of each bar denotes the average number of deletion operations (see Subsection 12.2.2) during one day in the corresponding configuration.

Figure 12.7a illustrates the deletion times as well as the number of deletion operations in web server scenarios with repofile thresholds of 2 GB, 4 GB and 8 GB. For every repofile threshold, the labels on the x-axis denote deletion ratios of 50% (DR50) and 25% (DR25). For small repofile thresholds of 2 GB, deletions with DR25 take only 16.2% less time than for DR50. This is because most of the time is required for updating file references and deleting entries in the B-tree (cleanup). For DR50, B-tree cleanup requires 86% of the total deletion time and for DR25, it requires even 92%. With increasing repofile threshold, the time for sorting increases. For a repofile threshold of 8 GB, sorting is responsible for 41.5% of the deletion time for DR50 and only for 18.6% for DR25. DR50 requires more than 300% additional time for sorting and 24.8% more time for cleanup compared to DR25, while their difference for copying is insignificant. For one deletion operation with a repofile threshold of 8 GB, DR25 requires 45% less time for the deletion than DR50. Considering that DR25 requires nearly twice as many deletions over one day, DR25 results only in 6% longer deletion times than DR50. However, due to the increased cache hit rate for DR25 (see last Subsection), it may be worth investing 6% more time for deletion.

Figure 12.7b shows deletion times and number of deletion operations for the YouTube scenario. A deletion operation with a repofile threshold of 8 GB takes considerably less time than for the web server scenario: for DR50, it is 36.4% less



(a) Web Server Scenarios



(b) YouTube Scenarios

Figure 12.7: Deletion Times and Number of Deletions for Persistent Caching.

time and for DR25 it is 16.9% less time. Because files are larger in the YouTube scenario, i.e., have more chunks, fewer files are stored in the repository for the same repofile threshold. Due to the sequential request strategy in NDN, chunks of the same file are already (more or less) ordered. However, since popular files are continuously pushed down in the *delete_queue*, the sorting overhead increases with the number of deleted files, i.e., no First-In-First-Out (FIFO) deletion strategy. Fewer files that contain more ordered chunks (YouTube scenario) result in a less fragmented repository file than many files with fewer ordered chunks (web server scenario) and can, therefore, be sorted faster. As a result, sorting requires only 18.1% of the deletion time for DR50 and only 7.5% for DR25. However, the larger the repofile becomes, the higher is the overhead for sorting and cleanup. For a repofile threshold of 16 GB, sorting is responsible for 30.6% of the deletion time for DR50 and 12.5% for DR25. Although sorting takes 3.5 times more time with DR50 compared to DR25, a deletion operation with DR25 requires only 31.6% less

time compared to DR50. This is because the overhead for copying is not negligible anymore, i.e., 41.6% more time for DR25, and cleanup becomes more expensive for DR25, i.e., only 16.9% less time compared to DR50. When taking into account the number of deletions, DR25 results in 15.6% longer deletion times than DR50. Considering that the overall hit rate for DR50 and DR25 is almost the same (less than 1% difference), it may be a better strategy to use DR50 instead of DR25 in the YouTube scenario.

12.4 Discussion

12.4.1 Integration of Persistent Caching with Regular Caching

While every received Data message is automatically stored in the content store (regular caching), persistent caching in repositories may only be used for static content with a certain lifetime, i.e., no real-time data. The basic idea of persistent caching is to store popular static content objects closer to requesters.

Every received Data message, which is not already cached in the content store, can be forwarded and stored in repositories, i.e., persistent caching does not result in additional traffic and we do not consider off-path caching. If content is cached persistently, the repository can indicate local content availability in the FIB such that received Interests can be forwarded via internal faces to the repository. Thus, forwarding decisions can be based on local decisions only (no coordination required). Internal faces in the FIB have priority over network faces such that Interests are only forwarded to another router if content is not locally available.

12.4.2 Chunk-based vs. Object-based Persistent Caching

Regular caching is chunk-based to enable retransmissions of specific chunks. For persistent caching, we process content based on object granularity because content is requested sequentially, i.e., up to n chunks at the same time depending on the pipeline size. Thus, a high delay variability between chunks, e.g., by caching chunks individually, would drastically degrade download performance.

However, for large files such a strategy may result in inefficient resource usage. For example, if only the first few seconds of a 2 hours movie would be retrieved, the content would be considered as popular as if the entire 2 hours would be requested. To address this issue, well-known techniques for Video on Demand (VoD) may be applied. VoD traffic is divided into segments (not to be confused with NDN chunks). A VoD segment denotes a specific interval of the video, e.g., a two minutes interval that contains multiple NDN chunks. Similar to VoD caching, only the beginning of a video [211] may be stored or the video clip may be divided into segments of increasing size [224] to better utilize storage space. To avoid disruptions for popular videos, pre-fetching [180] may retrieve missing segments once the initial frames are requested.

The frequency of queue updates, i.e., when content needs to be pushed back to the end of the *delete_queue*, should be independent of the content size to avoid discrimination of small files with only a few chunks compared to large files, which would otherwise have much more update operations. Since we assume that the first chunk is requested in every download, we push content entries back based on the request of the first chunk. If videos are divided into segments, a separate update operation can be performed for every segment. However, segment $i + 1$ should only be kept if segment i is still cached.

12.4.3 Video on Demand (VoD) Support

NDN is promising for video streaming due to its implicit multicast support, i.e., caching Data in the content store and aggregating Interests in the PIT. Recently, DASH (Dynamic Adaptive Streaming over HTTP) [208] has gained significant attention in the VoD community because it enables adaptive bitrate streaming based on the client's capabilities and network conditions. DASH can also be supported over NDN [195, 84, 130, 165]. After downloading a manifest file, which describes the available bit rates at a server, content downloads can be initiated by the client.

While DASH may benefit from NDN caching, it also introduces additional challenges. If the data rate is specified in the content name [195], [84], cache efficiency would be reduced significantly because different encodings of the same content would look differently. Furthermore, end-to-end throughput can not be assessed reliably in the presence of caching. Transcoding content adaptively at every node would result in significant processing overhead and would require new trust models to replace publisher signatures. Thus, to avoid transcoding, publishers can split content into different layers during encoding [165]. Then, a user needs to consume at least one layer (base layer) and can increase the perceived quality by retrieving enhancement layers. Persistent caching could support such an approach by storing layers individually giving higher priority to base layers compared to enhancement layers.

12.4.4 Deletion Overhead

Our evaluations have shown that deletion operations result in a large processing overhead to update the B-tree, sort the deleted chunks and delete content from the repofile. For some applications, the processing overhead may not be critical, e.g., delay-tolerant networking (see Chapter 11) or custodian-based information sharing [118], and it can be performed as maintenance operation during off-peak hours.

However, if persistent caching is used as network cache alongside content routers, service interruptions (see Subsection 12.2.2) have a larger impact on caching performance because no new content can be included during deletion operations. In this case, a (read-only) repository could be used during the deletion to continuously serve existing content. In addition, a router may use multiple repositories at the same time (load balancing). Then, if one repository performs a deletion,

the other repository may still accept content. In the worst case, i.e., if only one repository is used and Interests cannot be satisfied due to a local deletion, Interests may be forwarded to the next content router on the path to the content source.

12.5 Conclusions

Persistent caching is required to support delay-tolerant networking or provide high content availability similar to content delivery networks (CDNs). In this chapter, we have extended the repository implementation in CCNx to support persistent caching in repositories. A fundamental requirement for persistent caching is content deletion during operation, i.e., without deleting or resetting the entire repository, which is not supported by CCNx. Our approach for content deletion is based on a *delete_queue*, which keeps the most popular files at the tail of the queue. If disk space needs to be released, content can be removed from the head of the queue.

We have performed extensive experimental evaluations for different configurations in a web server and YouTube scenario. In every scenario, new content has been generated periodically such that deletion operations in the repository were necessary due to limited space. Evaluations have shown that our design can maintain high cache hit rates in both scenarios, but performance depends on the reserved repofile size for caching. Although repositories are slightly larger in the YouTube scenario due to larger file sizes, the repositories need to store a smaller percentage of all content to achieve high cache hit rates. For example, in web server scenarios a repofile size of 4 GB, which corresponds to 21% of all included content during a day, results in cache hit rates larger than 90%. In YouTube scenarios, a repofile size of 12 GB, which corresponds to 14% of all included content during a day, results in cache hit rates larger than 98%. High cache hit rates in network caches of edge routers are beneficial for both users and network operators. While network operators can reduce network traffic to the core network to improve network availability and reduce operational costs, users may benefit from faster content downloads (shorter delays, less RTT variability) as well as partial service and content availability if links to the core network are overloaded or disrupted.

In the web server scenario, it is a better strategy to have more frequent deletions of fewer content to obtain higher cache hit rates. More frequent deletions of fewer files do not require much more time for deletions, i.e., not much longer service interruptions, than fewer deletions of many files. In the YouTube scenario, fewer but larger deletions are better. Compared to the web server scenario, the sorting overhead is significantly smaller because fewer files can be stored at the same space (files are larger) and chunks of the same file are already ordered. In addition, because most requests are addressed to the most popular classes, the additional gain of keeping less popular content in the repository is only minimal and may not justify more frequent deletion operations.

We have implemented persistent caching based on the repository implementation of CCNx 0.8.2, which uses a B-tree to keep references to stored content in

the filesystem. For caching in delay-tolerant networking, the overhead for deletions may be negligible but to increase efficiency of network caches in content routers, other repository implementations may be evaluated, e.g., storing files or even chunks in separate repofiles to reduce cleanup and sorting overhead, or even develop a repository implementation with a database. However, all of these solutions would come with their own disadvantages that would need to be evaluated.

Part IV

Conclusions and Outlook

Chapter 13

Conclusions

Information-Centric Networking (ICN) is a new approach to address communication challenges of the Future Internet such as scalability with an increasing amount of traffic and security, e.g., exchange of authentic content. Although ICN has experienced a lot of momentum in recent years, research is still in its beginnings and researchers are facing many open challenges that need to be addressed. In this thesis, we have investigated ICN in the context of information-centric mobile and wireless ad hoc communication. Section 13.1 gives an overview of challenges addressed in this thesis. Our contributions are summarized in Section 13.2.

13.1 Addressed Challenges

There are multiple challenges and problem domains that were addressed in this thesis. In this section, we give an overview of these challenges.

- **Evaluation Methodology, Tools and Software:** Information-centric networking is a new communication paradigm based on names instead of endpoint identifiers. Analytical evaluation of ICN is very complex and nearly infeasible without significant simplifications because content requests are not independent of each other, i.e., cache states are influenced by different requests. Since simulation and emulation tools for NDN have only emerged recently, no common evaluation methodology has been established in related NDN literature. We have analyzed several different performance parameters and have used different evaluation tools. In particular, we have implemented an NDN framework for the network simulator OMNeT++ and extended NS3-DCE such that it enables the automated evaluation of many parameters in parallel on a Linux cluster.

- **Content Discovery:** Because content is routed based on names, requesters need to know available content names before they can request it. In a distributed environment, requesters may not be able to retrieve content via a centralized Google-like search engine because content availability may change

based on the connectivity. Thus, even if a certain content may exist, a requester may not be able to retrieve it. Another challenge is the design of namespaces to enable efficient discovery (collisions vs. discovery time). We have addressed this topic by the description of three opportunistic content discovery algorithms, namely, Regular Interest Discovery, Enumeration Request Discovery and Leaves First Discovery, as well as their evaluation in different namespaces. Furthermore, we have described alias mappings to enable expressive content descriptions without re-naming existing content objects and, hence, without degrading content availability in caches.

- **Performance of NDN broadcast:** NDN messages without source or destination addresses do not target specific hosts but (potentially) multiple hosts at the same time. Most existing wireless NDN protocols use broadcast communication for all Data transmissions. However, broadcast communication has implications on message overhead since broadcast requests may result in duplicate Data transmissions if multiple content sources are available. We have investigated information-centric wireless broadcast communication in terms of energy consumption and duplicate Data transmissions. Then, we have designed two approaches to increase wireless communication efficiency, namely, Dynamic Unicast and RC-NDN.

- **Routing in Wireless Multi-hop Environments:** Although NDN nodes do not need to know the existence of particular content objects (due to longest-prefix matching), they still need to know where content may be found, i.e., they need forwarding entries towards content sources. While forwarding entries may be efficiently configured in wired networks, it may be more challenging in mobile and wireless networks because connectivity may change frequently, i.e., configured forwarding entries may expire. Yet, flooding all messages in a wireless network may overload the network by redundant transmissions and result in frequent collisions. We have described two wireless routing protocols. The first routing protocol uses overhearing to register name prefixes and defines preferred forwarders to setup efficient forwarding paths to content sources. The second routing protocol (multi-hop Dynamic Unicast) uses Interest flooding to find a content source and registers a unicast forwarding path when Data returns on the reverse path.

- **Delay-tolerant NDN Communication:** NDN communication is symmetric, which means that Interests create soft states in forwarding nodes such that Data can travel on the reverse path. In intermittently connected networks, there may be no continuous paths between requesters and content sources and the intercontact time between mobile nodes may be long. Thus, Interests may expire at intermediate nodes and never reach a content source or the reverse Data path may break if nodes have moved away. Most existing routing approaches either target connected networks or intermittently connected networks but they do not work (efficiently) in case of other (or dy-

namically changing) network conditions. We have addressed this challenge by agent-based content retrieval that implements DTN support at the application layer and, thus, can be combined with multi-hop routing protocols.

- **Caching:** NDN can increase communication scalability and avoid routing loops since received content is cached in the content store and does not need to be requested again. However, the content store needs to support line-speed because every received Data message is automatically cached before being forwarded. Since fast memory is power hungry and only available in rather small capacities, content stores can not be used to store content for a long time. We have developed and implemented persistent caching, which stores popular delay-tolerant content persistently for a longer time while real-time data can still be stored in the content store. Our approach can not only be used for delay-tolerant networking via agents but also for network caches in edge routers to reduce network traffic.

13.2 Thesis Summary

In this thesis, we have investigated information-centric networking for mobile and wireless (ad hoc) communication. Our main contributions are presented in two parts. Part II includes our contributions for opportunistic one-hop communication and Part III lists our contributions for wireless multi-hop communication.

13.2.1 Part II: Opportunistic Information-Centric One-hop Communication

In Chapter 4, we have investigated opportunistic content discovery in disaster scenarios, where fixed infrastructures may not be available. We have described three content discovery algorithms. Enumeration Request Discovery (ERD) is similar to DNS service discovery (DNS-SD) and requests only name components, Regular Interest Discovery (RID) is based on implicit content discovery and requests always the first segment of a content object, and Leaves First Discovery (LFD) is a combination of both, which uses RID to quickly reach the leaf-level of a name tree and ERD on the leaf level. We have performed evaluations in different scenarios using flat and structured (e.g, hierarchical) namespaces. In flat namespaces, which are optimal for ERD, all three algorithms perform similar in terms of content discovery times, but LFD requires up to 89% fewer Data traffic than RID, while the difference to ERD is negligible. However, if namespaces are structured, ERD performs significantly worse than LFD and RID. For example, in hierarchical namespaces, LFD results in up to 54% shorter discovery times (and 62% lower Data overhead) than ERD as well as up to 40% shorter discovery times (and 55% lower Data overhead) than RID. Thus, for a general namespace structure, LFD performs best in terms of message overhead and discovery time.

After content names are known, desired content can be retrieved whenever there is an opportunity for retrieval. However, if contact times to content sources are short, content retrievals may be disrupted. In Chapter 5, we have implemented a content retrieval application for requesters that stores Meta Data and partial files persistently such that disrupted content retrievals can be resumed even in case of long disruptions (when the cache may be cleared). Evaluations on wireless mesh nodes have shown that our design works efficiently even in case of short contact times to content sources, and the processing overhead to maintain Meta data is negligible. Furthermore, we have measured the power consumption of NDN on wireless mesh nodes during unicast and broadcast communication. In particular, we have observed that listener nodes (neither requesters nor content sources) that overhear broadcast messages have a 22% higher power consumption compared to idle mode. Thus, if only a few requesters are interested in a content object, unicast communication could save a substantial amount of energy.

Furthermore, since broadcast requests can address multiple content sources at the same time, they may also trigger duplicate Data transmissions and collisions. In Chapter 6, we have shown that broadcast delays need to be considerably large to limit the number of duplicate Data transmissions in case of multiple content sources. Then, we have described Dynamic Unicast (DU), which uses broadcast communication only until a content source is found and addresses subsequent requests via unicast to the same content source until it becomes unavailable. While DU is clearly more efficient in case of single requesters and multiple content sources, our evaluations have shown that DU is also up to 56% faster than broadcast in networks with multiple concurrent requesters. In fact, broadcast communication is not as efficient as expected in terms of message efficiency if all nodes in the network request content concurrently. In particular, with DU, content sources transmit on average only 2% more Data messages and requesters transmit only 18% more Interests but receive at the same time 85% fewer duplicate Data messages compared to broadcast communication.

Efficiency of opportunistic NDN broadcast communication can also be improved by our Raptor Codes enabled NDN framework (RC-NDN), which has been described in Chapter 7. RC-NDN encodes Data messages at content sources to increase diversity of broadcast transmissions. Instead of requesting and retransmitting exactly the same content copies (as with NDN), requesters retrieve differently encoded Data messages such that multiple requesters can profit from each other. Our evaluations have shown that RC-NDN can significantly increase communication efficiency in dense wireless environments, such that requesters can send up to 93% fewer Interest messages and content sources transmit up to 85% fewer Data messages. Thus, with RC-NDN, requesters can reduce content retrieval times by up to 83% compared to original NDN, while they are also saving resources on the wireless medium. However, since re-encoding of Data messages in intermediate nodes would require new signatures and transitive trust models (because re-encoded Data is equivalent to new content), we only apply RC-NDN to opportunistic one-hop communication.

13.2.2 Part III: Information-Centric Wireless Multi-hop Communication

If a requester can not find a content source in direct transmission range, multi-hop routing is required to forward Interests towards potential content sources. In Chapter 8, we have investigated multi-hop broadcast communication in the context of wireless community networks. First, we have explored strategies to efficiently configure forwarding entries based on overheard Data messages. Since there may be many potential forwarders during multi-hop broadcast communication, we have, then, presented a concept to elect preferred forwarders, which forward Interest messages slightly faster than non-preferred forwarders. Evaluations have shown that this approach results in up to 47% shorter content retrieval times than regular NDN due to fewer duplicate Interest transmissions and collisions. Furthermore, since messages do not contain endpoint identifiers, concurrent requests can be efficiently aggregated such that content sources send only 3% more Data messages for three concurrent requesters (which are located in disjoint network partitions and can not overhear each other) compared to 1 requester. However, forwarding efficiency may degrade in case of high host churn, i.e., if preferred forwarders are regularly moving away.

In Chapter 9, we have extended DU for multi-hop communication by defining new forwarding strategies and FIB update mechanisms. In addition, we have implemented a Content Request Tracker, which keeps track of unicast requests and can replace multiple unicast transmissions with one broadcast transmission (for improved communication efficiency). Our evaluations have shown that despite unicast paths, multi-hop DU can result in significantly fewer Data transmissions by content sources compared to host-based communication due to Interest aggregation and in-network caching. In particular, for 32 concurrent requesters that retrieve content via multiple hops from a content source, the Interest and Data overhead of multi-hop DU (compared to 1 requester) increases only by a factor of 7 for Interests and 6.8 for Data messages, while content retrieval times increase by a factor of 10.2. Multi-hop DU performs best in static scenarios but the performance degrades only slightly if nodes move with pedestrian speeds. However, for vehicular speeds the performance degrades more due to more frequent path breaks, which are only detected after Interests have expired.

To increase throughput in mobile and wireless multi-hop networks, path breaks (in case of mobility) or collisions need to be detected quickly. This can be achieved by Interest lifetime values, which are close to round-trip times (RTTs) such that retransmissions can be performed quickly. In Chapter 10, we have evaluated existing retransmission timers and described two new algorithms to adapt Interest lifetimes, i.e., CCNTimer and WMA. Our evaluations have shown that in low traffic scenarios, CCNTimer can decrease content retrieval times by up to 45% compared to existing algorithms and up to 324% compared to the default Interest lifetime of 4 seconds (without increasing the number of Interest transmissions). In high traffic scenarios, CCNTimer performs similar ($\pm$2%) to existing algorithms. In particular,

we have observed that doubling Interest lifetimes in case of timeouts may not be required for NDN because content may often be found in intermediate caches such that slightly increased Interest lifetimes are enough to retrieve subsequent segments in time.

Yet, in case of intermittent connectivity, multi-hop routing may not be possible due to potentially long intercontact times and asymmetric forwarding paths. In Chapter 11, we have presented agent-based content retrieval (ACR) for delay-tolerant communication. In ACR, requesters can delegate content retrieval to mobile agents that move closer to content sources, retrieve the content and return it to requesters. Evaluations have shown that ACR is superior to multi-hop routing in low and intermediate node densities where multi-hop routing does not work or results in frequent disruptions. However, even for high node densities, ACR has resulted in a lower message overhead compared to multi-hop DU for up to 4 agent delegations since only agents need to transmit messages and not every node on the multi-hop path. Hence, ACR is beneficial for large content sizes, long path lengths and works even in case of high mobility when being combined with one-hop DU. In particular, the overhead of ACR to retrieve large content objects from multiple content sources with one-hop DU (resume operations) is negligible. Agents store retrieved content persistently in their repositories to support long intercontact times between communication opportunities. Therefore, repository sizes may grow with time for an increasing number of delegated content retrievals such that efficient deletion mechanisms are required.

In Chapter 12, we have presented a design for persistent caching that maintains popular delay-tolerant content in the repository and deletes unpopular content if free space is required. However, persistent caching is not limited to delay-tolerant networking via agents, but can also be applied to network caches at edge routers, where it can be combined with short-term caching in the content store. While short-term caches need to be implemented in fast memory to support line-speed, persistent caching for popular delay-tolerant content can relax this requirement and use larger (but slower) storage. Despite slower access latencies in persistent storage, requests can still be satisfied quicker than without persistent caching because Interests do not need to be forwarded further over the network (shorter path length). We have validated our design by extensive measurements using YouTube and web server traffic. Depending on the allocated storage size for persistent caching, high cache hit rates can be achieved. In the web server scenario, it is a better strategy to have more frequent deletions of fewer content to obtain higher cache hit rates. In the YouTube scenario, fewer but larger deletions are better because most requests are addressed to the most popular content classes, hence, the gain of keeping less popular content in the repository is only minimal and may not justify more frequent deletion operations.

Chapter 14

Future Work

Information-centric wireless networking is a new research direction and there are many open issues that could not be addressed in this thesis. In this section, we briefly describe topics that require further investigations. We divide the topics in six main fields: Content Discovery, Optimized MAC layer, Multi-hop Routing, Delay-tolerant Communication, Hierarchical Caching as well as Privacy and Trust.

14.1 Content Discovery

In Chapter 4, we have described Leaves First Discovery (LFD) for opportunistic content discovery. LFD uses enumeration requests to retrieve content names from repositories. Since enumeration responses are identified by the IDs of the repositories holding the content, enumeration responses may look different due to different repository IDs despite containing the same content. To better identify similar content at different repositories, it may be more efficient to identify repository collections by the hash of all content names included in the collection. In addition, implicit content discovery (with regular Interests) may become more efficient in terms of Data overhead when containing a flag to retrieve only meta information, e.g., only the content name without payload.

14.2 Optimized MAC Layer

Current MAC layer protocols are efficient for unicast communication (rate adaptation, sleep mechanisms) but they are not optimized for broadcast communication. Rate adaptation during unicast communication between two nodes is based on signal strength, which in turn is based on the distance between two nodes. Close nodes can, therefore, transmit at a higher rate while nodes farther away can only communicate at a lower rate. During broadcast communication, the data rate is usually set to the lowest supported rate such that all nodes can receive the content at the same rate. However, this may not be an efficient strategy if most nodes are close to a sender and could receive content at a higher rate. Thus, it may be be-

neficial to use rate adaptation also for broadcast communication. For example, if 80% of the nodes are close to a sender, the data rate can be switched to a higher rate. Depending on the type of content, the remaining 20% nodes could retrieve the content from closer nodes, e.g., from one of the previous requesters that belong to the 80% group, or in case of a video stream they may just display the video with a lower quality.

Furthermore, by avoiding redundant functionality at the MAC layer and the CCND, information-centric wireless communication could be made more efficient. For example, Data messages are delayed at the CCND to enable duplicate Data suppression but the same packets are also delayed at the MAC layer for collision avoidance. Thus, after NDN messages have been forwarded to the MAC layer (but not yet transmitted), duplicate messages cannot be identified anymore. By enabling duplicate Data suppression at the MAC layer, forwarding delays at the NDN layer could be reduced (because messages can still be suppressed at the MAC layer). Similarly, duplicate Interest transmissions could be avoided if duplicate Interests would be identified on the MAC layer shortly before transmission.

In addition, we have seen that broadcast multi-hop communication may result in many duplicate Interest and Data transmissions. Besides a higher energy consumption for transmission and reception of messages, nodes may switch to sleep modes less frequently because they are constantly receiving messages. In Chapter 8, we have described a concept for preferred forwarders in multi-hop broadcast communication. A similar concept may be used at the MAC layer to control duty cycles. In particular, since non-preferred forwarders are not essential for message forwarding, they could enter sleep modes more frequently than preferred forwarders to save energy even if they overhear broadcast messages.

Moreover, Dynamic Unicast (see Chapters 6 and 9) could be implemented more efficiently directly at the MAC layer (instead of the IP layer). Thus, a node could broadcast messages until it receives a response from a neighbor node. Then, it can send subsequent requests directly to the same node (or even allocate a separate wireless channel to the neighbor node for subsequent communication). Ideally, such an approach could avoid fragmentation by adapting frame sizes based on the NDN payload (e.g., jumbo frames), but otherwise, fragmentation functionality needs to be integrated. To optimize path establishment over multiple hops (area coverage vs. path stability), message forwarding may be based on the nodes' location. While LAL [100] appends GPS coordinates to messages, message forwarding may also be based on RSSI values, which do not require additional equipment.

14.3 Multi-hop Routing

In Chapter 9, we have described Dynamic Unicast for multi-hop communication. While this approach showed good performance in static and pedestrian mobility scenarios, the performance decreased slightly with random vehicular mobility. The reason for this are path breaks when neighbor nodes see each other only for a short

time. Path breaks can only be detected after an Interest expiration and we used the default Interest lifetime of 4 seconds in our evaluations. To increase data rates in high mobility networks, it may be more efficient to set NDN parameters adaptively. For example, Interest lifetimes can be set based on measured RTT values similar to our algorithms in Chapter 10. In addition, pipeline sizes can be set dynamically by increasing them on established paths (higher bandwidth utilization) and decreasing them in case of path breaks (e.g., set the pipeline size to 1 until a new path is discovered) or network congestion.

Another topic that requires more investigation is multi-homing. Since ICN supports ubiquitous caching, RTT measurements may not only include measurements for end-to-end transmissions but also shorter samples due to cached content in intermediate nodes. Cached content is problematic if only partial content can be found in caches and other parts of the content may need to be retrieved over the entire path (high RTT variation). To address this issue, related works [171, 66, 62, 40] include source identifiers in messages to differentiate between nodes (caches and content sources). A source identifier specifies the first node that replies to an Interest (hence, it is not changed on the reverse hop-by-hop path to the requester). Our algorithms in Chapter 10 could be extended with source identifiers to support multi-homing. However, source identifiers may affect information-centric forwarding, e.g., modifications may be required to Interest aggregation and content caching. An alternative approach to avoid source identifiers may be based on RTT outlier detection such that only RTT samples within a certain range may be processed while single short RTT samples (due to cached content) may be ignored.

Finally, the scope of Interest flooding may be limited via hop counters in Interest messages to avoid message transmissions in the entire network [215]. To support hop counters, Interest aggregation mechanisms may be modified to avoid that Interests with low hop counters (which may not be able to reach certain content sources) can block Interests with larger hop counters.

14.4 Delay-tolerant Communication

In Chapter 11, we have described agent-based content retrieval (ACR) for delay-tolerant communication. There are several ways to further improve ACR depending on scenario and application requirements. While we were selecting agents randomly within the one-hop neighborhood, there may be more efficient ways for doing this. For example, if it is known where the content might be found, agents can be selected based on their probability to travel to this area. The required information, e.g., based on social interactions [145] or GPS traces [109], could be collected locally on each device and could be exchanged during agent delegations as optional parameters. Agent selection can also be based on hybrid approaches, where requester may download potential agent information for certain areas when connected to the Internet and use this information when being disconnected [113, 185]. To enable agent selections based on locations, CCNx/NDNx

and NS3-DCE (for the evaluation) need to be extended by location parameters. In addition, LTE direct [21] or Wi-Fi Aware Networking [35] may also support more efficient information-centric device-to-device communication, e.g., to discover suitable agents or exchange environmental information.

To increase robustness of ACR, content retrieval can be delegated to multiple agents in parallel. Currently, the number of delegated agents is set statically, but it may be more efficient to set it adaptively based on the environment. For example, if there is a high probability (or even certainty) that an agent returns quickly to deliver the content, only one agent delegation is required while more agent delegations are only needed in case of a high uncertainty. To decrease duplicate Data transmissions and increase diversity of content transmissions, content sources may use our RC-NDN approach (see Chapter 7). By this, each agent may only retrieve a certain number of Data messages and deliver it to the requester where content can be combined and decoded.

Furthermore, our ACR approach can be combined with multi-hop routing. In general, the decision whether to use ACR or multi-hop routing may not only depend on environmental parameters (e.g., node density and node mobility) but also on application requirements (e.g., quick content retrieval times vs. message or energy efficiency). If a quick content retrieval is important, a requester may try to retrieve content via multi-hop routing whenever possible. Hence, in extreme environments, the combination may be straightforward. In a sparse environment, a requester may initially try to retrieve content via multi-hop routing and delegates content retrieval only if nothing can be retrieved. In a dense environment, a requester can directly retrieve content via multi-hop routing before delegating agents. However, in more dynamic time-varying environments, the combination of ACR with multi-hop routing may be more complex. For example, it needs to be explored what requesters should do if they have received 80% of the content via mult-hop communication before they have experienced a disruption. Should content retrieval be delegated to an agent node? How long should a requester try to retrieve the missing fragments before delegating content retrieval to an agent node? To answer these questions further investigation with more realistic mobility models, e.g., SUMO [58], SLAW [132] or SMOOTH [152], are required.

14.5 Persistent Caching

In Chapter 12, we have described persistent caching with repositories to support agent-based content retrieval (see Chapter 11) in delay-tolerant networks and support hierarchical caching (persistent caching combined with short-term caching) in network caches at edge routers. Our current persistent caching implementation maintains popular content as long as possible in a repository and deletes unpopular content when free space is required. While this strategy may be efficient in hierarchical static networks, alternative caching strategies may be explored for mobile and opportunistic networking scenarios to optimize storage utilization and cache

hit rates, e.g., availability of popular content vs. content diversity.

Furthermore, since persistent caching shows good performance with YouTube and web server traffic, it may be promising to evaluate it also in large realistic network topologies. Due to the resource requirements (memory and storage), it may not be possible to evaluate this with NS3-DCE, but it can be implemented in ndnSIM [30] instead. In addition, more efficient repository implementations may be evaluated for network caches to reduce the processing overhead for content deletions and cleanup of data structures. For example, alternative implementations may be based on databases such as SQLite [23].

14.6 Privacy and Trust

Security topics (including privacy and trust) are not addressed in this thesis. Nevertheless, they are important topics that need further attention. While signatures are important to ensure authenticity and non-repudiation of published content, this is only relevant if users trust the identity of a signer. Thus, suitable trust models [230] are required. In particular, the meaning of trust needs to be defined: does it mean that an identity is legitimate or that a publisher creates legitimate (high-quality) content? Hence, trust models for distributed environments may be based on several mechanisms and parameters such as certificate chains, ratings or reputation systems.

Another important topic is privacy. It is still not clear whether user privacy can be fully protected in information-centric communication. By avoiding endpoint identifiers in NDN messages, it may be more difficult to track a user's behavior because traffic is not exclusively routed to an IP address. Even if approaches use node IDs as temporary next hop or content locator (see Chapters 6, 9, 11), attackers could only infer that a message has been transmitted by a certain node ID (if they are sufficiently close to overhear the transmission) but they would not know whether the node is a requester or only a forwarder. Only the identity of content publishers may not be easily masked because of content signatures that are required to ensure authenticity and non-repudiation (see above).

However, while payloads can be encrypted, content names need to remain in clear text because they are required for routing. Thus, by observing content names in Interest or cached Data messages [29, 71], attackers may infer personal information about a group of users that is attached to the same router. There may be options to obscure traffic, e.g., via onion routing [85], but such solutions may increase message redundancy and decrease ICN efficiency, i.e., the main benefits to deploy ICN in the first place.

Chapter 15

Acronyms

AA	Application Agent
ACR	Agent-based Content Retrieval
ACK	Acknowledgment
ADAM	Administration and Deployment of Adhoc Mesh networks
AODV	Ad hoc On-demand Distance Vector Routing
ARQ	Automatic Repeat-Request
BLR	Beacon-less Routing
BP	Bundle Protocol
BPA	BP Agent
FIB	Forwarding Information Base
FIFO	First-In-First-Out
CCN	Content-Centric Networking
CCND	CCN Daemon
CDN	Content Distribution Networks
CEDO	Content-Centric Dissemination Algorithm
CFT	Content Flow Table
CLA	Convergence Layer Adapter
CRT	Content Request Tracker
CRT-S	CRT at Source

CRT-SR	CRT at Source and Requester
CS	Content Store
DASH	Dynamic Adaptive Streaming over HTTP
DCE	Direct Code Execution
DD	Directed Diffusion
DNS	Domain Name System
mDNS	Multicast DNS
DNS-SD	DNS Service Discovery
DoS	Denial-of-Service
DP	Data Pause
DRAM	Dynamic RAM
DREAM	Distance Routing Effect Algorithm for Mobility
DSR	Dynamic Source Routing
DTN	Delay-tolerant Networks
DU	Dynamic Unicast
DYMO	Dynamic MANET On-demand Routing
ERD	Enumeration Request Discovery
E-CHANET	Enhanced-Content-centric multi-Hop wireless NETwork
FIA	Future Internet Assembly
GPL	GNU General Public License
GPS	Global Positioning System
GPSR	Greedy Perimeter Stateless Routing
GPU	Graphics Processing Unit
HTTP	Hypertext Transfer Protocol
ICN	Information-Centric Networking
ICP	Interest Control Protocol
IFD	Interest Forwarding Delay

IoT	Internet of Things
IP	Internet Protocol
IPN	Interplanetary Internet
IT	Interest Table
LAL	Link Adaptation Layer
LDPC	Low-Density Parity-Check
LFBL	Listen-First-Broadcast-Later
LGPL	GNU Lesser General Public License
LFD	Leaves First Discovery
LSCR	Link State Content Routing
LT	Luby Transform
LTE	Long-Term Evolution
LRU	Least Recently Used
MAC	Medium Access Control
MANET	Mobile Ad-hoc Network
MPR	Multipoint Relay
MTU	Maximum Transfer Unit
NACK	Negative ACK
NDN	Named Data Networking
NLSR	Named-data Link State Routing
OLSR	Optimized Link State Routing
OSPFN	OSPF Based Routing Protocol for Named Data Networking
PFF	Parallel Face Forwarding
PIT	Pending Interest Table
PSN	Pocket Switched Networks
RAM	Random Access Memory
RCH	Raptor Coding Header

RC-NDN	Raptor Codes Enabled Named Data Networking
RID	Regular Interest Discovery
RONR	Reactive Optimistic Name-based Routing
RSSI	Received Signal Strength Indicator
RTO	Retransmission Timeout
RTT	Round Trip Time
SDN	Software-defined Networks
SFF	Single Face Forwarding
SLAW	Self-Similar Least-Action Walk
SLP	Service Location Protocol
SNR	Signal-to-Noise Ratio
SSDP	Simple Service Discovery Protocol
SUMO	Simulation of Urban Mobility
TCP	Transmission Control Protocol
TTL	Time to Live
UDP	User Datagram Protocol
UPnP	Universal Plug and Play
URI	Universal Resource Identifier
URL	Uniform Resource Locator
VoD	Video on Demand
WMA	Weighted Moving Average
WS	Web Service
WSN	Wireless Sensor Network
WWW	World Wide Web

Bibliography

[1] Android Issue 82: Support Wi-Fi ad hoc networking. January, 2008. [Online]. Available: https://code.google.com/p/android/issues/detail?id=82

[2] Asia Future Internet Forum (AsiaFI). [Online]. Available: http://www.asiafi.net/

[3] CCNx Basic Name Conventions. September, 2013. [Online]. Available: https://www.ccnx.org/releases/ccnx-0.8.2/doc/technical/NameConventions.html

[4] CCNx Content Object. July, 2013. [Online]. Available: https://www.ccnx.org/releases/ccnx-0.8.2/doc/technical/ContentObject.html

[5] CCNx Interest Message. July, 2013. [Online]. Available: https://www.ccnx.org/releases/ccnx-0.8.2/doc/technical/InterestMessage.html

[6] CCNx Name Enumeration Protocol. April, 2015. [Online]. Available: https://www.ccnx.org/releases/ccnx-0.8.2/doc/technical/NameEnumerationProtocol.html

[7] CCNx Repository Protocol. September, 2013. [Online]. Available: https://www.ccnx.org/releases/ccnx-0.8.2/doc/technical/RepoProtocol.html

[8] CCNx Software Archives. [Online]. Available: https://www.ccnx.org/software-archives/

[9] CCNx Synchronization Protocol. July, 2013. [Online]. Available: http://www.ccnx.org/releases/ccnx-0.8.2/doc/technical/SynchronizationProtocol.html

[10] European Future Internet Portal (FIA). [Online]. Available: http://www.future-internet.eu/

[11] INET Framework for the OMNeT++ network simulator. [Online]. Available: https://inet.omnetpp.org/

[12] LG Google Nexus 4 E960. [Online]. Available: http://www.lg.com/uk/mobile-phones/lg-E960-nexus-4-by-lg/technical-specifications

[13] Named Data Networking (NDN). [Online]. Available: http://named-data.net/

[14] NDN Interest Packet. July, 2015. [Online]. Available: http://named-data.net/doc/NDN-TLV/current/interest.html

[15] NDNx: The NDN Platform. [Online]. Available: http://named-data.net/codebase/platform/

[16] NS-3: Direct Code Execution. April, 2015. [Online]. Available: http://www.nsnam.org/\overview/projects/direct-code-execution/

[17] NS-3: discrete-event network simulator. [Online]. Available: https://www.nsnam.org/

[18] NSF Future Internet Architecture Project (NSF-FIA). [Online]. Available: http://www.nets-fia.net/

[19] PARC Offers Content-Centric Networking (CCNx) Software to Advance Next-Generation Internet. January, 2016. [Online]. Available: http://www.parc.com/news-release/111/parc-offers-content-centric-networking-ccnx-software-to-advance-next-generation-internet.html

[20] PCEngines. September, 2013. [Online]. Available: http://www.pcengines.ch/

[21] QUALCOMM LTE Direct. [Online]. Available: https://ltedirect.qualcomm.com/

[22] Rigol DM3058. September, 2013. [Online]. Available: http://www.rigolna.com/products/digital-multimeters/dm3000/dm3058/

[23] SQLite. [Online]. Available: https://www.sqlite.org/

[24] UMOBILE H2020 ICT Project. [Online]. Available: http://www.umobile-project.eu/

[25] Web Services Dynamic Discovery (WS-Discovery) Version 1.1. July, 2009. [Online]. Available: http://docs.oasis-open.org/ws-dd/discovery/1.1/os/wsdd-discovery-1.1-spec-os.html

[26] Wi-Fi Direct. [Online]. Available: http://www.wi-fi.org/discover-wi-fi/wi-fi-direct

[27] (2015, April) Project CCNx. April, 2015. [Online]. Available: http://www.ccnx.org/

[28] A. Abhari and M. Soraya, "Workload generation for YouTube," *Multimedia Tools and Applications*, vol. 46, no. 1, pp. 91–118, January 2010.

[29] G. Acs, M. Conti, P. Gasti, C. Ghali, and G. Tsudik, "Cache Privacy in Named-Data Networking," in *33rd IEEE International Conference on Distributed Computing Systems (ICDCS)*, Philadelphia, PA, USA, July 2013, pp. 41–51.

[30] A. Afanasyev. NS-3 based Named Data Networking (NDN) simulator. [Online]. Available: http://ndnsim.net

[31] B. Ahlgren, C. Dannewitz, C. Imbrenda, D. Kutscher, and B. Ohlman, "A Survey of Information-Centric Networking," *IEEE Communications Magazine*, vol. 50, no. 7, pp. 26–36, July 2012.

[32] R. Ahlswede, N. Cai, S.-Y. R. Li, and R. W. Yeung, "Network Information Flow," *IEEE Transactions on Information Theory*, vol. 46, no. 4, pp. 1204–1216, July 2000.

[33] J. Al-Karaki and A. Kamal, "Routing Techniques in Wireless Sensor Networks: A Survey," *IEEE Wireless Communications*, vol. 11, no. 6, pp. 6–28, December 2004.

[34] W. E. M. E. Alami, "An Agent-based Content-Centric Networking Application for Data Retrieval," University of Bern," Master Thesis, September 2013.

[35] W.-F. Alliance. Wi-Fi Aware Networking. [Online]. Available: http://www.wi-fi.org/discover-wi-fi/wi-fi-aware

[36] M. Allman. (2000, November) RFC 2988: Computing TCP's Retransmission Timer. [Online]. Available: http://tools.ietf.org/html/rfc2988

[37] M. Amadeo, C. Campolo, and A. Molinaro, "Enhancing content-centric networking for vehicular environments," *Computer Networks*, vol. 57, no. 16, pp. 3222–3234, November 2013.

[38] ——, "Multi-Source Data Retrieval in IoT via Named Data Networking," in *1st ACM Conference on Information-Centric Networking (ICN)*, Paris, France, September 2014, pp. 67–76.

[39] M. Amadeo, C. Campolo, A. Molinaro, and N. Mitton, "Named Data Networking: A Natural Design for Data Collection in Wireless Sensor Networks," in *IEEE Wireless Days (WD)*, Valencia, Spain, November 2013, pp. 1–6.

[40] M. Amadeo, A. Molinaro, C. Campolo, M. Sifalakis, and C. Tschudin, "Transport Layer Design for Named Data Networking," in *2nd IEEE Workshop on Name Oriented Mobility (NOM)*, Toronto, ON, Canada, May 2014, pp. 464–469.

[41] M. Amadeo, A. Molinaro, and G. Ruggeri, "E-CHANET: Routing, forwarding and transport in Information-Centric multihop wireless networks," *Computer Communications*, vol. 36, no. 7, pp. 792–803, April 2013.

[42] C. Anastasiades and T. Braun, "Dynamic Transmission Modes to Support Opportunistic Information-Centric Networks," in *2nd International Conference on Networked Systems (NetSys)*, Cottbus, Germany, March 2015, pp. 1–5.

[43] ——, "Towards Information-Centric Wireless Multihop Communication," in *13th International Conference on Wired & Wireless Internet Communications (WWIC)*, ser. LNCS 9071, Malaga, Spain, May 2015, pp. 367–380.

[44] C. Anastasiades, T. Braun, and V. Siris, "Information-Centric Networking in Mobile and Opportunistic Networking," *Lecture Notes in Computer Science on Wireless Networking for Moving Objects*, vol. 8611, pp. 14–30, August 2014.

[45] C. Anastasiades, W. El Maudni El Alami, and T. Braun, "Agent-based Content Retrieval for Opportunistic Content-Centric Networks," in *12th International Wired & WIreless Internet Communications (WWIC)*, ser. LNCS 8458, Paris, France, May 2014, pp. 175–188.

[46] C. Anastasiades, A. Gomes, R. Gadow, and T. Braun, "Persistent Caching In Information-Centric Networking," University of Bern, Institut für Informatik und Angewandte Mathematik, Technical Report IAM-15-001, May 2015.

[47] ——, "Persistent Caching in Information-Centric Networks," in *40th IEEE Conference on Local Computer Networks (LCN)*, Clearwater Beach, FL, USA, October 2015, pp. 237–245.

[48] C. Anastasiades, T. Schmid, J. Weber, and T. Braun, "Opportunistic Content-Centric Data Transmission During Short Network Contacts," in *IEEE Wireless Communications and Networking Conference (WCNC)*, Istanbul, Turkey, April 2014, pp. 2516–2521.

[49] ——, "Information-Centric Content Retrieval for Delay-Tolerant Networks," *Computer Networks*, 2016, DOI: 10.1016/j.comnet.2016.03.006.

[50] C. Anastasiades, A. Sittampalam, and T. Braun, "Content Discovery in Wireless Information-Centric Networks," in *40th IEEE Conference on Local Computer Networks (LCN)*, Clearwater Beach, FL, USA, October 2015, pp. 237–245.

[51] C. Anastasiades, N. Thomos, A. Striffeler, and T. Braun, "RC-NDN: Raptor Codes Enabled Named Data Networking," in *IEEE International Conference on Communications (ICC)*, London, UK, June 2015, pp. 3026–3032.

[52] C. Anastasiades, A. Uruqi, and T. Braun, "Content Discovery in Opportunistic Content-Centric Networks," in *5th International Workshop on Architectures, Services and Applications for the Next Generation Internet (WASA-NGI-V)*, Clearwater, FL, USA, October 2012, pp. 1048–1056.

[53] C. Anastasiades, L. von Rotz, and T. Braun, "Adaptive Interest Lifetimes for Information-Centric Wireless Multi-hop Communication," in *8th IFIP Wireless and Mobile Networking Conference (WMNC)*, Munich, Germany, October 2015, pp. 40–47.

[54] C. Anastasiades, J. Weber, and T. Braun, "Dynamic Unicast: Information-Centric Multi-hop Routing for Mobile Ad-hoc Networks," *Computer Networks*, 2016, DOI: 10.1016/j.comnet.2016.03.009.

[55] E. Baccelli, C. Mehlis, O. Hahm, T. C. Schmidt, and W. Wählisch, "Information Centric Networking in the IoT: Experiments with NDN in the Wild," in *1st ACM Conference on Information-Centric Networking (ICN)*, Paris, France, September 2014, pp. 77–86.

[56] B. Baron, P. Spathis, H. Rivano, and M. Dias, "Vehicles as Big Data Carriers: Road Map Space Reduction and Efficient Data Assignment," in *80th IEEE Vehicular Technology Conference (VTC Fall)*, Vancouver, BC, September 2014, pp. 1–5.

[57] S. Basagni, I. Chlamtac, V. Syrotiuk, and B. Woodward, "A Distance Routing Effect Algorithm for Mobility (DREAM)," in *4th Annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom)*, Dallas, TX, USA, October 1998, pp. 76–84.

[58] M. Behrisch, L. Bieker, J. Erdmann, and D. Krajzewicz, "SUMO - Simulation of Urban MObility," in *3rd International Conference on Advances in System Simulation (SIMUL 2011)*, Barcelona, Spain, October 2011, pp. 63–68.

[59] L. Blazevic, S. Giordano, and J.-Y. L. Boudec, "Self Organized Terminode Routing," *Cluster Computing*, vol. 5, no. 2, pp. 205–218, April 2002.

[60] E. Bourtsoulatze, N. Thomos, J. Saltarin, and T. Braun, "Content-Aware Delivery of Scalable Video in Network Coding Enabled Named Data Networks," University of Bern, Institut für Informatik, Tech. Rep. INF-003, September 2015.

[61] S. Braun, M. Monti, M. Sifalakis, and C. Tschudin, "An Empirical Study of Receiver-based AIMD Flow-Control Strategies for CCN," in *22nd International Conference on Computer Communications and Networks (ICCCN)*, Nassau, Bahamas, July - August 2013, pp. 1–8.

[62] ——, "CCN & TCP co-existence in the future Internet: Should CCN be compatible to TCP?" in *IFIP/IEEE International Symposium on Integrated Network Management (IM)*, Ghent, Belgium, May 2013, pp. 1109–1115.

[63] N. Briggs. (2014, January) CCNDC - Manual pages. [Online]. Available: http://www.ccnx.org/releases/ccnx-0.8.2/doc/manpages/ccndc.1.html

[64] J. Byers, M. Luby, M. Mitzenmacher, and A. Rege, "A Digital Fountain Approach to Reliable Distribution of Bulk Data," in *ACM SIGCOMM*, Vancouver, BC, Canada, January 1998, pp. 56–67.

[65] G. Carofiglio, M. Gallo, and L. Muscariello, "ICP: Design and Evaluation of an Interest Control Protocol for Content-Centric Networks," in *1st IEEE Workshop on Emerging Design Choices in Name-Oriented Networking (NOMEN)*, Orlando, FL, USA, March 2012, pp. 304–309.

[66] G. Carofiglio, M. Gallo, L. Muscariello, and M. Papalini, "Multipath Congestion Control in Content-Centric Networks," in *IEEE NOMEN*, Turin, Italy, April 2013, pp. 1–6.

[67] G. Carofiglio, L. Mekinda, and L. Muscariello, "FOCAL: Forwarding and Caching with Latency awareness in Information-Centric Networking," in *IEEE Global Communications Conference (GLOBECOM)*, San Diego, CA, USA, December 2015.

[68] V. Cerf, S. Burleigh, A. Hooke, L. Torgerson, R. Durst, K. Scott, K. Fall, and H. Weiss. (2007, April) RFC 4838: Delay-Tolerant Networking Architecture. [Online]. Available: https://tools.ietf.org/html/rfc4838

[69] V. Cerf, S. Burleigh, A. Hooke, L. Torgerson, R. Durst, K. Scott, E. Travis, and H. Weiss. (2001, November) Interplanetary Internet (IPN): Architectural Definition. [Online]. Available: https://tools.ietf.org/html/draft-irtf-ipnrg-arch-00

[70] M. Cha, H. Kwak, P. Rodriguez, Y.-Y. Ahn, and S. Moon, "I Tube, You Tube, Everybody Tubes: Analyzing the World's Largest User Generated Content Video System," in *ACM Internet Measurement Conference (IMC)*, 2007.

[71] A. Chaabane, E. D. Cristofaro, M. A. Kaafar, and E. Uzun, "Privacy in Content-Oriented Networking: Threats and Countermeasures," *ACM SIGCOMM Computer Communication Review*, vol. 43, no. 3, pp. 25–33, July 2013.

[72] W. Chai, D. He, I. Psaras, and G. Pavlou, "Cache "Less for More" in Information-centric networks," in *11th International IFIP Conference on Networking*, Valencia, Spain, May 2012, pp. 27–40.

[73] S. Cheshire and M. Krochmal. RFC 6762: Multicast DNS. [Online]. Available: https://tools.ietf.org/html/rfc6762

[74] ——. RFC 6763: DNS-based Service Discovery. [Online]. Available: https://tools.ietf.org/html/rfc6763

[75] R. Chiocchetti, D. Perino, G. Carofiglio, D. Rossi, and G. Rossini, "INFORM: a dynamic INterest FORwarding Mechanism for Information-Centric Networking," in *3rd ACM SIGCOMM workshop on Information-Centric Networking (ICN)*, Hong Kong, August 2013, pp. 9–14.

[76] R. Chiocchetti, D. Rossi, G. Rossini, G. Carofiglio, and D. Perino, "Exploit the known or explore the unknown? Hamlet-like doubts in ICN," in *2nd ACM SIGCOMM Workshop on Information-Centric Networking*, Helsinki, Finland, August 2012, pp. 7–12.

[77] K. Cho, M. Lee, K. Park, T. Kwon, Y. Choi, and S. Pack, "WAVE: Popularity-based and collaborative in-networking caching for content-oriented networks," in *IEEE INFOCOM workshop on Emerging Design Choices in Name-Oriented Networking (NOMEN)*, Orlando, FL, USA, March 2012, pp. 316–321.

[78] M. D. Choudhury, H. Sundaraman, A. John, D. D. Seligmann, and A. Kelliher, ""Birds of a Feather": Does User Homophily Impact Information Diffusion in Social Media?" *CoRR abs/1006.1702*, 2010.

[79] Cisco, "Visual Networking Index (VNI): Global Mobile Data Traffic Forecast Update, 2014 - 2019," February 2015, http://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/white_paper_c11-520862.pdf.

[80] T. Clausen, C. Dearlove, P. Jacquet, and U. Herberg. (2014, April) The Optimized Link State Routing Protocol Version 2. [Online]. Available: https://tools.ietf.org/html/rfc7181

[81] T. Clausen and P. Jaquet. (2003, October) RFC 3626: Optimized Link State Routing Protocol (OLSR). [Online]. Available: https://www.ietf.org/rfc/rfc3626.txt

[82] B. Cohen, "Incentives Build Robustness in BitTorrent," in *1st Workshop on Economics of Peer-to-Peer Systems (P2PEcon)*, Berkely, USA, June 2003, pp. 1–5.

[83] M. Demmer, J. Ott, and S. Perreault. (2014, June) RFC 7242: Delay-Tolerant Networking TCP Convergence-Layer Protocol. [Online]. Available: https://tools.ietf.org/html/rfc7242

[84] A. Detti, B. Ricci, and N. Blefari-Melazzi, "Peer-To-Peer Live Adaptive Video Streaming for Information Centric Cellular Networks," in *24th International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC)*, London, UK, September 2013, pp. 3583–3588.

[85] S. DiBendetto, P. Gasti, G. Tsudik, and E. Uzun, "ANDaNA: Anonymous Named Data Networking Application," in *19th Annual Network and Distributed System Security Symposium*, 2012.

[86] A. Donoho, B. Roe, M. Bodlaender, J. Gildred, A. Messer, Y. Kim, B. Fairman, and J. Tourzan, *UPnP Device Architecture 2.0*, UPnP Forum Std., February 2015. [Online]. Available: http://upnp.org/specs/arch/UPnP-arch-DeviceArchitecture-v2.0.pdf

[87] R. N. dos Santos, B. Ertl, C. Barakat, T. Spyropoulos, and T. Turletti, "CEDO: Content-Centric Dissemination Algorithm for Delay-Tolerant Networks," in *16th ACM international conference on Modeling, analysis & simulation of wireless and mobile systems (MSWiM)*, Barcelona, Spain, November 2013, pp. 377–386.

[88] Ericsson, "Ericsson Mobility Report," June 2015, http://www.ericsson.com/res/docs/2015/ericsson-mobility-report-june-2015.pdf.

[89] S. Eum, K. Nakauchi, M. Murata, Y. Shoji, and N. Nishinaga, "CATT: potential based routing with content caching for ICN," in *2nd ACM SIGCOMM workshop on Information-centric networking (ICN)*, Helsinki, Finland, August 2012, pp. 49–54.

[90] K. Fall, "A Delay-Tolerant Network Architecture for Challenged Internets," in *ACM SIGCOMM*, Karlsruhe, Germany, August 2003, pp. 27–34.

[91] ——, "Comparing Information Centric and Delay Tolerant Networking," http://kfall.net/seipage/talks/lcn-icn-dtn-keynote.pdf, October 2012, keynote at LCN 2012.

[92] S. Farrell, A. Lynch, D. Kutscher, and A. Lindgren. (2012, May) Bundle Protocol Query Extension Block. [Online]. Available: https://tools.ietf.org/html/draft-irtf-dtnrg-bpq-00

[93] R. Flickenger, *Building Wireless Community Networks*. O'Reilly Media, November 2001.

[94] C. Fragouli, J. Widmer, and J. Y. L. Boudec, "On the Benefits of Network Coding for Wireless Applications," in *4th IEEE International Symposium on Modeling and Optimization in Mobile, Ad Hoc and Wireless Networks (WiOpt)*, Boston, MA, USA, April 2006, pp. 1–6.

[95] R. Gadow, "Persistent Caching in Information-Centric Networking," University of Bern," Bachelor Thesis, April 2015.

[96] A. Galati, T. Bourchas, S. Siby, S. Frey, M. Olivares, and S. Mangold, "Mobile-Enabled Delay Tolerant Networking in Rural Developing Regions," in *IEEE Global Humanitarian Technology Conference (GHTC)*, San Jose, CA, USA, October 2014, pp. 699–705.

[97] R. Gallager, "Low-Density Parity-Check Codes," *IRE Transactions on Information Theory*, vol. 8, no. 1, pp. 21–28, January 1962.

[98] C. Ghali, G. Tsudik, E. Uzun, and C. Wood. (2015, December) Living in a PIT-less World: A Case Against Stateful Forwarding in Content-Centric Networking. arXiV:1512.07755. [Online]. Available: http://arxiv.org/pdf/1512.07755v1.pdf

[99] M. C. Gonzalez, C. A. Hidalgo, and L. Barabasi, "Understanding individual human mobility patterns," *Nature*, vol. 453, no. 5, pp. 779–782, 2008.

[100] G. Grassi, D. Pesavento, G. Pau, R. Vuyyuru, R. Wakikawa, and L. Zhang, "VANET via Named Data Networking," in *2nd IEEE Workshop on Name Oriented Mobility (NOM)*, Toronto, ON, Canada, April 2014, pp. 410–415.

[101] G. Grassi, D. Pesavento, G. Pau, L. Zhang, and S. Fdida, "Navigo: Interest Forwarding by Geolocations in Vehicular Named Data Networking," in *IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM)*, Boston, MA, USA, June 2015.

[102] P. Gupta, R. Gray, and R. Kumar, "An experimental scaling law for ad hoc networks," University of Illinois, Tech. Rep., 2001.

[103] E. Guttman, C. Perkins, J. Veizades, and M. Day. (1999, June) Service Location Protocol, Version 2. [Online]. Available: https://tools.ietf.org/html/rfc2608

[104] M. Heissenbüttel, T. Braun, T. Bernoulli, and M. Wälchli, "BLR: beacon-less routing algorithm for mobile ad hoc networks," *Computer Communications*, vol. 27, no. 11, pp. 1076–1086, February 2004.

[105] O. R. Helgason, E. A. Yavuz, S. T. Kouyoumdjieva, L. Pajevic, and G. Karlsson, "A Mobile Peer-to-Peer System for Opportunistic Content-Centric Networking," in *2nd ACM SIGCOMM workshop on Networking, systems, and applications on mobile handhelds (MobiHeld)*, New Delhi, India, August 2010, pp. 21–26.

[106] E. Hemmati and J. Garcia-Luna-Aceves, "A New Approach to Name-Based Link-State Routing for Information-Centric Networks," in *2nd ACM Conference on Information-Centric Networking (ICN)*, San Francisco, CA, USA, September 2015, pp. 29–38.

[107] F. Hermans, E. Ngai, and P. Gunningberg, "Mobile Sources in an Information-Centric Network with Hierarchical Names: An Indirection Approach," in *7th Swedish National Computer Networking Workshop (SNCNW)*, Linköping, Sweden, June 2011.

[108] ——, "Global Source Mobility in the Content-Centric Networking Architecture," in *1st ACM Workshop on Emerging Name-Oriented Mobile Networking Design - Architecture, Algorithms, and Applications (NoM)*, Hilton Head Island, South Carolina, USA, June 2012, pp. 13–18.

[109] A. Hess and J. Ott, "Extrapolating Sparse Large-Scale GPS Traces for Contact Evaluation," in *5th ACM HotPlanet Workshop*, Hong Kong, August 2013, pp. 39–44.

[110] P. Hofmann, C. An, L. Loyola, and I. Aad, "Analysis of UDP, TCP and Voice Performance in IEEE 802.11b Multihop Networks," in *13th European Wireless Conference (EW)*, Paris, France, April 2007.

[111] L. Hongbao, W. Muqing, H. Qian, P. Li, and W. Ning, "A Tracker-based Cache Utilization Strategy in CCN," in *26th IEEE International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC)*, Hong Kong, August 2015, pp. 1442–1446.

[112] M. Hoque, S. O. Amin, A. Alyyan, B. Zhang, L. Zhang, and L. Wang, "NLSR: Named-data Link State Routing Protocol," in *3rd ACM Workshop on Information-Centric Networking (ICN 2013)*, Hong Kong, August 2013, pp. 15–20.

[113] T. Hossmann, D. Schatzmann, P. Carta, and F. Legendre, "Twitter in Disaster Mode: Smart Probing for Opportunistic Peers," in *3rd ACM International workshop on Mobile Opportunistic Networks (MobiOpp)*, Zurich, Switzerland, March 2012, pp. 93–94.

[114] T. Hossmann, T. Spyropoulos, and F. Legendre, "Know Thy Neighbor: Towards Optimal Mapping of Contacts to Social Graphs for DTN Routing," in *IEEE INFOCOM*, San Diego, CA, USA, March 2010, pp. 866–874.

[115] P. Hui, A. Chaintreau, J. Scott, R. Gass, J. Crowcroft, and C. Diot, "Pocket Switched Networks and Human Mobility in Conference Environments," in *ACM SIGCOMM Workshop on Delay Tolerant Networking (WDTN)*, Philadelphia, PA, USA, August 2005, pp. 244–251.

[116] C. Intanagonwiwat, R. Govindan, and D. Estrin, "Directed Diffusion: A Scalable and Robust Communication Paradigm for Sensor Networks," in *6th ACM International Conference on Mobile Computing and Networking (MobiCom)*, Boston, MA, USA, August 2000, pp. 56–67.

[117] V. Jacobson, "Congestion Avoidance and Control," in *SIGCOMM Symposium on Communication Architectures and Protocols*, Stanford, CA, USA, August 1988, pp. 314–329.

[118] V. Jacobson, R. L. Braynard, T. Diebert, P. Mahadevan, M. Mosko, N. Briggs, S. Barber, M. Plass, I. Solis, E. Uzun, B. Lee, M.-W. Jang, D. Byun, D. K. Smetters, and J. D. Thornton, "Custodian-Based Information Sharing," *IEEE Communications Magazine*, vol. 50, no. 7, pp. p. 38–43, July 2012.

[119] V. Jacobson, D. K. Smetters, J. D. Thornton, M. F. Plass, N. H. Briggs, and R. L. Braynard, "Network Named Content," in *5th ACM International Conference on Emerging Networking Experiments and Technologies (CoNEXT)*, Rome, Italy, December 2009, pp. 1–12.

[120] S. Jain, K. Fall, and R. Patra, "Routing in a Delay Tolerant Network," in *ACM SIGCOMM*, Portland, OR, USA, September 2004, pp. 145–158.

[121] P. Jaquet, P. Mühlethaler, T. Clausen, and A. Laouiti, "Optimized Link State Routing Protocol for Ad Hoc Networks," in *IEEE International Multi Topic Conference (INMIC)*, Lahore, Pakistan, December 2001, pp. 62–68.

[122] D. Johnson, Y. Hu, and D. Maltz. (2007, February) RFC 4728: The Dynamic Source Routing Protocol (DSR) for Mobile Ad Hoc Networks for IPv4. [Online]. Available: https://tools.ietf.org/html/rfc4728

[123] D. Johnson and D. Malt, "Dynamic Source Routing in Ad Hoc Wireless Networks," in *Mobile Computing*. Kluwer Academic Publishers, 1996, pp. 153–181.

[124] B. Karp and H. Kung, "GPSR: Greedy Perimeter Stateless Routing for Wireless Networks," in *6th ACM Annual Conference on Mobile Computing and Networking (MobiCom)*, Boston, MA, USA, August 2000, pp. 243–254.

[125] D.-H. Kim, J.-H. Kim, Y.-S. Kim, H. s. Yoon, and I. Yeom, "Mobility Support in Content Centric Networks," in *2nd ACM Workshop on Information-Centric Networking (ICN)*, Helsinki, Finland, August 2012, pp. 13–18.

[126] I. Komnios and V. Tsaoussidis, "CARPOOL: Connectivity Plan Routing Protocol," in *12th International Conference on Wired & Wireless Internet*

Communications (WWIC), ser. LNCS 8458, Paris, France, May 2014, pp. 269–282.

[127] H. Kruse, S. Jero, and S. Ostermann. (2014, March) RFC 7122: Datagram Convergence Layers for the Delay- and Disruption-Tolerant Networking (DTN) Bundle Protocol and Licklider Transmission Protocol (LTP). [Online]. Available: https://tools.ietf.org/html/rfc7122

[128] T. Lauinger, N. Laoutaris, P. Rodriguez, T. Strufe, E. Biersack, and E. Kirda, "Privacy Implications of Ubiquitous Caching in Named Data Networking Architectures," Northeastern University, Tech. Rep. TR-iSecLab-0812-001, 2012.

[129] ——, "Privacy Risks in Named Data Networking: What is the Cost of Performance?" *ACM SIGCOMM Computer Communication Review*, vol. 42, no. 5, pp. 54–57, October 2012.

[130] S. Lederer, C. Mueller, C. Timmerer, and H. Hellwagner, "Adaptive Multimedia Streaming in Information-Centric Networks," *IEEE Network*, vol. 28, no. 6, pp. 91–96, November 2014.

[131] J. Lee, D. Kim, M.-W. Jang, and B.-J. Lee, "Proxy-based Mobility Management Scheme in Mobile Content Centric Networking (CCN) environments," in *IEEE International Conference on Consumer Electronics (ICCE)*, Las Vegas, NV, USA, January 2011, pp. 595–596.

[132] K. Lee, S. Hong, S. J. Kim, I. Rhee, and S. Chong, "SLAW: A New Mobility Model for Human Walks," in *IEEE INFOCOM*, Rio de Janeiro, Brazil, April 2009, pp. 855–863.

[133] M. Lee, K. Cho, K. Park, T. Kwon, and Y. Choi, "SCAN: Scalable Content Routing for Content-Aware Networking," in *IEEE International Conference on Communications (ICC)*, Kyoto, Japan, June 2011, pp. 1–5.

[134] M. Lee, J. Song, K. Cho, S. Pack, T. Kwon, J. Kangasharju, and Y. Choi, "Content Discovery for Information-centric Networking," *Computer Networks*, vol. 83, pp. 1–14, 2014.

[135] U. Lee, I. Rimac, and V. Hilt, "Greening the Internet with Content-Centric Networking," in *1st International Conference on Energy-Efficient Computing and Networking (e-Energy)*, Passau, Germany, April 2010, pp. 179–182.

[136] V. Lenders, G. Karlsson, and M. May, "Wireless Ad Hoc Podcasting," in *4th Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks (SECON)*, San Diego, CA, USA, June 2007, pp. 273–283.

[137] Y. Li, T. Lin, H. Tang, and P. Sun, "A Chunk Caching Location and Searching Scheme in Content Centric Networking," in *IEEE International Conference on Communications (ICC)*, Ottawa, Canada, June 2012, pp. 2655–2659.

[138] A. Lindgren, A. Doria, and O. Schelen, "Probabilistic Routing in Intermittently Connected Networks," *ACM SIGMOBILE Mobile Computing and Communications Review*, vol. 7, no. 3, pp. 19–20, 2003.

[139] M. Liu, Y. Yang, and Z. Qin, "A Survey of Routing Protocols and Simulations in Delay-Tolerant Networks," *Wireless Algorithms, Systems, and Applications*, vol. 6843, pp. 243–253, 2011.

[140] J. Llorea, A. Tulino, K. Guan, and D. Kilper, "Network-Coded Caching-Aided Multicast for Efficient Content Delivery," in *IEEE International Conference on Communications (ICC)*, Budapest, Hungary, June 2013, pp. 3557–3562.

[141] P. Mahadevan, "CCNx 1.0 Tutorial," PARC, Tech. Rep., March 2014.

[142] L. Mamatas, A. Papadopoulou, and V. Tsaoussidis, "Exploiting Communication Opportunities in Disrupted Network Environments," in *13th International Conference on Wired & Wireless Internet Communications*, ser. LNCS 9071, Malaga, Spain, May 2015, pp. 180–193.

[143] M. Meisel, V. Pappas, and L. Zhang, "Ad hoc networking via named data," in *5th ACM International Workshop on Mobility in the Evolving Internet Architecture (MobiArch)*, Chicago, USA, September 2010, pp. 3–8.

[144] ——, "Listen First, Broadcast Later: Topology-Agnostic Forwarding under High Dynamics," in *4th Annual Conference of the International Technology Alliance (ACITA)*, London, UK, September 2010, pp. 1–8.

[145] A. G. Miklas, K. K. Gollu, K. Chan, S. Saroiu, K. P. Gummadi, and E. de Lara, "Exploiting Social Interactions in Mobile Systems," in *9th International Conference on Ubiquitous Computing (UbiComp)*, Innsbruck, Austria, September 2007, pp. 409–428.

[146] E. Monticelli, B. M. Schubert, M. Arumaithurai, X. Fu, and K. K. Ramakrishnan, "An Information Centric Approach for Communications in Disaster Situations," in *20th International workshop on Local & Metropolitan Area Networks (LANMAN)*, Reno, NV, May 2014, pp. 1–6.

[147] M. Montpetit, C. Westphal, and D. Trossen, "Network Coding Meets Information-Centric Networking," in *1st ACM workshop on Emerging Name-Oriented Mobile Networking Design - Architecture, Algorithms, and Applications (NoM)*, Hilton Head, SC, USA, June 2012, pp. 31–36.

[148] W. Moreira and P. Mendes, "Pervasive Data Sharing as an Enabler for Mobile Citizen Sensing Systems," *IEEE Communications Magazine*, vol. 53, no. 10, pp. 164–170, October 2015.

[149] M. Mosko. (2015, January) CCNx Messages in TLV Format. Internet Draft. [Online]. Available: http://tools.ietf.org/html/draft-mosko-icnrg-ccnxmessages-00

[150] M. Mosko and I. Solis. (2015, June) CCNx Semantics. [Online]. Available: https://tools.ietf.org/html/draft-irtf-icnrg-ccnxsemantics-00

[151] N. Mujkanovic, "Synchome - Synchronization Application for Mobile Content Retrieval," University of Bern," Bachelor Thesis, March 2015.

[152] A. Munjal, T. Camp, and W. C. Navidi, "SMOOTH: A Simple Way to Model Human Mobility," in *14th ACM international conference on Modeling, analysis and simulation of wireless and mobile systems (MSWiM)*, Miami Beach, FL, USA, November 2011, pp. 351–360.

[153] N. Orlinski and N. Filer, "Neighbor discovery in opportunistic networks," *Ad Hoc Networks*, vol. 25, pp. 383–392, 2015.

[154] S. Oueslati, J. Roberts, and N. Sbihi, "Flow-aware traffic control for a content-centric network," in *IEEE INFOCOM*, Orlando, FL, USA, March 2012, pp. 2417–2425.

[155] L. Pelusi, A. Passarella, and M. Conti, "Opportunistic Networking: Data Forwarding in Disconnected Mobile Ad Hoc Networks," *IEEE Communications Magazine*, vol. 44, no. 11, pp. 134–141, November 2006.

[156] D. Perino and M. Varvello, "A Reality Check for Content Centric Networking," in *ACM SIGCOMM workshop on information-centric networking (ICN)*, Toronto, Canada, August 2011, pp. 44–49.

[157] D. Perino, M. Varvello, and K. Puttaswamy, "ICN-RE: Redundancy Elimination for Information-Centric Networking," in *2nd ACM Workshop on Information-Centric Networking (ICN)*, Helsinki, Finland, August 2012, pp. 91–96.

[158] C. Perkins, S. Ratliff, J. Dowdell, L. Steenbrink, and V. Mercieca. (2015, July) Ad Hoc On-demand Distance Vector (AODVv2) Routing. [Online]. Available: https://tools.ietf.org/html/draft-ietf-manet-aodvv2-11

[159] C. E. Perkins and E. M. Royer, "Ad-hoc On-Demand Distance Vector Routing," in *2nd IEEE Workshop on Mobile Computing Systems and Applications (WMCSA)*, New Orleans, LA, USA, February 1999, pp. 90–100.

[160] M. Pitkänen, T. Kärkkäinen, and J. Ott, "Mobility and Service Discovery in Opportunistic Networks," in *IEEE PerCom International Workshop on the Impact of Human Mobility on Pervasive Systems and Applications (PerMoby)*, Lugano, Switzerland, March 2012, pp. 204–210.

[161] M. Pitkänen, T. Karkkainen, J. Ott, M. Conti, A. Passarella, S. Giordano, D. Puccinelli, F. Legendre, S. Trifunovic, K. Hummel, M. May, N. Hegde, and T. Spyropoulos, "SCAMPI: Service platform for soCial Aware Mobile and Pervasive computIng," in *1st Mobile Cloud Computing (MCC) workshop*, Helsinki, Finland, August 2012, pp. 7–12.

[162] M. Plass and N. Briggs. (2012, January) CCNCAT - Manual pages. [Online]. Available: http://www.ccnx.org/releases/ccnx-0.8.2/doc/manpages/ccncat.1.html

[163] M. Plass. (2013, March) CCND Manual Page. [Online]. Available: http://www.ccnx.org/releases/ccnx-0.8.2/doc/manpages/ccnd.1.html

[164] S. Podlipnig and L. Böszörmenyi, "A survey of Web cache replacement strategies," *ACM Computing Surveys (CSUR)*, vol. 35, no. 4, pp. 374–398, 2003.

[165] D. Posch, C. Kreuzberger, B. Rainer, and H. Hellwagner, "Using In-Network Adaptation to Tackle Inefficiencies Caused by DASH in Information-Centric Networks," in *Workshop on Design, Quality and Deployment of Adaptive Video Streaming (VideoNext)*, Sydney, Australia, December 2014, pp. 25–30.

[166] I. Psaras, W. Chai, and G. Pavlou, "Probabilistic In-Network Caching for Information-Centric Networks," in *2nd ACM SIGCOMM workshop on Information-centric networking (ICN)*, Helsinki, Finland, August 2012, pp. 50–60.

[167] I. Psaras, R. Clegg, R. Landa, W. Chai, and G. Pavlou, "Modelling and evaluation of CCN-caching trees," in *10th IFIP NETWORKING*, Valencia, Spain, May 2011, pp. 78–91.

[168] R. Ravindran, S. Lo, X. Zhang, and G. Wang, "Supporting Seamless Mobility in Named Data Networking," in *IEEE International Conference on Communications (ICC)*, Ottawa, ON, Canada, June 2012, pp. 5854–5869.

[169] T. Rodrigues, F. Benevenuto, M. Cha, K. P. Gummadi, and V. Almeida, "On Word-of-Mouth Based Discovery of the Web," in *ACM SIGCOMM Internet Measurement Conference (IMC)*, Berlin, Germany, November 2011, pp. 381–396.

[170] D. Rossi and G. Rossini, "Caching performance of content centric networks under multi-path routing (and more)," Telecom ParisTech, Tech. Rep., 2011.

[171] L. Saino, C. Cocora, and G. Pavlou, "CCTCP: A Scalable Receiver-driven Congestion Control Protocol for Content-Centric Networking," in *IEEE International Conference on Communications (ICC)*, Budapest, Hungary, June 2013, pp. 3775–3780.

[172] J. Saltarin, E. Bourtsoulatze, N. Thomos, and T. Braun, "NetCodCCN: a Network Coding approach for Content-Centric Networks," in *IEEE International Conference on Computer Communications (INFOCOM)*, 2016.

[173] N. Sastry, D. Manjunath, K. Sollins, and J. Crowcroft, "Data Delivery Properties of Human Contact Networks," *IEEE Transactions on Mobile Computing*, vol. 10, no. 6, pp. 868–880, June 2011.

[174] A. Sathiaseelan, L. Wang, A. Aucinas, G. Tyson, and J. Crowcroft, "SCANDEX: Service Centric Networking for Challenged Decentralised Networks," in *ACM MobiSys Workshop on Do-it yourself Networking: an interdisciplinary Approach*, Florence, Italy, May 2015, pp. 15–20.

[175] S. Scellato, C. Mascolo, M. Musolesi, and J. Crowcroft, "Track Globally, Deliver Locally: Improving Content Delivery Networks by Tracking Geographic Social Cascades," in *20th World Wide Web Conference (WWW)*, Hyderabad, India, April 2011, pp. 457–466.

[176] T. Schmid, "Data Exchange in Intermittently Connected Content-Centric Networks," University of Bern," Bachelor Thesis, March 2013.

[177] ——, "Agent-based Data Retrieval for Opportunistic Content-Centric Networks," University of Bern," Master Thesis, August 2015.

[178] G. Scott, "CCNx 1.0 Simple Service Discovery," PARC, Tech. Rep., 2014.

[179] K. Scott and S. Burleigh. (2007, November) RFC 5050: Bundle Protocol Specification. [Online]. Available: http://tools.ietf.org/html/rfc5050

[180] S. Sen, J. Rexford, and D. Towsley, "Proxy Prefix Caching for Multimedia Streams," in *18th INFOCOM*, New York, NY, USA, March 1999, pp. 1310–1319.

[181] A. Shokrollahi, "Raptor Codes," *IEEE Transactions on Information Theory*, vol. 52, no. 6, pp. 2551–2567, June 2006.

[182] A. Sittampalam, "Content Discovery in Wireless Information-Centric Networks," University of Bern," Bachelor Thesis, May 2015.

[183] D. Smetters. (2012, September) CCNGETFILE - Manual pages. [Online]. Available: http://www.ccnx.org/releases/ccnx-0.8.2/doc/manpages/ccngetfile.1.html

[184] ——. (2012, September) CCNPUTFILE - Manual pages. [Online]. Available: http://www.ccnx.org/releases/ccnx-0.8.2/doc/manpages/ccnputfile.1.html

[185] A. Socievole, F. D. Rango, and S. Marano, "Face-to-face with Facebook Friends: Using Online Friendlists for Routing in Opportunistic Networks," in *24th IEEE International Symposium on Personal, Indoor and Mobile Radio Communications: Mobile and Wireless Networks (PIMRC)*, London, UK, September 2013, pp. 2989–2994.

[186] T. Spyropoulos, K. Psounis, and C. S. Raghavendra, "Spray and Wait: An Efficient Routing Scheme for Intermittently Connected Mobile Networks," in *ACM SIGCOMM workshop on Delay Tolerant Networks (WDTN)*, Philadelphia, PA, USA, August 2005, pp. 252–259.

[187] T. Staub, R. Gantenbein, and T. Braun, "VirtualMesh: an emulation framework for wireless mesh and ad hoc networks in OMNeT++," *SIMULATION*, vol. 87, no. 1-2, pp. 66 – 81, January 2011.

[188] T. Staub, S. Morgenthaler, D. Balsiger, P. Goode, and T. Braun, "ADAM: Administration and Deployment of Adhoc Mesh Networks," in *3rd IEEE Workshop on Hot Topics in Mesh Networking (HotMESH)*, Lucca, Italy, June 2011, pp. p. 1–6.

[189] J. Sterbenz, D. Hutchison, E. Cetinkaya, A. Jabbar, J. Rohrer, M. Schöller, and P. Smith, "Resilience and survivability in communication networks: Strategies, principles and survey of disciplines," *Computer Networks*, vol. 54, no. 8, pp. 1245–1265, 2010.

[190] A. Striffeler, "Raptor Coding in Mobile Content-Centric Networks," University of Bern," Bachelor Thesis, August 2014.

[191] J. Su, J. Scott, P. Hui, J. Crowcroft, E. D. Lara, C. Diot, A. Goel, M. H. Lim, and E. Upton, "Haggle: seamless networking for mobile applications," in *9th International Conference on Ubiquitous Computing (UbiComp)*, Innsbruck, Austria, September 2007, pp. p. 391–408.

[192] M. Tarique, K. E. Tepe, S. Adibi, and S. Erfani, "Survey of multipath routing protocols for mobile ad hoc networks," *Network and Computer Applications*, vol. 32, no. 6, pp. 1125–1143, November 2009.

[193] N. Thomos and P. Frossard, "Network Coding of Rateless Video in Streaming Overlays," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 20, no. 12, pp. 1834–1847, December 2010.

[194] S. Trifunovic, M. Kurant, K. A. Hummel, and F. Legendre, "WLAN-Opp: Ad-hoc-less Opportunistic Networking on Smartphones," *Ad Hoc Networks*, vol. 25, pp. 346–358, 2015.

[195] C. Tsilopoulos, G. Xylomenos, and G. C. Polyzos, "Are Information-Centric Networks Video-Ready?" in *20th International Packet Video Workshop*, San Jose, CA, USA, December 2013, pp. 1–8.

[196] TubeMogul. (2009, May) Half Of YouTube Videos Get Fewer Than 500 Views. [Online]. Available: http://www.businessinsider.com/chart-of-the-day-youtube-videos-by-views-2009-5?IR=T

[197] G. Tyson, J. Bigham, and E. Bodanese, "Towards an Information-Centric Delay-Tolerant Network," in *2nd IEEE International Workshop on Emerging Design Choices in Name-Oriented Networking (NOMEN)*, Turin, Italy, April 2013, pp. 387–392.

[198] G. Tyson, E. Bodanese, J. Bigham, and A. Mauthe, "Beyond Content Delivery: Can ICNs Help Emergency Scenarios?" *IEEE Network*, vol. 28, no. 3, pp. 44–49, June 2014.

[199] G. Tyson, N. Sastry, R. Cuevas, I. Rimac, and A. Mauthe, "A Survey of Mobility in Information-Centric Networks," *ACM Communications Review*, vol. 56, no. 12, pp. 90–98, December 2013.

[200] Ubelix. (2015, May) Linux Cluster of University of Bern. [Online]. Available: http://www.ubelix.unibe.ch

[201] A. Uruqi, "Content Discovery and Retrieval Application forMobile Content-Centric Networks," University of Bern," Bachelor Thesis, April 2014.

[202] G. T. . V11.1.0. (2012, June) Technical Specification Group Services and System Aspects; Multimedia Broadcast/Multicast Service; Protocols and Codecs. [Online]. Available: http://www.3gpp.org/DynaReport/26346.htm

[203] A. Vahdat and D. Becker, "Epidemic routing for partially connected ad hoc networks," Duke University, Tech. Rep. CS-200006, 2000.

[204] D. Vardalis and V. Tsaoussidis, "Exploiting the potential of DTN for energy-efficient internetworking," *Systems and Software*, vol. 90, pp. 91–103, April 2014.

[205] A. Varga, "The OMNeT++ Discrete Event Simulation," in *European Simulation Multiconference (ESM)*, Prague, Czech Republic, June 2001.

[206] M. Varvello, I. Rimac, U. Lee, L. Greenwald, and V. Hilt, "On the Design of Content-Centric MANETs," in *8th International Conference on Wireless On-Demand Network Systems and Services (WONS)*, Bardonecchia, Italy, January 2011, pp. 1–8.

[207] C. Ververidis and G. Polyzos, "Service Discovery for Mobile Ad hoc Networks: A Survey of Issues and Techniques," *IEEE Communications Surveys & Tutorials*, vol. 10, no. 3, pp. 30–45, September 2008.

[208] A. Vetro, "The MPEG-DASH Standard for Multimedia Streaming Over the Internet," *IEEE MultiMedia*, vol. 18, no. 4, pp. 62–67, November 2011.

[209] M. Virgilio, G. Marchetto, and R. Sisto, "PIT Overload Analysis in Content-Centric Networks," in *3rd ACM Workshop on Information-Centric Networking (ICN)*, August, Hong Kong 2013, pp. 67–72.

[210] L. von Rotz, "Adaptive Interest Lifetime for Content-Centric Requests," University of Bern," Bachelor Thesis, February 2015.

[211] B. Wang, S. Sen, M. Adler, and D. Towsley, "Optimal proxy cache allocation for efficient streaming media distribution," in *IEEE INFOCOM*, New York, NY, USA, June 2002, pp. 1726–1735.

[212] J. Wang, J. Ren, K. Lu, J. Wang, S. Liu, and C. Westphal, "An Optimal Cache Management Framework for Information-Centric Networks with Network Coding," in *13th IFIP/IEEE Networking*, Trondheim, Norway, June 2014, pp. 1–9.

[213] J. Wang, R. Wakikawa, and L. Zhang, "DMND: Collecting Data from Mobiles Using Named Data," in *2nd IEEE Vehicular Networking Conference (VNC)*, Jersey City, NJ, USA, December 2010, pp. 49–56.

[214] L. Wang, A. Afanasyev, R. Kuntz, R. Vuyyuru, R. Wakikawa, and L. Zhang, "Rapid Traffic Information Dissemination Using Named Data," in *1st ACM workshop on Emerging Name-Oriented Mobile Networking Design - Architecture, Algorithms, and Applications*, Hilton Head Island, SC, USA, June 2012, pp. 7–12.

[215] L. Wang, S. Bayhan, J. Ott, J. Kangasharju, A. Sathiaseelan, and J. Crowcroft, "Pro-Diluvian: Understanding Scoped-Flooding for Content Discovery in Information-Centric Networking," in *2nd ACM Conference on Information-Centric Networking (ICN)*, San Francisco, CA, USA, September 2015, pp. 9–18.

[216] L. Wang, M. Hoque, C. Yi, A. Alyyan, and B. Zhang, "OSPFN: An OSPF Based Routing Protocol for Named Data Networking," NDN Technical Report NDN-0003, Tech. Rep., July 2012.

[217] L. Wang, R. Wakikawa, R. Kuntz, R. Vuyyuru, and L. Zhang, "Data naming in Vehicle-to-Vehicle communications," in *1st IEEE Workshop on Emerging Design Choices in Name-Oriented Networking (NOMEN)*, Orlando, FL, USA, March 2012, pp. 328–333.

[218] X. Wang, M. Chen, Z. Han, D. O. Wu, and T. Kwon, "TOSS: Traffic Offloading by Social Network Service-Based Opportunistic Sharing in Mobile Social Networks," in *IEEE INFOCOM*, May 2014, pp. 2346–2354.

[219] Y. Wang, K. Lee, B. Venkataraman, R. Shamanna, I. Rhee, and S. Yang, "Advertising Cached Contents in the Control Plane: Necessity and Feasibility," in *1st IEEE Workshop on Emerging Design Choices in Name-Oriented Networking (NOMEN)*, Orlando, FL, USA, March 2012, pp. 286–291.

[220] J. Weber, "Automatic Detection of Forwarding Opportunities in Intermittently Connected Content-Centric Networks," University of Bern," Bachelor Thesis, March 2013.

[221] ——, "Dynamic Adaptation of Transmission Modes for Opportunistic Content-Centric Networks," University of Bern," Master Thesis, August 2015.

[222] A. Williams, M. Arlitt, C. Williamson, and K. Baker, "Web Workload Characterization: Ten Years Later," *Web Content Delivery*, vol. 2, no. 1, pp. 3–21, 2005.

[223] M. P. Wittie, V. Pejovic, L. Deek, K. C. Almeroth, and B. Y. Zhao, "Exploiting Locality of Interest in Online Social Networks," in *6th ACM International Conference on emerging Networking EXperiments and Technologies (CoNEXT)*, December 2010.

[224] K. L. Wu, P. S. Yu, and J. L. Wolf, "Segment-Based Proxy Caching of Multimedia Streams," in *10th international conference on World Wide Web (WWW)*, Hong Kong, May 2001, pp. 36–44.

[225] Q. Wu, Z. Li, and G. Xie, "CodingCache: Multipath-aware CCN Cache with Network Coding," in *3rd ACM SIGCOMM Workshop on Information-Centric Networking (ICN)*, Hong Kong, China, August 2013, pp. 41–42.

[226] G. Xylomenos, C. N. Ververidis, V. Siris, N. Fotiou, C. Tsilopoulos, X. Vasilakos, K. V. Katsaros, and G. C. Polyzos, "A Survey of Information-Centric Networking Research," *IEEE Communications Surveys & Tutorials*, vol. 16, no. 2, pp. 1024–1049, July 2014.

[227] C. Yi, J. Abraham, A. Afanasyev, L. Wang, B. Zhang, and L. Zhang, "On the Role of Routing in Named Data Networking," in *1st ACM International Conference on Information-Centric Networking (ICN)*, Paris, France, September 2014, pp. 27–36.

[228] C. Yi, A. Afanasyev, L. Wang, B. Zhang, and L. Zhang, "Adaptive Forwarding in Named Data Networking," *ACM SIGCOMM Computer Communication Review*, vol. 42, no. 3, pp. 62–67, July 2012.

[229] T. Yu, C. Tian, H. Jiang, and W. Liu, "Measurements and Analysis of an Unconstrained User Generated Content System," in *IEEE International Conference on Communications (ICC)*, Kyoto, Japan, June 2011, pp. 1–5.

[230] Y. Yu, A. Afanasyev, D. Clark, K. Claffy, V. Jacobson, and L. Zhang, "Schematizing Trust in Named Data Networking," in *2nd ACM Conference on Information-Centric Networking (ICN)*, San Francisco, CA, USA, September 2015, pp. 177–186.

[231] F. Zhang, Y. Zhang, A. Reznik, H. Liu, C. Qian, and C. Xu, "A Transport Protocol for Content-Centric Networking with Explicit Congestion Control," in *23rd International Conference on Computer Communications and Networks (ICCCN)*, Shanghai, China, August 2014, pp. 1–8.

[232] W. Zhao, Y. C. M. Ammar, M. Corner, B. Levine, and E. Zegura, "Capacity Enhancement using Throwboxes in DTNs," in *IEEE International Conference on Mobile Adhoc and Sensor Systems (MASS)*, Vancouver, BC, Canada, October 2006, pp. 31–40.

[233] Y. Zhu, B. Xu, X. Shi, and Y. Wang, "A Survey of Social-Based Routing in Delay-Tolerant Networks: Positive and Negative Social Effects," *IEEE Communication Surveys & Tutorials*, vol. 15, no. 1, pp. 387–401, 2013.

[234] Z. Zhu, S. Wang, X. Yang, V. Jacobson, and L. Zhang, "ACT: Audio Conference Tool Over Named Data Networking," in *ACM SIGCOMM 2011 Workshop on Information-Centric Networks (ICN)*, Toronto, ON, Canada, August 2011, pp. 68–73.

[235] M. Zink, K. Suh, Y. Gu, and J. Kurose, "Characteristics of YouTube network traffic at a campus network Measurements, models, and implications," *Computer Networks: The International Journal of Computer and Telecommunications Networking*, vol. 53, no. 4, pp. 501–514, March 2009.

List of Publications

Refereed Papers (Journals, Conferences, Workshops)

- C. Anastasiades, J. Weber, and T. Braun, "Dynamic Unicast: Information-Centric Multi-hop Routing for Mobile Ad-hoc Networks", accepted for publication in *Computer Networks*, 2016, DOI: 10.1016/j.comnet.2016.03.009

- C. Anastasiades, T. Schmid, J. Weber, and T. Braun, "Information-Centric Content Retrieval for Delay-Tolerant Networks", accepted for publication in *Computer Networks*, 2016, DOI: 10.1016/j.comnet.2016.03.006

- C. Anastasiades, A. Sittampalam, and T. Braun, "Content Discovery in Wireless Information-Centric Networks", in *40th IEEE Conference on Local Computer Networks (LCN)*, Clearwater Beach, FL, USA, October, 2015, pp. 28-36, DOI: 10.1109/LCN.2015.7366280

- C. Anastasiades, A. Gomes, R. Gadow, and T. Braun, "Persistent Caching in Information-Centric Networks", in *40th IEEE Conference on Local Computer Networks (LCN)*, Clearwater Beach, FL, USA, October, 2015, pp. 64-72, DOI: 10.1109/LCN.2015.7366284

- C. Anastasiades, L. von Rotz, and T. Braun, "Adaptive Interest Lifetimes for Information-Centric Wireless Multi-hop Communication", in *8th IFIP Wireless and Mobile Networking Conference (WMNC)*, Munich, Germany, October, 2015, pp. 40-47, DOI: 10.1109/WMNC.2015.38

- C. Anastasiades, N. Thomos, A. Striffeler, and T. Braun, "RC-NDN: Raptor Codes Enabled Named Data Networking", in *IEEE International Conference on Communications (ICC)*, London, UK, June, 2015, pp. 3026-3032, DOI: 10.1109/ICC.2015.7248788

- C. Anastasiades, and T. Braun, "Towards Information-Centric Wireless Multi-hop Communication", in *13th International Conference on Wired & Wireless Internet Communications (WWIC)*, Malaga, Spain, May, 2015, LNCS 9071, pp. 367-380, DOI: 10.1007/978-3-319-22572-2_27

- C. Anastasiades, and T. Braun, "Dynamic Transmission Modes to Support Opportunistic Information-Centric Networks", in *International Conference*

on Networked Systems (NetSys), Cottbus, Germany, March, 2015, pp. 1-5, DOI: 10.1109/NetSys.2015.7089077

- C. Anastasiades, W. El Maudni El Alami, and T. Braun, "Agent-based Content Retrieval for Opportunistic Content-Centric Networks", in *12th International Conference on Wired & Wireless Internet Communications (WWIC)*, Paris, France, May, 2014, LNCS 8458, pp. 175-188, DOI: 10.1007/978-3-319-13174-0_14

- D. Mansour, T. Braun, and C. Anastasiades, "NextServe Framework: Supporting Services Over Content-Centric Networking", in *12th International Conference on Wired & Wireless Internet Communications (WWIC)*, Paris, France, May, 2014, LNCS 8458, pp. 189-199, DOI: 10.1007/978-3-319-13174-0_15

- C. Anastasiades, T. Schmid, J. Weber, and T. Braun, "Opportunistic Content-Centric Data Transmission During Short Network Contacts", in *IEEE Wireless Communications and Networking Conference (WCNC)*, Istanbul, Turkey, April, 2014, pp. 2516-2521, DOI: 10.1109/WCNC.2014.6952784

- C. Anastasiades, A. Uruqi, and T. Braun, "Content Discovery in Opportunistic Content-Centric Networks", in *5th International Workshop on Architectures, Services and Applications for the Next Generation Internet*, Clearwater Beach, FL, USA, October, 2012, pp. 1044-1052, DOI: 10.1109/LCNW.2012.6424042

- S. Trifunovic, F. Legendre, and C. Anastasiades, "Social Trust in Opportunistic Networks", in *IEEE Infocom workshop on Network Science for Communication Networks (NetSciCom)*, San Diego, CA, USA, March, 2010, pp. 1-6, DOI: 10.1109/INFCOMW.2010.5466696

- B. Distl, G. Csucs, S. Trifunovic, F. Legendre, and C. Anastasiades, "Extending the Reach of Online Social Networks to Opportunistic Networks with PodNet", Demo Session of ACM MobiOpp, pp. 179-181, Pisa, Italy, February 2010, DOI: 10.1145/1755743.1755779

Bookchapters

- C. Anastasiades, T. Braun, and V. Siris, "Information-Centric Networking in Mobile and Opportunistic Networks", I. Ganchev, M. Curado, A. Kassler (eds.), *Wireless Networking for Moving Objects. Protocols, Architectures, Tools, Services and Applications*, Lecture Notes in Computer Science vol. 8611, pp. 14-30, 2014, DOI: 10.1007/978-3-319-10834-6_2

Unrefereed Papers (Technical Reports, White Papers, Demonstrations)

- C. Anastasiades, A. Gomes, R. Gadow, and T. Braun, "Persistent Caching in Information-Centric Networks", Universität Bern, Institut für Informatik und angewandte Mathematik, Bern, Switzerland, May 22, 2015, IAM-15-001.

- V. Siris, C. Boldrini, R. Bruno, M. Conti, C. Anastasiades, T. Braun, M. Curado, D. Palma, P. Mendes, B. Batista, I. Zarko, and K. Pripuzic, "Content- Centric Architectures for Moving Object," COST Action IC0906 white paper, June 6, 2012

- C. Anastasiades, A. Uruqi, T. Braun, "Evaluation of CCNx in Mobile Environments using VirtualMesh", Presentation and Poster at CCNx Community Meeting, Palo Alto, CA, USA, September 9, 2011,
  Link: `http://www.ccnx.org/events/ccnxcon-2011/`

- S. Trifunovic, C. Anastasiades, B. Distl, and F. Legendre, "PodNetSec - Secure Opportunistic Content Dissemination", Demo Session at ACM Mobisys, San Francisco, CA, USA, June 2010

Supervised Student Theses

- T. Schmid, "Agent-based Data Retrieval for Opportunistic Content-Centric Networks", Master Thesis, University of Bern, August 2015

- J. Weber, "Dynamic Adaptation of Transmission Modes for Opportunistic Content-Centric Networks", Master Thesis, University of Bern, August 2015

- A. Sittampalam, "Content Discovery in Wireless Information-Centric Networks", Bachelor Thesis, University of Bern, May 2015

- R. Gadow, "Persistent Caching in Information-Centric Networking", Bachelor Thesis, University of Bern, April 2015

- N. Mujkanovic, "Synchome - Synchronization Application for Mobile Content Retrieval", Bachelor Thesis, University of Bern, March 2015

- L. von Rotz, "Adaptive Interest Lifetime for Content-Centric Requests", Bachelor Thesis, University of Bern, February 2015

- A. Striffeler, "Raptor Coding in Mobile Content-Centric Networks", Bachelor Thesis, University of Bern, August 2014

- A. Uruqi, "Content Discovery and Retrieval Application for Mobile Content-Centric Networks", Bachelor Thesis, University of Bern, April 2014

- W. El Maudni El Alami, “An Agent-based Content-Centric Networking Application for Data Retrieval”, Master Thesis, University of Bern, September 2013
- J. Weber, “Automatic Detection of Forwarding Opportunities in Intermittently Connected Content-Centric Networks”, Bachelor Thesis, University of Bern, March 2013
- T. Schmid, “Data Exchange in Intermittently Connected Content-Centric Networks”, Bachelor Thesis, University of Bern, March 2013
- C. Knecht, “Secure Key Distribution in Wireless Sensor Networks (WSNs)”, Bachelor Thesis, University of Bern, December 2010