UNIVERSITY OF NICE - SOPHIA ANTIPOLIS
DOCTORAL SCHOOL STIC
SCIENCES ET TECHNOLOGIES DE L'INFORMATION
ET DE LA COMMUNICATION

P H D T H E S I S

to obtain the title of

PhD of Science

of the University of Nice - Sophia Antipolis
Specialty : COMPUTER SCIENCE

Defended by

Mohamed Karim SBAI

Architecture for content sharing in wireless networks

Thesis Advisors: Chadi BARAKAT & Walid DABBOUS

prepared at INRIA Sophia Antipolis, PLANETE Project-Team

defended on October 1, 2010

Jury :

President :	Jean-Paul RIGAULT	-	University of Nice, Sophia Antipolis, France
Reviewers :	Gunnar KARLSSON	-	KTH, Stockholm, Sweden
	Jerome LACAN	-	ISAE, Toulouse, France
Advisors :	Chadi BARAKAT	-	INRIA, Sophia Antipolis, France
	Walid DABBOUS	-	INRIA, Sophia Antipolis, France
Examinator :	Martin MAY	-	Thomson Labs, Paris, France

Contents

Chapter 1

Introduction

Wireless ad hoc networks and P2P file sharing applications are two emerging technologies based on the same paradigm: the P2P paradigm. This paradigm aims to establish large scale distributed services without the need for any infrastructure. Within this paradigm, users have symmetric roles. The global service is ensured thanks to their collaboration. In the case of a wireless ad hoc network, the network is a set of wireless nodes with no central administration or base station. Nodes in such a network operate both as routers and hosts. Multi-hop routing approaches are used to ensure connection between distant nodes. For P2P file sharing applications, peers collaborate in downloading data and multimedia content. Each peer shares some of its upload capacity by serving other peers. The global capacity of the system grows then exponentially with the number of peers. Gnutella [Gnu 2010] and BitTorrent [Bit 2010b] are two examples of P2P content sharing applications in the Internet.

Both P2P file sharing applications and wireless ad hoc networks are mature fields of research. They have been studied heavily but separately in the literature. Only few works try to study how they perform together (e.g., [Oliviera 2003][Klemm 2003][Das 2004]). These works focus on the content lookup problem in wireless ad hoc networks without studying the efficiency of the content sharing itself. Studying the performance of file sharing applications over wireless ad hoc networks is challenging because of the diverse constraints imposed by the use of wireless channels. Indeed, as nodes are both routers and end-users, the routing overhead must be taken into consideration. Furthermore, the performance of transport protocols such as TCP drops seriously when multi-hop paths are used. That is why current topology-unaware P2P file sharing applications are not expected to perform well when deployed over wireless ad hoc networks.

Designing efficient file sharing solutions for such networks is an important area of research. Indeed, a P2P solution for file sharing has diverse advantages over other data dissemination techniques like multicast in general and this applies to wireless ad hoc networks in particular. For instance, in case of multicast, the construction and update of the virtual topology (tree or mesh) is costly in terms of bandwidth consumption namely in dynamic scenarios. Moreover, the data replication in multicast follows the virtual topology and so nodes like leaves of a tree only receive data and do not spend resources to provide it to other nodes. Thus, no fair cooperation is ensured when using multicast unless constructing a different virtual topology (or tree) per piece of data, which is technically unfeasible.

In this thesis, we start by studying the performance of BitTorrent [Bit 2010b] the well-known content sharing protocol when it is deployed in wireless ad hoc networks. Our main observations were that the current Internet version is not adapted to the constraints and nature of the wireless environment. In fact, the neighbourhood of a peer is selected independently on any information about its localisation in the network. In wireless ad hoc networks, this topology unaware choice of effective peers overwhelms seriously the underlying network. For instance, there will be a lot of routing overhead due to the fact that nodes that are peers act both as routers and end-hosts. So, pieces of the content will transit at layer three without necessarily profiting from them at the application level. As a consequence of this routing overhead, the download time of peers decreases considerably. A first idea we had to avoid this problem was to reduce the scope of neighbourhood of peer to some routing hops. In this way, we could ameliorate significantly the download time namely for very dense P2P networks. However, when considering networks where only few nodes are peers, we noticed that considering a narrow neighbourhood disconnects the sharing overlay. In deed, some nodes are isolated from the rest of the P2P network and never finish the download. Hence, there is a trade-off between diminishing the routing overhead and ensuring the connectivity of the P2P overlay.

Furthermore, studying the performance of BitTorrent in regard to only one metric, the download time, do not cover all its objectives namely the goal of ensuring a fair distribution of the data transfer burden between peers by ensuring fair cooperation. We consider in a second step of our study the sharing opportunities dimension of the problem. Whereas limiting the scope of the neighbourhood diminishes the download times, it decreases dramatically the sharing opportunities of peers. In fact, it engenders a very low diversity of pieces in the network. The pieces propagate in one direction from the initial source of the content to the edge of the network. Hence, peers do not posses original pieces to exchange with others. So, one face again a dilemma between diversifying the content in the network which enquires sending pieces to distant peers and reducing the routing overhead by having topology awareness. However, we show in mobile cases, that the movements of nodes shuffles pieces in the network and the diversity of pieces is obtained by exchanging original pieces when nodes meet. In this particular case, we prove that it is enough to limit the scope of the neighbourhood to have both short download times and high sharing ratios. This study of the performance of BitTorrent is detailed in Chapter 3.

Starting from this preliminary study, we wanted to design our own version of BitTorrent that is adapted to the constraints of wireless ad hoc networks. We tried to solve the different tradeoffs and challenges that came out from the study done on the Internet version of BitTorrent. First, considering the fact that we need to minimize the number of hops between neighboring peers while preserving the connectivity of the communication overlay, we propose to organize peers in a minimum spanning tree that adapts to the topology of the underlying networks. Selecting nodes with this knowledge allowed us to have a minimal routing overhead while connecting isolated nodes to the rest of the P2P network. The obtained results show that the same gain in download time is conserved and at the same time, all peers finish downloading the content. Then, we moved to study how to increase the diversity of pieces in the network without adding an important routing overhead. The idea was to separate the sharing functionality from the diversification functionality. This last effort will be affected only to seeds of the content because lechers have not enough incentives to communicate with far peers. We design a diversification strategy which considers that one of the connections of a seed is used to send pieces to nodes located in an area wider than the restricted sharing area. The scope of the diversification area is adapted to the underlying topology, to the range of pieces, to cross traffic and to the mobility of nodes. Surprisingly, the sharing ratios of peers increase significantly and even shorter download times are recorded. This is mainly due to the fact that piece diversity encourages cooperation between peers and many parallel transfers are possible in the network. The details of our P2P content sharing protocol are described in Chapter 4.

Another important drawback of the Internet version of BitTorrent and similar P2P protocols is that they base the discovery of the members of the overlay on centralised servers. In the case of BitTorrent, the users contact a central rendezvous server in order to obtain a list of peers interested in the content. In a wireless ad hoc network, even if a node devotes its capacity to provide others with membership information, this will never scale because the bandwidth and resources in a wireless network are limited and never guaranteed. Furthermore, wireless ad hoc networks are subject to portioning specially in highly mobile scenarios which can isolate the central server node from the rest of the network. Definitely, a client/server membership management solution is not adapted to wireless ad hoc network. Instead, we propose in this thesis a membership management protocol that offers a distributed and efficient problem for discovering and updating the list of peers in wireless ad hoc networks. The proposed solution is to organize peers in a minimum spanning tree and use it to disseminate information on arrivals and departures of peers. Furthermore, the structure of the tree is adapted to changes in network topology and network partitioning . The same tree is used for selecting the neighborood of a peer in our content sharing protocol. The details of our membership management protocol are presented in Chapter 5.

The performance evaluation of our two components: the membership management protocol and the content sharing protocol is done both through extensive ORBIT experiments and NS-2 simulations. In Chapter 6, we implement BitHoc a real content sharing application for communities of mobile handhelds. The proposed application contains three modules: a content lookup module, a membership management module and a content sharing module. The algorithms and mechanisms of these module are inspired from those did in our study and are evaluated on networks of real mobile devices.

The remainder of this thesis is organized as follows. In chapter 2, we describe the state of the art on P2P networks and wireless ad hoc networks. We also analyse the existing studies on applying P2P applications in the wireless environment. After that, we move to studying the performance of the Internet version of BitTorrent in wireless ad hoc networks in Chapter 3. Then, in Chapter 4, we design our content sharing protocol for wireless multi-hop networks and in Chapter 5, we present our solution for membership management in the wireless ad hoc environment. In Chapter 6, we describe our implementation of a content sharing application (BitHoc) in real ad hoc networks of mobile handhelds. Finally, in Chapter 7, we summarize the work done in this thesis and give some ideas on future directions of research in the field of P2P applications in the wireless environment.

Chapter 2

Background and Motivations

Contents

The Peer-to-peer (P2P) paradigm is nowadays a popular content sharing paradigm in the Internet since it has shown its simplicity and efficiency. In the philosophy of P2P applications, all users are both servers and clients. They offer some of their upload capacities to provide content to other users. The field of P2P networks has attracted a lot of research effort in the last years because it has many interesting applications and it generates a lot of challenging problems. In fact, many mechanisms and algorithms have been proposed in the literature to optimize P2P applications in the Internet. However, the work was focused on Internet P2P networks without a deep study of the possibility of applying the P2P paradigm to wireless networks.

Wireless ad hoc networks share some interesting proprieties with P2P networks. Indeed, nodes in such networks are both routers and end-hosts. They play symmetric roles and offer some of their capacities to forward packets to other nodes. Furthermore, these networks are formed spontaneously and do not require any existing infrastructure. The success of such networks is guaranteed thanks to the collaboration of nodes, mainly at the routing level. In the last years, there has been a big interest in optimizing these networks and creating innovative applications for their users. However, content sharing using the P2P paradigm in wireless ad hoc networks has generally been overlooked in the literature, mainly the data distribution plane.

This chapter describes the background behind our work, namely P2P networks in the Internet and Wireless multi-hop networks. Then, it motivates our study by describing the challenges of P2P content sharing in wireless ad hoc networks and summarizes previous work on this topic.

2.1 P2P networks in the Internet

A Peer-to-Peer (P2P) network can be defined as an application layer overlay network in which all entities are equal and all contribute some of their resources, thus giving rise to a network in which each entity (peer) is both a content requestor and a content provider. Unlike the classical Client/Server model, in the P2P model, the service is not provided by any central dedicated server. The peers are completely autonomous and do not depend on any existing service provider. The global system profits from the resources existing at the edge nodes. As these hosts are heterogeneous and has different levels of collaboration, the global capacity of the system depends on the resources shared by peers. These resources can be added or removed during the session making a P2P network a dynamic and challenging environment.

In the last few years, the Internet community, encouraged by the abundant network resources existing in edge hosts, wanted to get rid of client/server applications and migrate the most important applications to the P2P world. Sample applications of the P2P paradigm are not limited to content sharing applications but contain many others such as instant messaging, voice-over-IP, content localization using Dynamic Hash Tables (DHT), video Streaming, etc. The number of users using P2P applications is growing from one day to another making the P2P traffic dominating the Internet traffic. Indeed, it occupies 50 to 60 percent of the current global Internet traffic [P2P 2007]. This success of P2P systems, explained by their simplicity and their efficiency, is currently at the origin of the great interest the research community is giving to optimizing P2P overlays and inventing new P2P algorithms and applications.

Some of the first application-level protocols, showing the possibility of having a service without a dedicated central node, were the simple Direct Client Connection (DCC) protocol [DCC 2010] and the Client-To-Client protocol (CTPC) [CTC 2010]. These protocols can be used both for instant messaging and file sending and have gained popularity among the Internet Relay Chat (IRC) [IRC 2010] users. For example, DCC enables a simple transmission and negotiation between two peers over the Transport Control Protocol (TCP). More advanced overlay network protocols and technologies for file sharing and instant messaging have been introduced to enable communication and data transfer between several peers. These protocols include BitTorrent [Bharambe 2005] [Bit 2010b], FastTrack (KaZaA) [KaZ 2010], Gnutella [Harjula 2004] [Gnu 2010] and eDonkey/ed2k [eDo 2010] and instant messaging systems like Microsoft messenger [Mic 2010] and ICQ [ICQ 2010]. The list is very long in this chapter we will focus on the most known and relevant ones.

In the following paragraph, we classify the existing P2P systems following many criteria ranging from the nature of the overlay membership management to the structure of the overlay.

2.1.1 Classification of P2P networks

2.1.1.1 Centralized and decentralized P2P networks

When designing a P2P system, one main task, which has to be considered, is how peers discover each other and how they locate the desired contents. A first classification of P2P networks can be derived following whether a central node is used or not for finding peers and contents. In the case an index server is used, the P2P network is called a centralized network. In the opposite case, it is called a decentralized P2P network.

Centralized P2P networks: Searching in a centralized P2P network is an easy task thanks to the existence of a single centralized server, which maintains directories of resources shared in the network [Harjula 2004]. If a user needs to access a specific file or content, it sends a search request to the central server. This centralized server creates a list of matching results for the particular request of the user. The matching mechanism is a simple check of which nodes among those claiming to hold the requested file, are currently connected to the P2P network. The user can choose from this list of nodes those from which it wants to get the requested content.

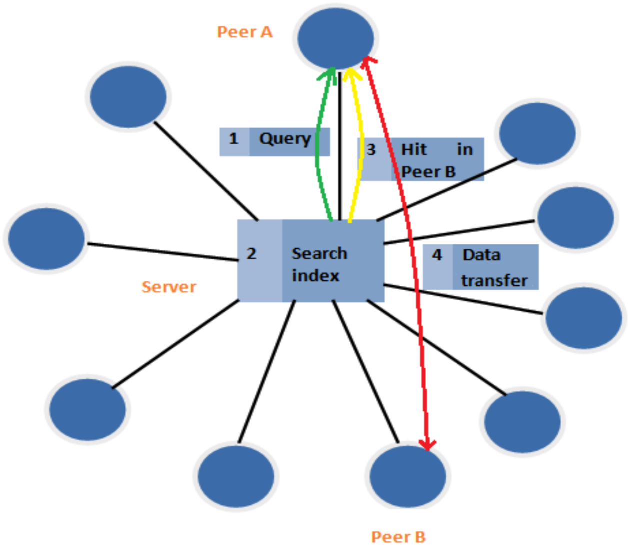


Figure 2.1: Content discovery mechanism of Napster

For instance, Napster [Nap 2010] is based on such a centralized and indexed system [Harjula 2004]. Figure 2.1 shows the operating principles of Napster. The Napster main server maintains matches between identifiers of nodes and files that are stored on these nodes. Another important example of a centralized P2P network is the first versions of BitTorrent [Bit 2010b], where the IP addresses of the peers interested in some content are stored in a central rendezvous server called Tracker. The Tracker periodically provides each peer with a list of peers that are interested in the content and information about pieces of content that are in their possession. We describe later in this section the BitTorrent protocol in details.

Despite of the simplicity of using a central server for searches, the central server is a single point of failure for the system. When this dedicated server is down or is overloaded due to an important amount of requests, the P2P network is totally paralyzed and no new requests can be possible. This means that, unless back-up servers and clusters of servers are deployed, centralized P2P networks scale with a great difficulty. Furthermore, the consistency of information hold by the central server is not always guaranteed since it requires a lot of exchanges between peers and the server. In reality, there is a tradeoff between the consistency and the performance of the global search system. One other important drawback is the vulnerability of the central server to security attacks mainly deny of service attacks. All these drawbacks encouraged the research community to design decentralized P2P systems.

Decentralized P2P networks: Two of the first popular decentralized P2P protocols are FastTrack [Sen 2004] and KaZaA protocol [KaZ 2010], which define two types of clients: nodes and supernodes. The supernode functionality is attributed to clients with for example fast Internet connections and powerful processors. The purpose of a supernode is similar to that of the eDonkey [eDo 2010] server or BitTorrent [Bit 2010b] tracker. It indexes the connections (IP addresses) and the list of contents the client want to share. The supernode also handles the search requests from the other clients and communicates with other supernodes in order to extend the search.

Decentralization consists in relaxing the central structure of a network in such away that each peer can communicate equally to all other nodes in the network. For example, when a peer A wants to send a query request to the network, it announces the fact that it is alive to the whole network. B, a direct neighbor of A, receives this message and announces the fact that A is alive to all its directly connected hosts (C, D, E, and F for instance) and so forth. In their turn, C, D, E, and F repeat this message. As soon as A has announced that it is alive, it can send search requests to the network. In a first step, the query message is directly sent to B, which forwards the message to its neighbors C, D, E, and F. If for example D has the information which is requested by A, it sends a copy of this information back to B which forwards the response to A.

For example, in decentralized Hash Table networks (DHTs), to avoid using central servers, each file stored in the system is given a unique identifier, typically a hash of its content, which is used to identify a resource. Given this unique ID, a resource can be located quickly and almost independently of the size of the network. For instance, CAN [Ratnasamy 2001] is a distributed and decentralized P2P network structure based on hash tables, which is built around a virtual multi-dimensional Cartesian coordinate space in which the entire coordinate space is dynamically partitioned among all the peers in the system such that every peer possesses its individual, distinct zone within the overall space. For routing request messages, all peers maintain routing tables holding virtual coordinates and IP addresses of all neighbors. To find the best routing decision, a simple greedy-algorithm can be used in this structured environment. More details about the mechanisms of CAN are described in 2.1.1.2

2.1.1.2 Structured and unstructured P2P networks

P2P networks can be classified into structured and unstructured networks based on whether content location is based on the knowledge of the overlay topology.

Unstructured P2P networks: In unstructured P2P networks, we neither have a centralized directory nor any precise control over the network topology and the location of files. For example, Gnutella [Gnu 2010] is an unstructured P2P network (see Figure 2.2) where the placement of files cannot be based on any knowledge of the topology as separate couples of nodes form the whole network [Harjula 2004].

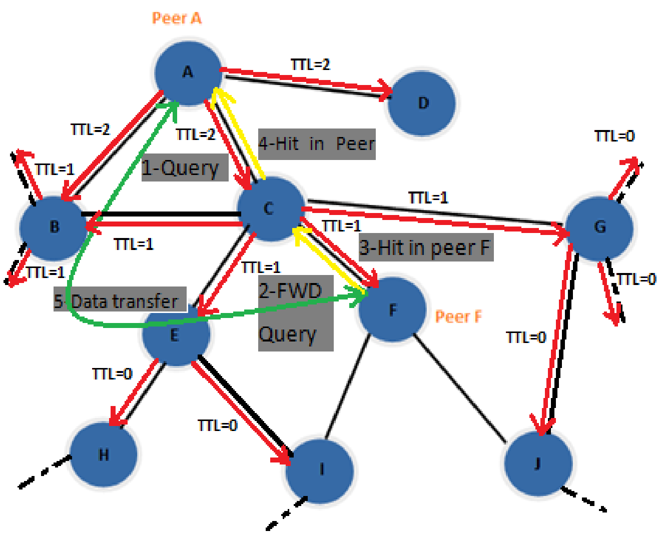


Figure 2.2: Gnutella operations

- **Searching files in unstructured P2P networks:** To find a file in an unstructured P2P network, a node queries all its neighbors by flooding a limited-range query. We illustrate through Figure 2.2 an example of the operating principle of Gnutella in the case of searching files [Harjula 2004]. In the beginning, peer A sends a query message to all its neighbors, B, C and D (1.). If the queried resource is not found, they again broadcast the query to all their neighbors, in C 's case, to E, F, and G (2.). If the resource is again not found, this pattern will continue recursively with each new level of nodes, until the resource is found or TTL reaches zero. In the case of Figure 2.2, the resource is found from F. The latter peer responds to C with a message containing its location information (3.). Then, C delivers this message to A (4.). Finally, the file transfer is made directly between A and F (5.), just like in the Napster's case. However, in other peers, E and G in Figure 2.2, the query is forwarded again. In this case, the search ends in peers H, I, and J because the TTL reaches zero in them.

- **Joining unstructured P2P networks:** An unstructured P2P overlay network forms a random graph, which can be flat or hierarchical such as an architecture containing some super-nodes. We assume in this paragraph that the unstructured P2P overlay network is flat and already created as shown in Figure 2.3 and we describe the operations to be done by a new peer joining the P2P overlay. We take Gnutella operations as an example. First, the new peer must find one of the peers that are already members of the overlay. Second, it simply copies the links of this peer and uses them as its overlay information. In Figure 2.4, the new node detects that node A is the nearest neighbor, so it connects to node it. To join overlay, the peer copies existing overlay link information from node A. Hence, it connects directly to nodes B and C. Generally, having an unstructured P2P overlay, there is no need to exchange signaling messages to maintain the overlay.

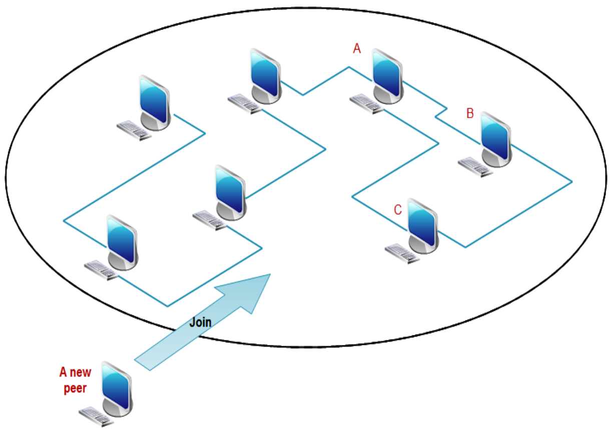


Figure 2.3: A New peer joins the existing overlay network

The bootstrap procedure for finding nodes that are in the overlay can be either one of followings. The new node uses previous list of the overlay peers from the last time. The new node retrieves overlay information from centralized places, such as DNS servers or web sites. The next bootstrap strategy is the use of limited broadcast query. The new node broadcasts a peer discovery message by specifying a value of TTL to limit distance of the query message.

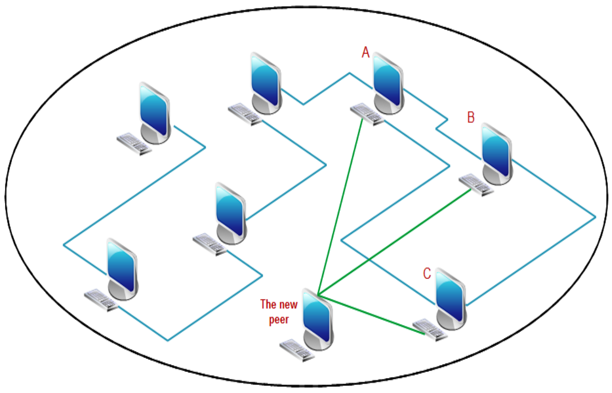


Figure 2.4: The new peer copies overlay link information from peer A

From the previous description, one can notice that the unstructured designs are extremely reliable to nodes entering and leaving the system very frequently. But unfortunately, their current search mechanisms are extremely un-scalable, and are often generating large additional loads on the network.

Structured P2P networks using DHTs: A structured P2P network is based on the fact that the location of an object (resource) can always be determined by a globally agreed-upon scheme, e.g., hashing the resource's key. If one knows the key for a desired object, one can easily find the location where that object should be found. If the desired object exists several times in the network; for instance, the network holds many copies of a given file; the network has to decide which object (file) should be associated with the given key.

In the structured architecture, the nodes and the objects to be stored in them are linked logically. Hence, locating a file in this architecture is much faster than in the unstructured design. Furthermore, any rare content can be easily located using the structured architecture. In unstructured P2P systems such as FastTrack and Gnutella, where the overlay is organized in a random topological manner, locating content may take a very long time as the search is done randomly. However, in structured P2P networks, a particular node is assigned to a particular object using matching functions. One of the concepts used by structured file sharing applications is the one of Distributed Hash Tables (DHTs). In a DHT, when a node wants an object, it contacts the node supposed to have it either directly or through routing in the DHT. Many DHT architectures exist in the literature and share the following basics:

- Each object (content) is assigned an identifier (ID), typically the hash of its name. The hash function used for this purpose can be anyone.
- Nodes are also given identifiers in the same namespace as objects. These IDs are assigned to them randomly as the probability of collision is very low in a large space.
- An object is stored in a specific node, not a random selected one, but the node, which has the closest ID to the ID of the object.
- Each node has a table that includes the information about some other peers like the node ID and the IP address.

Needing to access some object when knowing its hash, a node can read its local table and find the node ID which has a value closest to the ID object. Then, it asks this node for the object and the download can start. In this case, the solution is very fast as it uses only $O(1)$ messages. However, in real world, there can be millions of nodes and hence there are $O(N)$ neighbors. These neighbors join and leave the network during the session. Hence, the nodes in the DHT must be organized in an overlay so that every node knows about some few other nodes that are its neighbors rather than knowing all the nodes. These neighbors are not picked up randomly but are decided by a predefined structure. In this case, when a node needs to query for an object, the queries are routed over the overlay. Whenever a node receives a query for an object with an ID, it forwards to the neighbor whose ID is the closest. The main issue in the DHT is that one should design it so that a node does not have too many neighbors, while not increasing indefinitely the lookup delay.

The DHT architectures can also be further classified into various types according to the algorithms they use to search the responsible nodes and the routing information they store. Some examples of them are explained below:

- **Pastry:** In Pastry [Rowstron 2001b], keys and node IDs are 128 bits in length and can be thought of as a sequence of digits in base 16. A node's routing table has about $O(log_{16}N)$ rows and 16 columns (N is the number of nodes in the overlay). The entries in row n of the routing table refer to nodes whose node IDs share the first n digits with the present node's ID. The $(n+1)$th node ID digit of a node in column m of row n equals m. The column in row n corresponding to the value of the $(n+1)$th digit of the local node's ID remains empty. At each routing step in Pastry, a node normally forwards the message to a node whose ID shares with the key a prefix that is at least one digit longer than the prefix that the key shares with the present node's ID. If no such node is known, the message is forwarded to a node whose ID shares a prefix with the key as long as the current node but is numerically closer to the key than the present node's ID. Each Pastry node maintains a set of neighboring nodes in the node ID space (leaf set), both to locate the destination in the final routing hop, and to store replicas of data items for fault tolerance. The expected number of routing hops is less than $O(log_{16}N)$.

- **Tapestry:** Tapestry [Tap 2001] is very similar to Pastry but differs in its approach to mapping keys to nodes in the sparsely populated ID space, and in how it manages replication. In Tapestry, there is no leaf set and neighboring nodes in the namespace are not aware of each other. When a node's routing table does not have an entry for a node that matches a key's nth digit, the message is forwarded to the node in the routing table with the next higher value in the nth digit modulo $2b$. This procedure, called surrogate routing, maps keys to a unique live node if the node routing tables are consistent. For fault tolerance, Tapestry inserts replicas of data items using different keys. The expected number of routing hops is $O(log_{16}N)$.

- **Chord:** Chord [Stoica 2001] uses a circular 160 bit ID space. Unlike Pastry, Chord forwards messages only clockwise in the circular ID space. Instead of the prefix-based routing table in Pastry, Chord nodes maintain a finger table, consisting of node IDs and IP addresses of up to 160 other live nodes. The ith entry in the finger table of the node with node ID n refers to the live node with the smallest node ID clockwise from $n + 2i - 1$. The first entry points to n's successor and subsequent entries refer to nodes at repeatedly doubling distances from n. Each node also maintains pointers to its predecessor and to its k successors in the ID space (the successor list). Similar to Pastry's leaf set, this successor list is used to replicate objects for fault tolerance. The expected number of routing hops in Chord is $O(\frac{1}{2}log_2N)$.

- **CAN:** CAN [Ratnasamy 2001] routes messages in a d-dimensional space, where each node maintains a routing table with $O(d)$ entries and any node can be reached in $O(dN^{\frac{1}{d}})$ routing hops. The entries in a node's routing table refer to its neighbors in the d-dimensional space. Unlike Pastry, Tapestry and Chord, CAN's routing table does not grow with the network size but the number of routing hops grows faster than $O(logN)$ in this case, namely $O(dN^{\frac{1}{d}})$.

2.1.2 BitTorrent: An efficient content sharing protocol

Unlike DHT-based P2P applications described earlier, BitTorrent does not concentrate on the content localization plane but it goes one step further to optimize the data transfer plane. BitTorrent [Bit 2010b] is a P2P system for efficient and scalable replication of large amounts of static data. In fact, the global capacity of the system increases with the number of downloaders and the protocol utilizes a large amount of available network bandwidth. The file to be distributed is split up in pieces and an SHA-1 [SHA 1995] hash is calculated for each piece. A metadata file (*.torrent*) is distributed to all peers usually via HTTP. The metadata contains: The SHA-1 hashes of all pieces, a mapping of the pieces to files and a Tracker reference. The tracker is a central server keeping a list of all peers participating in the Torrent (swarm). A Torrent is the set of peers that are participating in distributing (sharing) the same file. A peer joins a Torrent by asking the tracker for a peer list and connects to those peers.

2.1.2.1 Goals of BitTorrent

The main goals of BitTorrent are efficiency and reliability.

Efficiency: To increase the speed of downloads; BitTorrent uses the concept of multi-sourcing. Indeed, a peer is able to download pieces of the content from many peers in parallel. Moreover, to increase the interest of peers in each other, the protocol tries to increase the entropy of information in the network. In fact, it minimizes pieces overlap between peers and hence, one obtains a better diversity of content in the network. In this case, a peer can exchange pieces with as many peers as possible. To encourage the collaboration and boost fair sharing among the set of peers, BitTorrent implements some incentive techniques based on the measurements of downloads.

Reliability: To increase its reliability, BitTorrent is tolerant against dropping peers. In fact, each peer that leaves the Torrent means decreased piece availability. Hence, the protocol uses a piece selection strategy that encourages peers to download pieces with least redundancy in the network. In this way, it maximizes piece redundancy and as a result, it increases the number of distributed copies.

2.1.2.2 Terminology of BitTorrent

We will briefly define several terms commonly used in the BitTorrent system.

- **Piece (Chunk):** A torrent is split up into pieces that are of the same size except for the last piece which may be less than the piece size. Typical sizes are 64kB, 128kB, 256kb, 512kB, and 1MB.
- **Block:** A part of a piece that is transferred over the wire at a time. It is typically 16kB.
- **Torrent:** A set of peers communicating together to download the same content.
- **Peer:** An entity in the BitTorrent system that uploads/downloads file pieces.
- **Leecher:** A peer that does not have a full copy of the file and is downloading from/uploading to the network.
- **Seed:** A peer that has a complete copy of the file and is uploading to the network.
- **Neighbors:** The neighbors of a peer are the peers that it has discovered. They are stored in its peer list.

- **Effective neighbors:** The neighbors to whom a peer is sending pieces of the content. In other words, it is the set of peers with whom it has active outgoing TCP data transfer connections.
- **Choke:** A peer chokes another when it stops sending pieces of the content to it.
- **Unchoke:** A peer unchokes another when it starts proposing pieces of content to it.
- **Tracker:** A central entity that peers communicate with periodically to help them discover one another.

2.1.2.3 Main algorithms of BitTorrent

In the following paragraphs, we describe the piece selection and the effective neighbors selection strategies of BitTorrent [Legout 2006][Al-Hamra 2009].

Piece selection strategy: Rarest Piece First The rarest first algorithm works as follows. Each peer maintains a list of the number of copies of each piece in its neighborhood (peer set). It uses this information to define a rarest pieces set. Let m be the number of copies of the rarest piece, then the index of each piece with m copies in the peer set is added to the rarest pieces set. The rarest pieces set of a peer is updated each time a copy of a piece is added to or removed from its peer set. Each peer selects the next piece to download at random in its rarest pieces set. The behavior of the rarest first algorithm can be modified by three additional policies. First, if a peer has downloaded strictly less than 4 pieces, it chooses randomly the next piece to be requested. This is called the random first policy. Once it has downloaded at least 4 pieces, it switches to the rarest first algorithm. The aim of the random first policy is to permit a peer to download its first pieces faster than with the rarest first policy, as it is important to have some pieces to reciprocate for the choke algorithm. Indeed, a piece chosen at random is likely to be more replicated than the rarest pieces, thus its download time will be on average shorter. Second, BitTorrent also applies a strict priority policy, which is at the block level. When at least one block of a piece has been requested, the other blocks of the same piece are requested with the highest priority. The aim of the strict priority policy is to complete the download of a piece as fast as possible. As only complete pieces can be sent, it is important to minimize the number of partially received pieces. Finally, the last policy is the end game mode [Cohen 2003]. This mode starts once a peer has requested all blocks, i.e., all blocks have either been already received or requested. While in this mode, the peer requests all blocks not yet received to all the peers in its peer set that have the corresponding blocks. Each time a block is received, it cancels the request for the received block to all the peers in its peer set that have the corresponding pending request. As a peer has a small buffer of pending requests, all blocks are effectively requested close to the end of the download. Therefore, the end game mode is used at the very end of the download, thus it has little impact on the overall performance.

The objective of the rarest first strategy is to increase the diversity of pieces in the network. In fact, if a peer selects the piece with least redundancy in its neighborhood, it increases the chance of uploading it to other nodes later. If the entropy of information is low, the interest of peers in each other will decrease considerably. Hence, using the rarest first strategy, one minimizes the overlap of pieces among peers and then, boosts reciprocation between them.

Effective neighbors selection strategy: Choke algorithm Among the members of the Torrent, neighbors are those with whom a peer can open a TCP connection to exchange data and information. Only four simultaneous outgoing active TCP connections are allowed by the protocol. These neighbors are called effective neighbors. They are selected according to the choke algorithm of BitTorrent. The choke algorithm was introduced to guarantee a reasonable level of upload and download reciprocation. As a consequence, free riders, i.e., peers that never upload, should be penalized. In the lecher case, this algorithm is executed periodically and aims at identifying the best uploaders. Once the choking period expires, a peer chooses to unchoke the 3 peers uploading to him at the highest rate. This strategy, called tit-for-tat, ensures reciprocity and enforces collaboration among peers. Now to discover new upload capacities, a peer chooses randomly a fourth peer to unchoke.

In the seed case, peers selected as effective neighbors can be chosen following their download capacities in order to fasten the replication of the content or can be selected randomly in order to increase the entropy of pieces in the network. No standard method exists in the literature.

2.1.2.4 Trackerless BitTorrent

Some new versions of BitTorrent want to get rid of the centralized Tracker because it is a single point of failure and that may be the bottleneck of the system. They aim at decentralize the membership management of the system by using multiple trackers or using a DHT tracker. A Distributed Hash Table (DHT) works as a hash table using the SHA-1 hashes of content names as keys. A SHA-1 hash identifies in unique manner a Torrent. The data stored in a DHT node (peer) is the peer list for a specific Torrent. Each node is assigned an identifier ID in the namespace of keys (SHA-1 hashes). Using Kademlia [Kad 2010], the BitTorrent's DHT organizes nodes in a defined topology so that each node knows much more about near nodes than close ones. Hence, it facilitates the search operations. The search complexity is about $O(logn)$.

2.2 Wireless multi-hop networks

Nowadays, wireless networks have become more and more popular as they are easily deployable. These networks play a crucial role among computer networks, since they offer solutions to support mobility and essential services without the need of any installed infrastructure. Wireless networks can be classified into two categories: Infrastructure wireless networks using generally the cellular communication model and wireless networks without any infrastructure called ad hoc networks. An ad hoc network consists of a set of mobile entities (computers, PDAs, mobile phones, etc) moving in any environment and using wireless interfaces as communication links. The main sources of problems encountered in such networks are bandwidth limitation, energy limitation and the pseudo-random mobility of nodes. Here are more details about the characteristics of Mobile ad hoc networks (MANETs):

- A dynamic topology due to the mobility of nodes and churn. The network can even be partitioned into separate islands when nodes go out of the wireless range of each others.
- A limited bandwidth that diminish dramatically with the size of the exchanged information. Apart from being scarce, this bandwidth is shared among the set of nodes. In fact, nodes in the transmission rage of each other can not send packets simultaneously because of radio interferences. Furthermore, sending packets to far away destinations steals bandwidth in intermediate nodes acting as routers.
- Nodes are constrained energetically and their autonomy is dependant to their battery loads. Hence, they must minimize packet transmissions to limit energy consumption.
- Wireless channels can be subject to severe errors and losses due to fading and collisions and other exterior interferences.

Despite of these constraints, wireless ad hoc networks have the particularity of being self-constructed, self-organized and self-configurable without needing any fixed infrastructure. Wireless ad hoc networks are traditionally used in military applications, emergency services (earthquakes, fires, flooding, etc) and sensor networks (climatology, meteorology, detecting earth movements, etc). Nevertheless, the tremendous increase in computing abilities of devices and in their network capacities has encouraged users to connect to each other to form communities in order to share their experience (Social networks, content sharing, video streaming, etc). Hence, new applications already popular in the Internet such as Instant messaging, file sharing and social networks are implemented for the wireless ad hoc environment.

To run the applications described above, one need to ensure the connectivity of the network and the routing of packets. The routing algorithm is a strategy that ensures the connectivity between each couple of nodes at any moment. This strategy must take into consideration the changes in the network topology and other important characteristics such as the bandwidth, the number of links, the limitation of energy, etc. Considering this challenge, many routing protocols have emerged to answer different objectives and solve different problems. In the following paragraphs, we present the main routing solutions for wireless ad hoc networks using different current strategies. There are many criteria to design and classify routing protocols for wireless ad hoc networks: how nodes exchange routing information, when and how paths are computed, etc. Hence, three large categories of routing protocols can be distinguished:

- **Proactive protocols:** Paths are pre-established based on a periodic exchange of routing tables.
- **Reactive protocols:** Paths are found by the network on-demand. Nodes request each other in order to detect a possible path to the destination.
- **Hybrid protocols:** They combine the proactive and reactive approaches to profit of their advantages and reduce their drawbacks.

2.2.1 Proactive routing protocols

Proactive routing protocols try to maintain, in each node, up-to-date routing information for all destinations in the networks. Every node stores this routing information in a table called routing table. As the topology of a MANET is very dynamic, every node needs to send periodically control messages to its physical neighbors in order to update its routing table. To accelerate the routing decisions, the main idea is to keep, in each node, information on how to reach all destinations in the network. Hence, the changes in the topology are taken into consideration by propagating information about new paths among nodes.

Unfortunately, these protocols suffer from bad performance for very large networks or for high speeds of mobility. As topology changes are frequent, the network will be constantly flooded by control messages, which reduces dramatically the bandwidth. However, when many packets must be sent successively to the same destination, it is better to have a pre-established path than probing the network each time for a new path. In the reactive protocols, the source node must establish a new path for each new destination. In the following paragraphs, we explain the operations of some proactive routing protocols

2.2.1.1 Destination Sequenced Distance Vector (DSDV)

The DSDV protocol [Perkins 1994] is based on the distributed Bellman-Ford algorithm (DBF). Each node maintains in its routing table for each destination the following set of information:

- The address of the destination.
- The number of hops to reach the destination.
- A sequence number in order to distinguish new routes from the old ones. Hence, one can avoid routing loops.

In order to conserve the consistency of routing tables in a dynamic topology, every node sends periodically its routing table to its physical neighbors. To limit the overhead caused by these updates, nodes do two types of updates: complete updates and incremental updates. In complete updates, nodes send the totality of their routing tables, whereas in incremental updates, only new or modified entries are sent to physical neighbors.

2.2.1.2 Optimized Link State Routing (OLSR)

The OLSR protocol [Jacquet 2001] is a link-state routing protocol that establishes shortest paths to destinations. Unlike traditional link-state routing algorithms where each node broadcasts its neighbors (the state of its links) to all its physical neighbors, nodes using OLSR send their link-state information only to a sub-set of nodes called multipoint relays. These nodes are chosen in a way to reach all two-hop nodes. The role of multipoint relays is to reduce the traffic yielded by flooding control messages and decreasing the number of link states sent in the network. In order to maintain up-to-date information, needed for selecting multipoint relays, nodes send periodically HELLO messages containing the list of their physical neighbors. Another type of messages called Topology control is used to announce sub-sets of multi-point relays. Having all this information, each node has permanently a map of the network containing all nodes and only links necessary for the computation of the routing table. Hence, nodes compute routing tables using the graph of the topology of the network.

2.2.2 Reactive routing protocols

Reactive routing protocols keep in theirs tables only routes that are currently used. In fact, when a node wants to send a packet to a destination, it probes the network to find a path to this destination. This approach is called on-demand establishment of routes and avoids flooding the network by control messages. Furthermore, the node does not keep in their tables unused routes. However when a packet is sent to a new destination, the protocol needs some time to determine a path to this destination.

2.2.2.1 Dynamic Source Routing (DSR)

The DSR protocol [DSR 1996] is based on the source routing technique. Following this technique, the source of a packet finds the complete sequence of nodes by which this packet will transit to reach the destination. Before sending a packet to a destination, the source broadcasts a Route Request message. If the route discovery is successful, the source receives a Route Response packet containing the sequence of nodes in the path to the destination. In fact, receiving a Route Request message, an intermediate node adds its identifier in a specific field of this message and forwards it to its physical neighbors until the packet arrives to the destination. Using the source routing strategy, intermediate nodes do not need to store anything about the paths to destinations since they obtain this information in the data packets.

In addition to the route discovery mechanism [DSR 1996], DSR has a second mechanism that adapts the existing source routes to changes in the topology. In fact, when a link between two nodes becomes obsolete, the source is informed and then, it can use an alternative path stored in its cache. Otherwise, it searches a new route.

2.2.2.2 Ad hoc On-demand Distance Vector (AODV)

This protocol is essentially an enhancement of the DSR algorithm [Perkins 1999]. It reduces the number of messages broadcasts to establish routes. Unlike DSR, intermediate nodes maintain the totality of routes as long as they are used by the source node. AODV uses sequence numbers to date different routes in order to keep those that are up-to-date. As nodes of wireless ad hoc network are mobile, routes change frequently and hence some of routes stored in nodes' caches become obsolete. Like DSR, AODV uses two main mechanisms: route discovery and route maintenance.

AODV builds routes using a route request/route reply query cycle. When a source node desires a route to a destination for which it does not already have a route, it broadcasts a route request (RREQ) packet across the network. Nodes receiving this packet update their information for the source node and set up backwards pointers to the source node in the route tables. In addition to the source node's IP address, current sequence number, and broadcast ID, the RREQ also contains the most recent sequence number for the destination of which the source node is aware. A node receiving the RREQ may send a route reply (RREP) if it is either the destination or if it has a route to the destination with corresponding sequence number greater than or equal to that contained in the RREQ. If this is the case, it sends in unicast a RREP back to the source. Otherwise, it rebroadcasts the RREQ. Nodes keep track of the RREQ's source IP address and broadcast ID. If they receive a RREQ which they have already processed, they discard the RREQ and do not forward it.

As the RREP propagates back to the source, nodes set up forward pointers to the destination. Once the source node receives the RREP, it may begin to forward data packets to the destination. If the source later receives a RREP containing a greater sequence number or contains the same sequence number with a smaller hopcount, it may update its routing information for that destination and begin using the better route.

As long as the route remains active, it will continue to be maintained. A route is considered active as long as there are data packets periodically travelling from the source to the destination along that path. Once the source stops sending data packets, the links will time out and eventually be deleted from the intermediate node routing tables. If a link break occurs while the route is active, the node upstream of the break propagates a route error (RERR) message to the source node to inform it of the now unreachable destination(s). After receiving the RERR, if the source node still desires the route, it can reinitiate route discovery.

2.3 The P2P paradigm in wireless networks

A mobile ad hoc network is a collection of mobile wireless nodes, which form dynamically a temporary network without using any infrastructure or centralized administration. In such a network, nodes are both routers and end-users. They forward packets at the routing layer to other mobile nodes. Hence, if one node is not in the transmission range of another, it can use a succession of nodes that are in range of each others to forward a packet to the destination. MANETs are constructed with the goal of profiting from the collaboration among the set of nodes. Applications supporting such collaboration generally need resource localization services, data and information sharing services, the support of multipoint, etc. Moreover, these applications must be designed in a decentralized manner and must not suppose the existence of any infrastructure.

Recently, P2P systems have gained a lot of popularity since users profit from resources offered by thousands of other users. In fact, such systems are composed of dynamic sets of nodes connected to the Internet and was initially designed to ensure file sharing. Nowadays, the P2P paradigm emerges as a general philosophy for constructing large scale services and distributed applications in the Internet. Hence, one can define without loss of generality P2P systems as being self-organizing and distributed systems. The nodes of a P2P network play symmetric roles. Indeed, they are both clients and servers. However, some of these nodes can be potentially non reliable and can show different levels of collaboration. Furthermore, P2P networks are a good example of Overlay networks. An Overlay network is an abstraction of the physical network at the application level. Consequently, an ideal P2P overlay network must be self-organized and decentralized and must hide the diversity and heterogeneity of its nodes.

Although they are used independently, P2P overlay networks and MANETs share many common characteristics like self-organization and decentralization. This is due to the common nature of their distributed components: On one hand, a P2P Overlay network is composed of a dynamic set of nodes connected through the Internet. On the other hand, a mobile ad hoc network is composed of mobile nodes communicating together with multi-hop wireless links. These common characteristics yield other similarities:

- Both networks have flat topologies with frequent changes caused by nodes that join and leave the network. For wireless ad hoc networks, the mobility of nodes also causes changes in the topology.

- Both networks establish connections hop by hop. Multi-hop connections in P2P networks are typically constructed thanks to TCP connections without any physical limitations. Whereas in MANETs, the multi-hop wireless connections are limited by the range of radio transmission.

The Common characteristics between P2P Overlay networks and MANETs show that these two types of networks share the same main challenge of ensuring the connectivity in a dynamic and decentralized environment. Hence, there is a synergy between the two networks in terms of their goals and the design principles of their algorithms and protocols. These algorithms and protocols must consider the dynamic nature of the network topology due to churn or mobility. The similarities between P2P networks and MANETs and the design concerns shared among them brought to life a new research direction in computer networks, which profits from the synergy existing between P2P overlay networks and mobile ad hoc networks to design better routing protocols and applications.

Figure 2.5 presents the design space of network protocols for both the Internet and MANETs. It also gives examples of protocols designed in each sub-space.

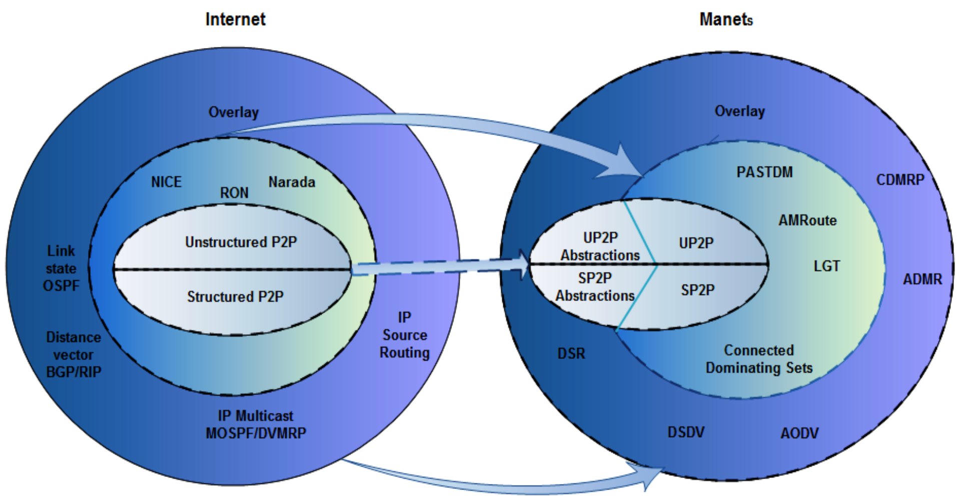


Figure 2.5: Mapping between Internet protocols and MANET protocols

The protocols developed for the Internet can be classified into two main categories:

- Protocols developed with an approach using routers. Examples of such protocols include Internet routing protocols like RIP [RIP 1988], OSPF [OSP 1994] and BGP [BGP 1995], and IP multipoint protocols like MOSPF [MOS 1994] and DVMRP [DVM 1988].

- Protocols developed to ensure overlay networks (limited by the internal circle in Figure 2.5.). They concern hosts and hence the edge of the network.

The research effort on P2P in the Internet has been focused on overlay networks of end-hosts. The set of overlay-based protocols include unstructured P2P protocols

(Examples: Gnutella [Gnu 2010], KaZaA [KaZ 2010]) and structured P2P networks (examples: Chord [Stoica 2001], Tapestry [Tap 2001], Pastry [Rowstron 2001b], CAN [Ratnasamy 2001]).

Similarly, MANETs have protocols equivalent to those proposed for the Internet. In mobile ad hoc networks, some approaches involve all nodes of the network. They include routing protocols such as DSR [DSR 1996] and AODV [Perkins 1999], and multipoint protocols like ADMR [Jetcheva 2001] and ODMRP [Lee 2001]. Other approaches acts on an overlay of nodes (example: AMRoute [Xie 2002]).

In the following paragraphs and the remainder of this thesis, we will be answering the following important questions:

- Can one profit from the similarity existing between the architectures of P2P networks and MANET to design protocols and applications for MANETs?
- If we want to apply what has been done for P2P networks in the Internet to mobile ad hoc networks, must-we apply it to an overlay of nodes (a sub-set of nodes) or all nodes of the network?
- How to adapt P2P overlays to mobile ad hoc networks?

In the remainder of this section, we dress the state of art of recent studies on mapping Internet P2P overlay protocols to MANETs. We first describe some investigations that study the efficient support of P2P overlay abstractions in mobile ad hoc networks. Then, we discuss how and when the MANET applications can profit from the use of P2P abstractions and we study the challenges of this use. Finally, we discuss the open research issues that motivated this thesis.

2.3.1 Supporting abstractions of P2P overlay networks in MANET

In this paragraph, we first show the difference between supporting P2P overlays and abstractions of P2P overlays in MANETs. Then, we discuss challenges and motivations behind supporting P2P overlay abstractions in MANETs. Finally, we expose different approaches of supporting abstractions of P2P overlays in MANETs.

2.3.1.1 Why supporting Overlay network abstractions and not overlay networks?

Although overlay networks can be constructed in wireless ad hoc networks based on a sub-set of nodes, almost studies of P2P overlays in MANET have chosen to involve all nodes of the network. In fact, an ad hoc network is composed of nodes collaborating to ensure the global connectivity among them. Since all nodes are already involved, the notion of overlay network is no longer precise. Recent studies support P2P overlay abstractions in MANET, they apply object-localization techniques of P2P overlays in MANETS. As Internet overlays have been constructed in order to avoid involving intermediate routers, the routing of request messages is done at the application layer. However, in MANETs, nodes are routers and hosts at the same time. Hence, P2P overlay abstractions can be implemented at the network layer of at the application layer. These design alternatives are discussed later in this section.

2.3.1.2 Why supporting Overlay network abstractions in MANETs?

The main motivation behind supporting P2P overlay abstractions in MANETs is the fact that MANETs and P2P overlay networks have many common characteristics such as decentralization. Hence, P2P applications developed for the Internet are best candidates to be deployed in wireless ad hoc networks. For example, P2P file sharing applications like Gnutella [Gnu 2010] are designed following an architecture without servers and consequently, they can potentially be used in environments without infrastructure such as MANETs.

Structured P2P overlay networks developed for the Internet have shown their efficiency in constructing a variety of robust applications for the Internet such as distributed storing systems [Dabek 2001] [Rowstron 2001a], application-layer multicast [Castro 2003a][Castro 2003b] and content-based search [Tang 2004]. The implementation of a DHT abstraction to obtain a structured P2P overlay network needs considering many aspects including fault tolerance, object localization, scalability, availability and load balancing. The objective of supporting DHT abstractions in MANETs is the same objective as in Internet.

Several fundamental differences exist between the Internet and mobile ad hoc networks. Hence, many challenges rise up when implementing P2P overlay abstractions in MANETs. Here are some of these challenges:

- **Bandwidth limitation:** Unlike the wired Internet, MANETs have low network capacities due to the use of wireless channels. P2P protocols designed for the Internet can yield a very important overhead in wireless networks and then their use can be limited.

- **Multiple-access interference:** To access the channel and transmit data, nodes of a MANET use multiple access mechanisms like CSMA/CA. Since there are no central point of coordination in mobile ad hoc networks, when acquiring the wireless channel, nodes can suffer from collisions and waiting delays due to interferences. While in the Internet, when a peer sends some data, it does not interfere with transmissions of other peers.

- **Routing overhead:** As nodes of a MANET are both routers and hosts, they have to forward packets for other nodes even if they are not interested in the content they receive. Sometimes if the P2P overlay does not adapt to the topology of the network, pieces of content can arrive to a node at the network layer without profiting from them at the application layer.

- **Mobility of nodes:** In the Internet, the topology of a P2P overlay network changes in a large temporal scale. However, in a MANET, the mobility of nodes and the limitation of their transmission ranges engender frequent changes in the topology. Hence, when applied to MANETs, P2P applications must update their topologies with a higher frequency in order to keep a good mapping of the P2P overlay topology to the topology of the physical network. The management of the topology in P2P overlays is ensured thanks to probing current neighbors and candidate neighbors and by selecting those that corresponds better to a performance criteria. These criteria can be for example the closeness or the upload capacity. The management in this case focuses on testing the availability of neighbors. In the Internet, the management of P2P neighbors is easy since the paths between nodes changes scarcely. However, in MANETs, due to the limitation of the capacity of links and interference, probing the network to discover new neighbors is very costly.

- **Churn:** In the Internet, structured P2P protocols are affected by arrivals and departures of nodes of the overlay as most of them are desktop computers and not always online servers. Whenever structured P2P protocols are used in MANETs, they can eventually suffer from bad performance because they are affected in addition to churn by network partitions caused by mobility.

- **Lack of infrastructure:** Some P2P protocol involve in their design some infrastructure components. For example, a P2P routing protocol can assign identifiers to nodes based on their locations compared to those of some fixed landmarks. As there is no fixed infrastructure in MANETs, these techniques can not be used in the wireless environment.

- **Limited energy:** Almost P2P applications of the Internet are not designed to send a minimum number of messages. In an environment, where the energy is limited, it is mandatory to reduce the number of sent packets while keeping an acceptable performance. For example, the majority of Internet P2P protocols use a proactive update of their state. However, reactive approaches in MANET such as AODV [Perkins 1999] and DSR [DSR 1996] have shown to be more efficient.

- **Addressing:** Although no standard addressing architecture exist for MANETs, one can assume that nodes connect and reconnect many times and obtain each time new IP addresses. The overlay network must adapt to this change in addresses.

2.3.1.3 Design approaches

The different design approaches of P2P applications in MANETs can be classified following the nature of their P2P overlays (structured or not structured). Moreover, as we discussed earlier, abstractions of P2P overlay networks can be implemented at the routing layer or at the application layer. If the P2P protocol is constructed on the top of a routing protocol, the design approach is called a layered approach. In case, the P2P protocol is integrated in the functionalities of a routing protocol, it is a cross-layer design approach.

The layered design allows P2P applications developed for the Internet to migrate easily to the MANET world. This approach completely decouples the functionalities of the P2P application layer from those of the network layer. Nevertheless, MANETs are environments where the resources are limited. If one can profit from existing routing protocols to integrate P2P functionalities at the routing layer, this can save resources and obtain better performance while loosing portability and separation of layers.

Inspired from the classification work done in [Hu 2004], we classify the design approaches of P2P overlay networks abstractions in MANETs into four categories:

- **Layered and unstructured design:** unstructured P2P protocols are deployed on the top of existing MANET routing protocols.
- **Cross-layer and unstructured design:** In this design, the operations of an unstructured P2P protocol are integrated in the operations of a MANET routing protocol. The latter protocol supports the APIs of an unstructured P2P network.
- **Layered and structured design:** Following this design, a structured P2P protocol such as Pastry [Rowstron 2001b], CAN [Ratnasamy 2001], Chord [Stoica 2001] and Tapestry [Tap 2001] runs as an application on the top of a MANET routing protocol.
- **Cross-layer and structured design:** In this design, the operations of a structured P2P protocol are integrated in the operations of a MANET routing protocol in order to offer an abstraction of a distributed routing table.

2.3.2 Unstructured P2P overlay networks abstractions in MANETs

2.3.2.1 Layered design

The research work done in [Oliviera 2003] studies the performance of a P2P application running on the top of three different MANET routing protocols. Figure 2.6 shows a diagram of a P2P application running on the top of the network layer of a MANET. The authors deployed a Gnutella-like P2P protocol [Gnu 2010] on the top of DSR [DSR 1996], AODV [Perkins 1999] and DSDV [Perkins 1994] routing protocols. They compare their relative performance and show that the observed results are different from those obtained with unicast applications. Figure 2.7 draws an example of a comparison between the three routing protocols. It dresses the curves of the delivery ratio as function of the number of nodes in the wireless ad hoc network. This results where compared in [Broch 1998] to unicast applications.

Considering a research request, we will explain the drawbacks of layered unstructured design. Like in the Internet, Gnutella operates at the application layer and each peer node contacts its overlay neighbors to solve a request. These neighbors forward this request to their neighbors if its decremented TTL is higher than 0.

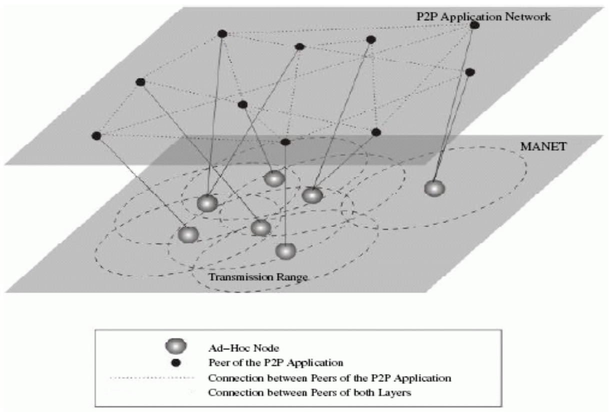

P2P Application Network
MANET
Transmission Range
Ad–Hoc Node
Peer of the P2P Application
Connection between Peers of the P2P Application
Connection between Peers of both Layers

Figure 2.6: A P2P application on top of a MANET

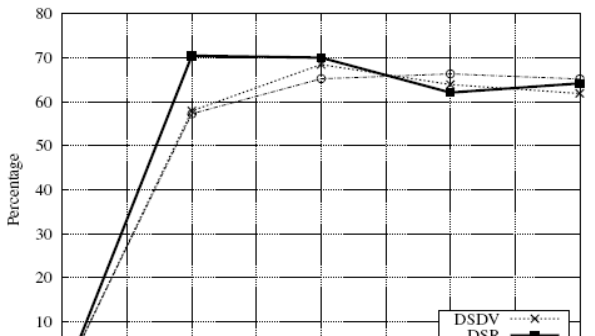


Figure 2.7: Delivery ratio for the P2P application

However, due to the mobility of the nodes, neighbors in the Overlay network do not reflect the current physical topology of the wireless ad hoc networks. Hence, the destination is reachable through a path of several hops. So, flooding request messages is a very costly task. To avoid this, the P2P overlay must adapt to the frequent changes in the physical network topology, which is not an easy task in such challenging networks.

2.3.2.2 Cross-layer design

In [Klemm 2003], authors propose an integration of a unstructured P2P application like Gnutella [Gnu 2010] in the network layer. They compare this cross layer design called ORION to the layered design proposed in [Oliviera 2003]. ORION allows establishing connections on demand at the Overlay level. These connections correspond practically to connections at the physical level. In fact, ORION integrates the requesting process necessary to P2P operations in AODV [Perkins 1999] routing techniques. When receiving a request, an ORION node broadcasts it to its physical neighbors. Hence, only one packet is sent to reach all the one-hop neighbors. The operation is done by all nodes until the request reaches a node that can answer the request. The answer message is sent directly to the initiator of the request in a multi-hop path. However, this path is already established by the AODV [Perkins 1999] routing protocol.

2.3.2.3 Comparison of the two approaches

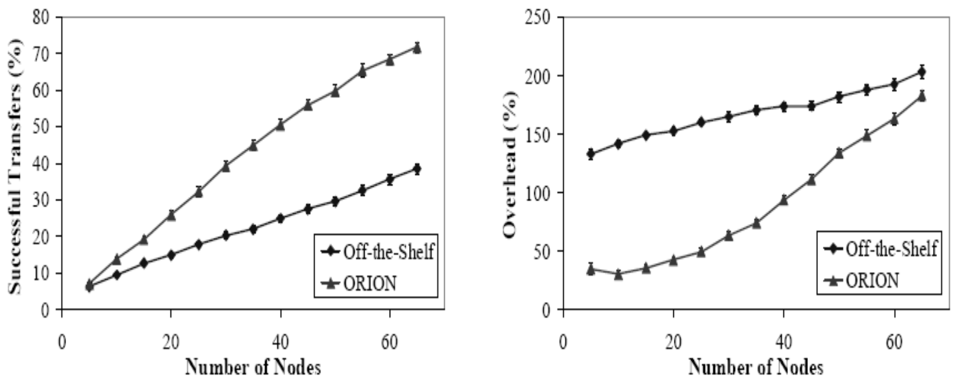


Figure 2.8: Comparison of the two design approaches

The overperformance of a cross-layer P2P file sharing application like ORION was proven in [Klemm 2003]. ORION has been implemented in the NS-2 network simulator and compared to the unstructured design that runs Gnutella [Gnu 2010] on the top of DSR [DSR 1996]. The simulations have been conducted on a topology of 60 nodes using the random waypoint mobility model. Nodes are moving with a maximal speed of 2 m/s with a pause time equal to 50s. Two main metrics have been considered the ratio of successful transfers and the overhead in number of sent packets. Figure 2.8 plots these metrics for both designs (ORION: Cross-layer design, Off-the-shelf: layered design). The results show that the ratio of successful transfers for the cross-layer design increases with the number of nodes but is always higher than that of the layered design. In fact, the number of successful transfers decreases with the increase of the size of the network as we have an important routing overhead. As a conclusion, the main factor behind bad performances observed for the layered design is not the inefficiency of the searching mechanism but the overhead of maintaining static connections with overlay neighbors. These connections are not adapted to the dynamic nature of the physical topology.

2.3.3 Structured P2P overlay networks abstractions in MANETs

We distinguish in our classification two methods of deploying DHTs in MANETs. The first method uses GPS (Global Positioning System) and the second one does not use it. When GPS is supported, every node can determine its geographic position. This localization information has been used before to enhance the efficiency of multi-hop routing protocols or in supporting other types of services like Geocast [Geocast]. In the first method, this information is used to improve the performance of DHTs in MANETs.

2.3.3.1 DHTs in MANETs without supporting GPS

Both layered and Cross-layer design approaches of supporting DHT abstraction in dynamic mobile ad hoc networks have been compared in [Das 2004]. Mainly, a simple DHT superposed on a MANET multi-hop routing protocol has been compared to Ekta [Das 2004], which integrates a DHT in a routing protocol of the network layer.

Layered design: In the layered design, a proximity-aware Pastry [Rowstron 2001b] has been constructed on the top of the DSR [DSR 1996] routing protocol without significant changes. Indeed, the layered design is similar to that in the Internet. Although this approach follows the ISO model, it does not profit from optimization opportunities offered by the interaction between the DHT and the underlaying routing protocol. For example, one can use the same structures used by the routing protocol to store routes in the DHT. When the routing protocol is DSR [DSR 1996], the routing cache can contain in addition to source routes those at the DHT layer.

Cross-layer design: In the cross-layer design, named Ekta in [Das 2004], the functionalities ensured by a Pastry DHT [Rowstron 2001b] are integrated in the operations of the DSR routing protocol [DSR 1996]. The logical namespace of Pastry is mapped to the physical namespace of DSR. Hence, the routing data structures of the DHT and the routing protocol are merged in a same structure. Using this new structure, one can profit from the interactions between both protocols in order to enhance the performance at the two routing levels.

- Addressing nodes: Ekta assigns unique identifiers to nodes of the MANET by using a collision-resistant hashing function such as SHA-1 [SHA 1995] applied to IP addresses of hosts.
- Routing: In Ekta, a message is routed using its key based on Pastry pre-fixes. The message is delivered to the destination having a node identifier the nearest to the key.
- Optimizations: Ekta inherits from the optimization of routes and their maintenance from the DSR routing protocol. Moreover, Ekta updates its tables using routes detected when the node forwards a packet or when hearing the channel. Hence, it is always discovering new optimal paths to destinations.

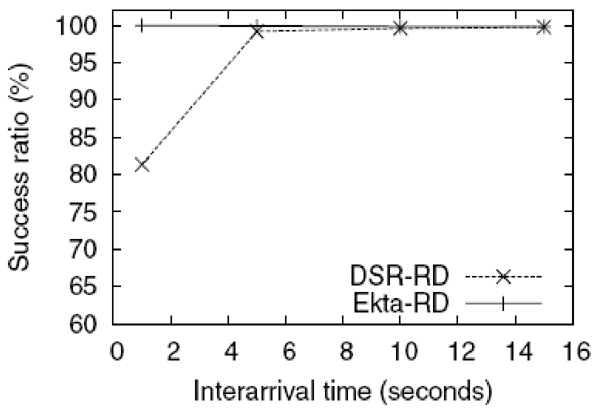


Figure 2.9: Success ratio for both layered and cross layer design

Comparison: A detailed study based on simulations has been conducted to compare the layered and the cross-layer designs. The results show that the cross-layer design is better than the layered design when considering the ratio of transfer successes and the average delivery delay of packets. Figure 2.9 presents the delivery ratio as function of the inter-arrival time for the two methods when the pause time is equal to 300s. (DSR-RD: Layered approach, Ekta-RD: Cross-layer approach). These results show that the integration of a DHT functionalities in the routing layer is more efficient that having two separate and independent layers.

2.3.3.2 DHTs in MANETs with supporting of GPS

The geographic localization system (GLS) in GRID [Li 2000] is a scalable localization service that matches the nodes identifiers to their locations. The implementation of GLS engenders an abstraction of a DHT: A message with a key Y is routed to the node having the identifier the nearest to Y. However, the implementation of GLS requires the support of GPS and the construction of a distributed localization data base.

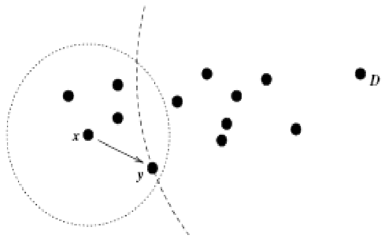


Figure 2.10: Greedy geographic routing

Geographic Hash Tables (GHT) [Ratnasamy 2002] are similar to Internet DHTs. But, they hash a search key in geographic coordinates of nodes and store information in the node that is the nearest geographically to these coordinates.

2.4 Opportunistic content-centric networking

In this thesis, we will mainly focus on scenarios where connectivity among the set of nodes is almost guaranteed. Differently speaking, one of the basic assumptions behind our work is that nodes form one or more big connected clusters so that the current Internet protocol stack (TCP/IP) still works. However, in reality due to high speed mobility, the wireless propagation phenomena and the limited energy of devices, one can imagine scenarios where the connectivity is episodic and intermittent. For these extreme and challenging environments, our assumptions do not hold and the existing Internet architecture, basically replying on the end-to-end principle, is known to perform very badly. The solutions proposed for these challenging environments, even if not applicable within our context, are relevant to ours, this is why we overview them here.

In the literature, many works have concentrated on finding solutions to route messages to their destinations in challenging environments where mobility and disconnections are dominant. They are mainly based on store, carry and forward mechanisms. Indeed, intermediate nodes store some messages in their caches until delivery opportunities occur. The research communities answered mainly the questions of how to route messages in these networks in order to optimize some performance metrics such as delivery ratio, delivery time and network load. Other works have focused on resource management by identifying optimal policies for message scheduling and drop upon congestion. In the first paragraph of this section, we overview the routing and resource management solutions which are available for opportunistic wireless networks.

When content sharing is considered in an environment with intermittent connectivity, the research community proposes to consider the opportunistic networks as a content-centric network [Jacobson 2009]. The objective is that contents are shared directly between intermittently connected mobile devices. They aim to provide an architecture that virtualises the underlying network and they consider that devices must concentrate on contents rather than reaching specific network destinations. An opportunistic content centric network such as Haggle [Olafur R. Helgason 2010] provides temporal and spatial decoupling so that contents can be distributed independently of their locations and the availability of their sources. Such temporal and spatial decoupling is widely argued as a corner-stone of a modern communication architecture [Erik Nordstrom]. Haggle implements such decoupling using a data-centric communication model with a publish-subscribe (pub/sub) API [Carzaniga 2003], which spreads application data from device to device based on the data's match against a user's declared interests.

In the Internet, network use has evolved to be dominated by content distribution and retrieval, while networking technology still speaks only of connections between hosts. Accessing content and services requires mapping from the what that users care about to the network's where. In [Jacobson 2009], authors present Content-Centric Networking (CCN) which treats content as a primitive - decoupling location from identity, security and access, and retrieving content by name. Using new approaches to routing named content, derived heavily from IP, they can simultaneously achieve scalability, security and performance.

To ensure the virtual aspect of the episodic network, in data-centric communication, unlike classical IP communication, there are no such clear a priori mappings between nodes and contents and there are no centralized lookup and mapping services because such mappings do not work without continuous infrastructure connectivity. Haggle, for instance, is based on a distributed search-based resolution mechanism. This search mechanism is designed to optimize network resource utilization considering users interests. The main idea is that each device can limit the amount of disseminated data to only the top ranked nodes with the most interest in the data. In the second paragraph of this section, we overview the stat of the art related to the opportunistic content-centric architecture.

2.4.1 Intermittent connectivity networks

With nodes mobility, limited energy, limited wireless range and churn, one can no longer consider a wireless ad hoc network as a connected graph. Indeed, the connectivity of the wireless network is in most cases intermittent. Despite this episodic connectivity, many applications such as those providing remote areas with Internet connectivity [Fall 2003][Lenders 2007], vehicular networks [Su 2007], point-to-point connectivity [S. Jain 2004][A. Pentland 2004][Basu 2002], content sharing, advertising, sensing, etc can still be supported. This is possible thanks to the Delay Tolerant Networking (DTN) paradigm [DTN 2010].

To ensure the routing of messages in DTNs, store-carry-and-forward protocols are proposed, where a node may store a message in its buffer and carry it along for long periods of time, until it can forward it further. This routing may happen randomly, be based on statistical information [A. Lindgren 2003], or even other relevant information about the destination (e.g. social links, affiliation, etc.). Furthermore, due to the inherent uncertainty caused by the lack of complete (or any) information about other nodes in the network, many replicas of the same message may be propagated to increase probability of delivery. For example, one of the first and most popular routing protocols in this context, namely Epidemic routing [Vahdat 2000], disseminates a message replica to every node in the network.

2.4.1.1 Epidemic routing

Vahdat and Becker present a routing protocol for intermittently connected networks called Epidemic Routing [Vahdat 2000]. This protocol relies on the theory of epidemic algorithms [X. Zhang 2006][A. Balasubramanian 2007] by doing pair-wise information of messages between nodes as they get contact with each other to eventually deliver messages to their destination. Hosts buffer messages even if it there is currently no path to the destination available. An index of these messages called a summary vector is kept by the nodes, and when two nodes meet they exchange summary vectors.

After this exchange, each node can determine if the other node has some message that was previously unseen to this node. In that case, the node requests the messages from the other node. This means that as long as buffer space is available, messages will spread like an epidemic of some disease through the network as nodes meet and infect each other.

2.4.1.2 Probabilistic routing

In [A. Lindgren 2003], authors propose to reduce the number of copies exchanged among nodes by supposing that their mobility is not completely random. They proposed PROPHET, a probabilistic protocol for routing in intermittently connected networks that is more sophisticated than previous protocols, using history of node encounters and transitivity to enhance performance over previously existing protocols.

Many other works have been conducted to optimize the Delay tolerant routing protocols [A. Krifa 2008][Alan Demers 1987][Werner Vogels 2002]. They mainly optimize the buffer management policy and the scheduling of messages. The objective is to minimize the delivery time and to maximize the delivery ratio while having a minimum consumption of network and node resources.

2.4.2 Opportunistic content-centric networks

The opportunistic content-centric networking approach is very relevant to our work. It considers the case of intermittently connected networks and supposes that contents are exchanged opportunistically when nodes are within communication range. The success of such an architecture is obtained thanks to the virtualization of the network routing and the decoupling of the location (address) from the content itself.

A content can be obtained independently of its location and the application installed in a node does not even need to know about the routing mechanism.

The service of a content-centric network such as haggle [Olafur R. Helgason 2010][Erik Nordstrom] is accessed thanks to publish/suscribe interface and therefore do not have to deal with low-level opportunistic networking issues or matching and soliciting of contents.

Contents are grouped by logical topics in order to efficiently answer content lookups under intermittent connectivity. Moreover, nodes use a solicitation protocol in order to discover contents available in the neighborhood and to download contents' chunks disjointly from different nodes.

This design is general and facilitates the implementation of advanced applications. However, the caching mechanism is interest-based. Indeed, nodes do not store contents they are not interested in. This may be not efficient in the case of a low mobility network or the case of non popular contents.

2.5 Conclusions and open issues

P2P networks and wireless ad hoc networks share several common characteristics like decentralization and self-organization. Although these similarities, P2P applications designed for the Internet can not be directly deployed in wireless ad hoc networks. In fact, the wireless ad hoc environment is very constrained. The resources are limited and shared among the set of nodes. Every packet sent across the network steals bandwidth from intermediate nodes in the path acting as routers. Hence, P2P protocols adapted to the characteristics of the wireless environment must be invented.

In this chapter, we presented the main solutions proposed in the literature to deploy the P2P paradigm in mobile ad hoc networks. However, these solutions are focused on the content localization space. They experiment with structured and unstructured content-lookup overlays when they run on the top of a MANET routing protocol or when they are integrated in the routing operations. The conclusion was that having a cross-layer design allows to save bandwidth and to send less request messages in the network. In fact, the neighborhood at the overlay level corresponds better to a neighborhood at the network level.

However, no extensive study has been conducted to study the best design choices and algorithms to be deployed when a P2P content sharing application runs in the MANET environment. How to construct the overlay from a data transfer prospective is still an open issue mainly when the content sharing application uses the multi-sourcing concept. In chapter 3, we study the performance of BitTorrent in wireless ad hoc networks and we show that one can opt for a layered design while considering topology information given by a proactive routing protocol. . In chapter 4, we continue concentrating on the data transfer plane, and present our P2P content sharing protocol whose design and algorithms are adapted to the constraints of wireless multi-hop networks. They mainly take into consideration the limited resources, the level of congestion and the speed of mobility.

Furthermore, the resource discovery mechanisms described in this chapter suppose that the resource is located at only one node or that the service is offered by only one node. However, in some cases, a peer needs to know about the majority of the peers interested in the same service. For example, in a Torrent, a peer needs to have a list of peers with whom to exchange pieces of the content. In the remainder of this thesis, we study the case where the same service (content sharing) is supported by many nodes in the network. In this case, the search mechanism must discover the members of the service overlay and have constantly up-to-date information about them. In chapter 5, we will study in details the optimization of such membership mechanisms. We design a completely distributed approach that finds a good equilibrium between minimizing the overhead and having fresh information about the peers of the network.

Chapter 3

Performance of BitTorrent in wireless ad hoc networks

Contents

The proliferation of wireless devices (Laptops, PDAs, Smartphones, etc) motivates end users to connect to each other to form spontaneous communities. A multi-hop wireless network of devices, where the end-to-end communication is rendered possible by the help of ad hoc routing protocols, can be a good opportunity to share some content (data, audio, video, etc) among the members of the same community without using any established infrastructure. But, the resources of a wireless ad hoc network are very limited and shared among the devices, which play the role of both end-users and routers. Considering this constrained nature, any content sharing application must optimize the use of the resources and distribute the content replication load equally among the set of nodes. The classical client/server architecture is not the most appropriate for wireless ad hoc networks because of the burden of the multi-hop communication and the local congestion around the server. This latter node may become the main point of failure of the application. On the other hand, application-level multicast solutions are known not to distribute the load equally among nodes and hence lack incentives for collaboration. Reliability is also an issue given the difficulty to retransmit lost packets on an end-to-end basis over the multicast tree.

Considering this, content sharing applications based on the peer-to-peer (P2P) paradigm are good candidate solutions to run over spontaneous wireless ad hoc networks for the following reasons. First, they have become, in a few years, the most popular applications in the Internet and users are familiar with their functionalities and features. Moreover, a P2P file sharing solution like BitTorrent for example

[Bit 2010b] decentralizes the data transfer plane using the multi-sourcing concept and provides enough incentives to encourage fair sharing. It is thus important to have the same principles applied in a wireless environment where nodes tend to save capacity and energy.

In this chapter, we study the performance of the Internet version of BitTorrent when it is deployed on the top of a wireless ad hoc network. To ensure our study, we conduct both NS-2 simulations [NS- 2010] and ORBIT experiments [ORB 2010]. In the considered scenarios, we vary the percentage of nodes interested in sharing the same content in order to have different Torrent densities. First, we study the impact of the scope of the neighborhood both on routing overhead and connectivity of the overlay. We show that when limiting the scope of sharing, one reduces considerably the routing overhead and as a result one obtains shorter download times. In fact, in the classical version of BitTorrent, the neighborhood of a peer is constructed independently of any information on the underlying topology. In a wireless ad hoc network where the resources are limited and nodes are routers and end-hosts, this large scope of sharing engenders a large number of pieces that are forwarded at the network layer of peers interested at the content without profiting from them at the application layer. Furthermore, TCP connections suffer from very bad performance when the number of hops between peers is big. Hence, to have a better performance, the scope of sharing must be decreased to some routing hops. In this case, results show an amelioration of the download time for all Torrent densities. However, limiting the neighborhood in the case of sparse Torrent disconnects the sharing overlay and some peers will never finish downloading the file since they are isolated from the rest of the Torrent. We study in this chapter how the completion ratio varies with the scope of the neighborhood for different densities of the Torrent.

In a second part of this chapter, we consider another important factor of the performance of BitTorrent, which is sharing opportunities. In fact, the goal of the Internet version of BitTorrent is to divide the burden of data transfers between peers so that a peer uploads the same quantity of data as it downloads. In our study, we evaluated at which extent the classical version of BitTorrent ensures this fair sharing among the set of peers when it is deployed in wireless ad hoc networks. Our results showed that the sharing ratios between peers were very low compared to those in the Internet. In reality, even if peers are willing to offer their upload capacities to others, they are limited by the constraints of the wireless communications, mainly when the communications occur between distant peers. However, although limiting the scope of the neighborhood ameliorates the performance of TCP connections, it limits the diversity of pieces in the network and then yields sharing ratios worse than those of the large neighborhood. Indeed, in case of a narrow neighborhood, pieces of the content propagate like a wave from the original seed of the content to the edge of the network. Hence, peers wait for pieces to arrive to their neighborhoods and do not have original pieces to exchange them with others. This decreases considerably sharing opportunities.

In a third part of this chapter, we study the impact of the mobility of nodes on the performance of BitTorrent. We mainly show that in a mobile network, there is no need to have a large scope of sharing for two main reasons. First, the TCP connections suffer for very bad performance due to frequent changes in the topology of the network, which engenders an important instability of long paths. Second, the mobility of nodes increases the diversity of pieces in the network and hence there are enough original pieces to exchange with others without the need of communicating with far a way peers.

This chapter is a summary of all studies we did on the performance evaluation of the Internet version of BitTorrent. The majority of the results shown here has appeared in different international conferences [Sbai 2009a] [Sbai 2010] [Sbai 2008] [Salhi 2008]. The remainder of this chapter is organized as follows. In Section 3.1, we describe the methodology used in the evaluation of BitTorrent in wireless ad hoc networks. Then, in Section 3.2, we motivate our study by comparing the performance of the P2P paradigm to other possible content sharing strategies such as client server and point-to-point. The next two sections present our study of the performance of BitTorrent in wireless networks following different metrics. In Section 3.3, we focus on the routing overhead and overlay connectivity problems and in Section 3.4, we concentrate on the sharing opportunities and study the piece diversity problem. Then, we study the impact of the mobility of nodes in Section 3.5. Finally, Section 3.6 summarizes our study and motivates the solutions proposed in the next chapter.

3.1 Methodology

To evaluate the performance of BitTorrent in wireless ad hoc networks in different scenarios, we conduct extensive simulations and extensive experiments. By varying the settings of our scenarios, we aim at making our results the most general possible.

3.1.1 Simulations

We extend the NS-2 network simulator [NS- 2010] by adding a general and tunable content sharing module, which is based on the algorithms of the well-known Internet protocol BitTorrent. Using our implementation, one can change different strategies of BitTorrent mainly the neighbor selection strategy and the choking algorithm. In addition to the data transfer plane, our module implements a peer discovery mechanism. This mechanism emulates for the BitTorrent client the existence of a centralized tracker providing it with the list of Torrent members. Furthermore, our module profits from the existing NS-2 modules to ensure wireless communication and multi-hop routing of packets. The wireless ad hoc network that we are simulating consists of 50 nodes randomly distributed in a 500mx80m square area. In our simulations, we discard all the realizations where the topology is not connected at the physical level. Nodes connect to each other using the 802.11 MAC Layer with the RTS/CTS-Data/ACK mechanism enabled. The data rate is set to 11 Mb/s and the wireless range to 50m. Without loss of generality, the ad hoc routing service is ensured thanks to the DSDV proactive protocol. At the beginning of each simulation and for each Torrent a random node is chosen as the seed of the corresponding content and another random set of nodes are selected as leechers. Each content is a 10 Mbytes data file that is subdivided into 100 pieces. All peers start downloading the file at the same time (a flash crowd scenario). The BitTorrent choking period is set to 40s.

3.1.2 Experiments

In parallel to the NS-2 simulations, we conduct extensive experiments over the ORBIT wireless testbed [ORB 2010]. We use and modify LibTorrent the open-source library and implementation of the BitTorrent protocol [Lib 2010] to compare the performance of the different variants of the protocol implementing different neighborhood selection strategies. In each experiment, we randomly select 100 nodes among the 400 nodes of the ORBIT testbed. Each node in this testbed is a PC equipped with Atheros AR5002X Mini PCI 802.11a/b/g wireless card attached to an omnidirectional antenna. We configure the wireless interface card to operate in 802.11b ad hoc mode, and set the transmission power level at 20 dBm, and the bit-rate at 11Mbps. Each node in the testbed runs Linux Debian kernel v2.6.22, Mad-Wifi v0.9.3.3 [Mad 2010], the OOLSR open source implementation of the OLSR routing protocol [OOL 2010]Motivations and the modified BitTorrent protocol. At the beginning of each experiment and for each Torrent considered, one of the nodes of the constructed ad hoc network plays the role of the initial seed and a randomly selected sub-set of the remaining nodes play the role of leechers. Each content is a 100 MB file. The number of peers participating in the sharing session for each file is determined by the Torrent density, a parameter of the experimentation.

3.2 Motivations and preliminary study

Spontaneous wireless multi-hop networks are an adequate field for content sharing among communities of users. Indeed, users can connect to each other in order to share data and multimedia files without being connected to any infrastructure network. To ensure this connection at the data transfer layer, they need to agree on a content distribution protocol. The classical data transfer methods namely the client/server and the application level multicast methods are not the most suitable for wireless ad hoc networks for many reasons. First, they yield important overheads on the underlying wireless network as the communication scheme is not designed for networks where resources are limited and shared. Moreover, the load of data transfers are not fairly distributed among the set of nodes since the nodes that are nearer to the source of the content will send more packets than other nodes that are far from it. The target of these methods is to have a hierarchy of nodes where some of them sacrifice some of their capacities to serve others without any incentives built in the protocol. Hence, a suitable content sharing paradigm must minimize the consumption of network resources and must divide the burden of sharing data equally among the set of nodes by thinking about the topology of the network and giving enough incentives for fair sharing. Furthermore, it must maximize the global capacity of the system by using the ability to have parallel communications in different areas of wireless ad hoc networks. Figures 3.1 and 3.2 compare between the client/server paradigm, the one-hop P2P paradigm and the classical Internet BitTorrent paradigm when they are deployed in a wireless ad hoc network. These figures plot the average download time of the users as a function of respectively the size of the content and the density of interest. Both figures show that P2P approaches outperform the client/server approach. However, the classical Internet version of BitTorrent shows a download time higher than that of a one-hop limited P2P protocol. This difference in the observed performance motivated our investigation. We mainly study how to select neighbors in a wireless ad hoc network in order to respect the underlying topology of the network. Can one do better than the simple one-hop narrow neighborhood and the wide neighborhood of the Internet version of BitTorrent ?

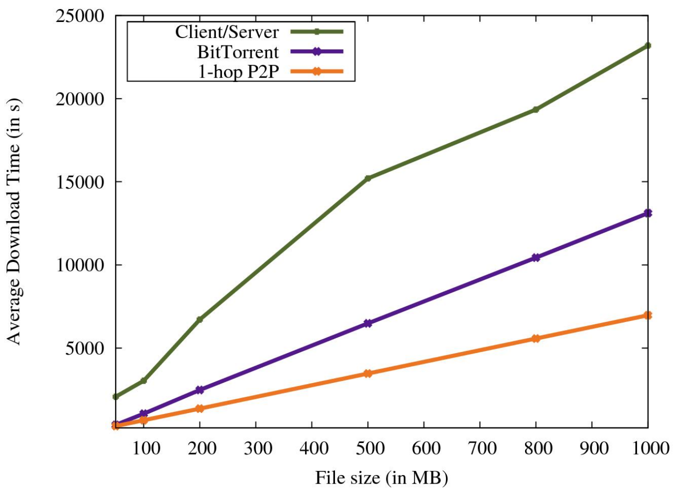


Figure 3.1: Download time for different paradigms (Different file sizes-ORBIT experiments)

3.3 Routing overhead Vs. Overlay connectivity

In this section, we focus on the data plane of BitTorrent in spontaneous multi-hop wireless networks. We mainly consider the case of a single Torrent in the network. As one of the main objectives of BitTorrent is to minimize the download time of peers, we compare the average download time of peers that have completed the download for different scopes of the sharing neighborhood by considering different Torrent densities.

We vary in our experiments the scope of the sharing area of BitTorrent and measure the average download time. A sharing scope set to h routing hops means that a peer cannot communicate with other peers located at a distance longer than

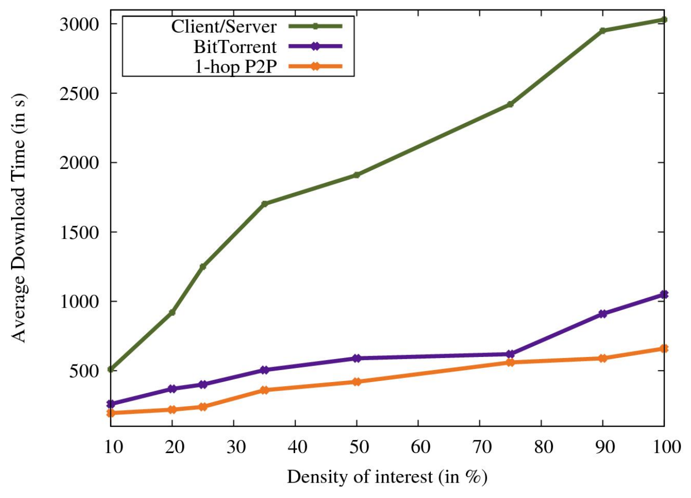


Figure 3.2: Download time for different paradigms (Different densities - ORBIT experiments)

h hops. The classical BitTorrent strategy is topology unaware. It can be seen as a one where the neighborhood scope is taken equal to the maximum number of hops in the network. This maximum is equal to 15 hops in our experiments. The following figures represent the same result differently: Figure 3.3 plots the average download time of users as a function of the sharing scope for different Torrent densities and Figure 3.4 draws the average download time of users as a function of the Torrent density for different sharing scopes. From these figures, one can make the following observations:

- The average download time increases with the increase of the Torrent density for all considered sharing scopes. The maximum download time is reached for the 100% Torrent density case. In fact, the stress on the underlying network increases with the number of peers.
- The classical version of BitTorrent (a sharing scope equal to 15 in experiments) yields the highest download times particularly when the overlay is dense. This is mainly due to the important routing overhead it engenders. In fact, following this strategy, all nodes communicate with each other independently of their locations. Intermediate nodes will be relaying packets at the routing layer without profiting of them at the content sharing layer. From here came our idea to test other strategies where the scope of the neighborhood is reduced.
- For a reduced scope of sharing like 2 hops, one can see that the download time decreases dramatically. This can be explained in the dense cases by the reduced routing overhead. But, in the sparse cases, this can also mean that the overlay is disconnected and that only a sub-set of peers can finish downloading the content. Globally, it is always beneficial to reduce the distance between

P2P neighbors provided that the overlay stay connected.

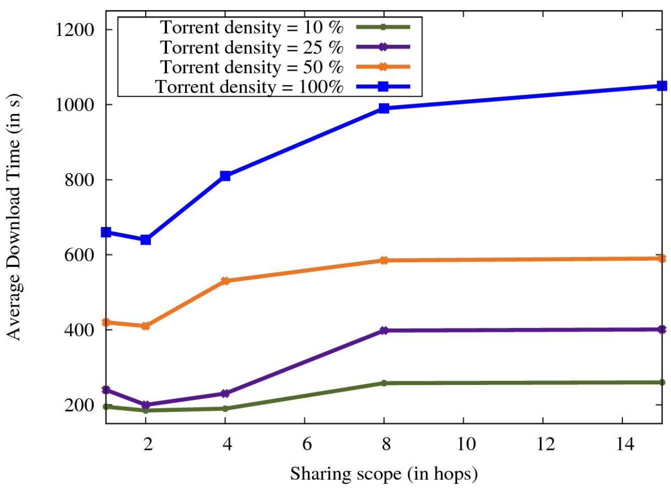


Figure 3.3: Download Time Vs Sharing scope (Different densities - ORBIT experiments)

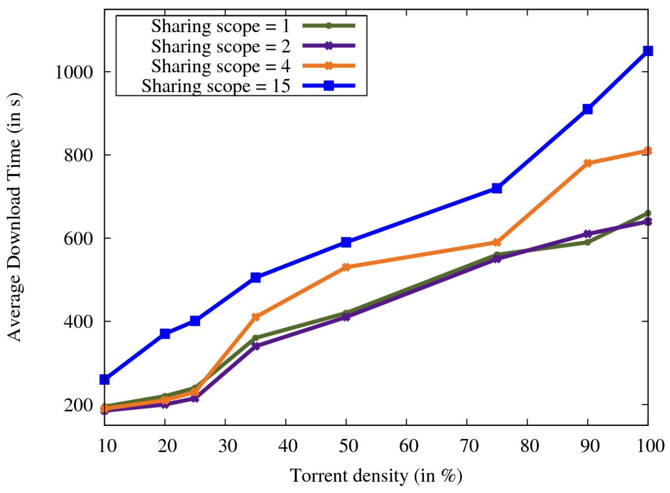


Figure 3.4: Download time Vs. Torrent density (Different sharing scopes - ORBIT experiments)

To measure the impact of limiting the sharing scope on the connectivity of the overlay, we study the completion ratio, which is the percentage of peers that have downloaded all the pieces of the file. We plot in Figures 3.5 and 3.6 the ratio of completion respectively as of function of the sharing scope and the Torrent density. One can easily notice that the completion ratio of the classical version of BitTorrent (a sharing scope equal to 15) is always equal to 100% as all peers are neighbors of

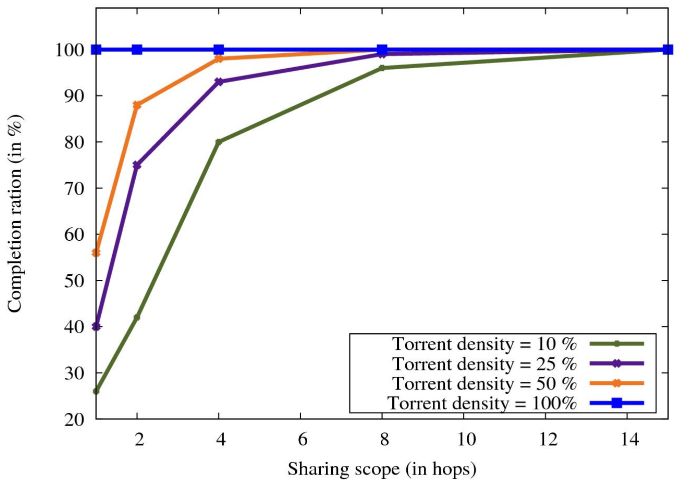


Figure 3.5: Completion ratio Vs. sharing scope (Different densities - ORBIT experiments)

each other. This is at the expense of larger download times as we have already seen. Moreover, for a Torrent density equal to 100 %, the network is always connected for all sharing scopes then we always record a 100% completion ratio. Now, when we limit the scope of the neighborhood in the case of sparse Torrents, the completion ratio decreases subsequently. Peers start to be disconnected from each other and from the initial seed of the file. Hence, it is necessary to find a neighborhood selection strategy that guarantees the connection of the overlay and at the same time minimizes the number of hops between neighbors.

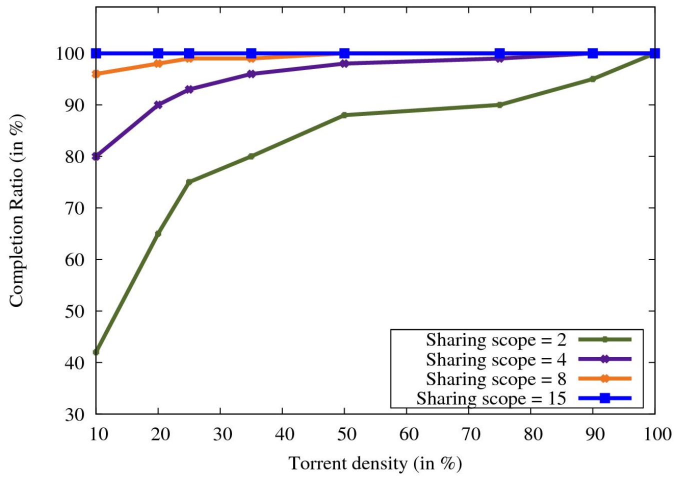


Figure 3.6: Completion ratio Vs. Torrent density (Different sharing scopes- ORBIT experiments)

3.4 Piece diversity ameliorates sharing opportunities

In the previous section, we concluded that the best way to select neighbors is to reduce the scope of the neighborhood as it yields the best download times, but in some cases it disconnects the sharing overlay. In this section, we consider another important metric, the sharing ratio, which measures the degree of cooperation of peers. An ideal content sharing protocol must divide the sharing load fairly among the set of peers. Differently speaking, a peer should upload the same amount of data it downloads. We define the sharing ratio between a couple of peers i and j as follows:

$$R_{ij} = \frac{min(D_{ij}, U_{ij})}{max(D_{ij}, U_{ij})} \tag{3.1}$$

, where D_{ij} is the amount of data that peer i downloads from peer j during the Torrent lifetime. This ratio measures the magnitude of the reciprocity of data between the two peers. A value nearing null means a one-way propagation of data. The fair sharing ideal case is obtained when the sharing ratio is equal to 1.

In this section, we try to answer the following questions: By limiting the scope of sharing, are we limiting the sharing opportunities between peers? Is there fair sharing among them in this case? Is there a piece diversity problem? When sharing scope is very limited, the pieces of the content will most likely propagate from the initial seed to the farthest peers in a unique direction via other peers in between. Far peers do not have original pieces to provide to upstream nodes that are closer to the initial seed. That is why peers fail to reciprocate data with each other, and hence, the load of sharing is not equally divided among them. In general, the farther the nodes are from the initial seed, the fewer packets they will have to send. Moreover, there will be no diversity of pieces in the network. The same pieces will propagate from one neighborhood to another, which cannot result in a fair exchange.

To strengthen this claim, we plot in figures 3.7 and 3.8 the sharing ratio as a function of respectively the sharing scope and the Torrent density for the same experiments used in the previous section. From these figures, one can observe the following:

- The sharing ratio increases with the Torrent density for all sharing scopes. This can be explained by the fact that for sparse Torrents, there are few peers in the neighborhood of each others and the fact that pieces of the content travel from the source of the content to near peers quickly and they take a lot of time to reach far ones. Generally, no exchanges are done between peers due to the absence of intermediate peers and because no neighboring nodes are interested in pieces, which the peers receive. For higher densities, pieces can find alternative paths to propagate in the network and then, the farther peers are from the source of the content the more peers in their neighborhood will have original pieces to exchange with them.

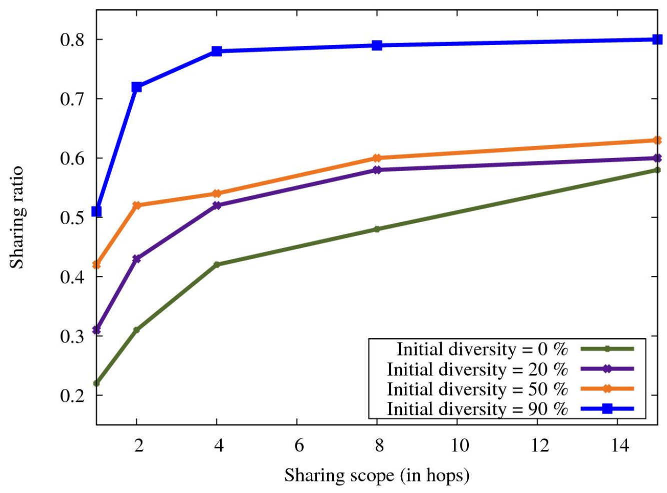


Figure 3.7: Sharing ratio Vs. Sharing scope (Different initial diversity - ORBIT experiments)

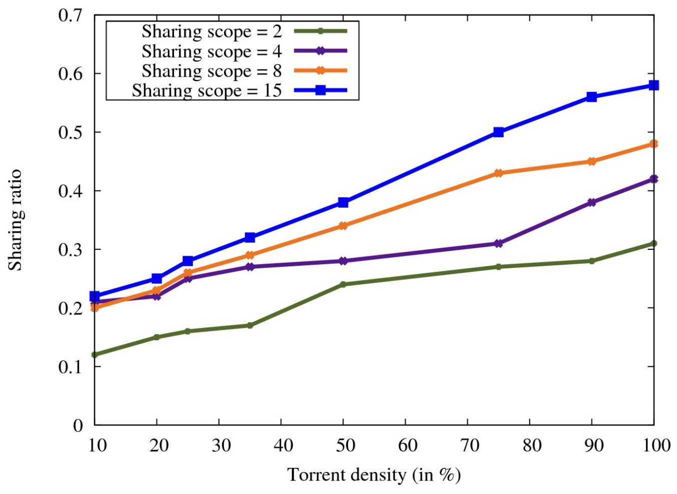


Figure 3.8: Sharing ratio Vs. Torrent Density (Different scopes of sharing - ORBIT experiments)

- When the sharing scope is decreased, the sharing ratio goes down dramatically for all Torrent densities. The cases where the scope is set to one or two hops yield the lowest sharing ratios mainly in case of dense Torrents, which can be explained by the fact that pieces propagate like a wave from the initial seed to the farthest peers. The resources of the network are not fully used since nodes wait for pieces to arrive to their neighborhood and rarely have original pieces to reciprocate with their neighbors. The P2P file sharing application then behaves like a simple piece relaying protocol that ignores the parallel transmission capabilities of the network and the distribution of the load among peers.

- For large sharing scopes, for instance the classical BitTorrent case, the sharing ratio is still lower than 0.6 because the routing overhead is big and very few pieces can be downloaded during a choking slot, mainly when the number of hops between neighbors is high.

As limiting the sharing scope decreases the diversity of pieces in the network, we wanted to study the impact of this diversity on both the sharing ratio and the download time. Up to now, we supposed that at the beginning of an experiment, only one peer (the seed) holds the pieces of the content. In the following experiments, we distribute one copy of the content equally among a sub-set of the peers before starting the sharing session. The percentage of peers having pieces of the content to the total number of peers is called the initial diversity of pieces. In figures 3.7 and 3.9, we plot the sharing ratio as a function respectively the sharing scope for different initial diversities of pieces and the initial diversities of pieces for different sharing scopes. From these figures, one can make the followings conclusions:

- Increasing the diversity of pieces increases the sharing ratio for all scopes of sharing. This is mainly due to the fact that peers will have original pieces to exchange between each others. Hence, the initial diversity of pieces increases the interest that the peers have in each other.

- For a high initial diversity of pieces (90 % for example), the case of a small sharing scope is no longer the one that yields the lowest sharing ratio and it seems that all sharing scopes engender good sharing opportunities.

From a sharing prospective, it is important to have a good diversity of pieces to encourage fair exchanges between peers. This can not be obtained when we limit the sharing scope and that at the beginning of the session only one peer has the complete content. But, adding some diversification artificially before starting the sharing session helped increasing the sharing opportunities. But, increasing the diversity of pieces for the same scope sharing does it ameliorate the download time? In figures 3.10 and 3.3, we plot the download time as a function of respectively the sharing scope for different initial diversities of pieces and the diversity of pieces for different sharing scopes. Surprisingly, the results show an amelioration of the

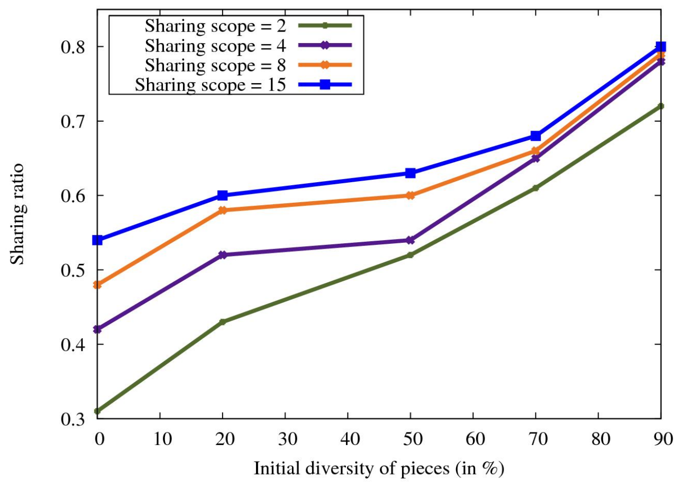


Figure 3.9: Sharing ratio Vs. Initial diversity (Different scopes of sharing - ORBIT experiments)

download time for all sharing scope when the initial diversity of pieces increases. Furthermore, a sharing scope equal to 2 yields the best download time for all initial diversities. Consequently, a sharing scope equal to 2 with a high initial diversity of pieces gives the best download time and sharing ratio. It has the advantage of reducing the routing overhead while profiting from the collaboration between nodes to better utilize the network resources.

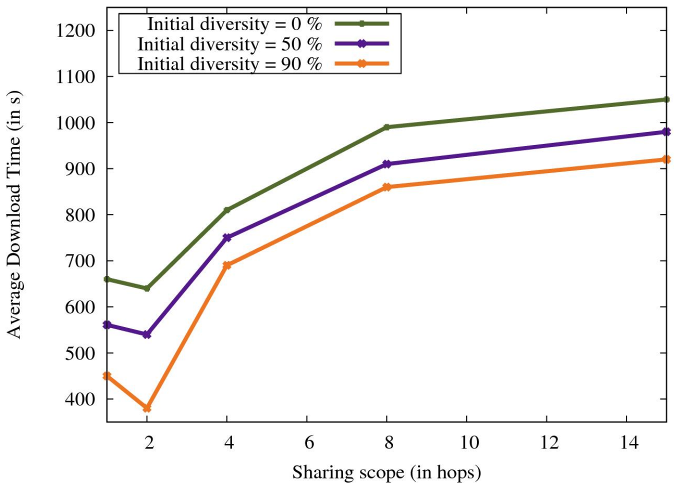


Figure 3.10: Download Time Vs. Sharing Scope (Different initial diversities - ORBIT experiments)

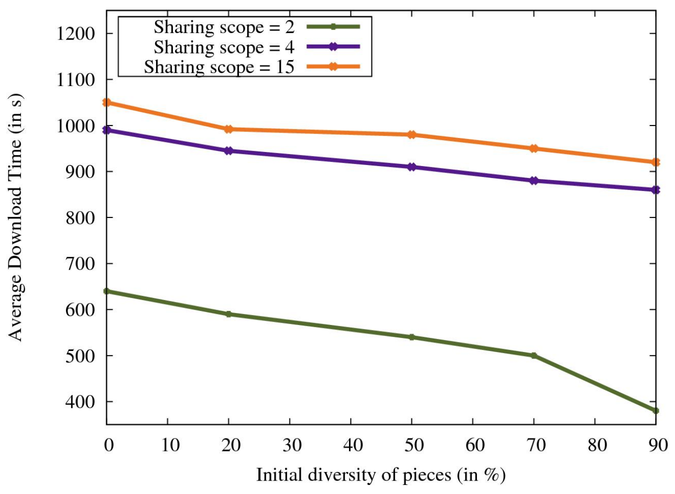


Figure 3.11: Download time Vs. Initial diversity (Different scopes of sharing - ORBIT experiments)

3.5 Mobility increases sharing opportunities

In the previous sections, we supposed that the network was fixed. In the case of mobility of nodes, two main factors must be considered. On one hand, the mobility naturally increases the diversity of pieces, since the neighborhood of a peer is changing while moving. In this case, one can hope that there will be enough sharing opportunities and hence there will be no need for sending pieces to far away nodes to boost diversity. On the other hand, as long paths suffer from bad performances in mobile ad hoc networks, it would be better to have a short range of sharing and diversification. The simulation results show indeed that the mobility has a beast and a beauty. The beast is that it increases packet losses over long multi-hop paths, thus it makes it almost inefficient to send pieces to faraway peers. The beauty is that the mobility efficiently improves piece diversity since peers are continuously exchanging new pieces with the new peers they meet while moving across the network. We give an idea on the results in Figures 3.12 and 3.13, where we compare respectively the download time and the average sharing ratio for both the fixed and the mobile scenarios. In these simulations, the Random Way point mobility model [Hyytia 2006] is used. The speed of nodes is set to 10 m/s and two pause times have been considered (2s and 10s). We can observe how the best performance is obtained for mobile networks limiting the sharing scope. Indeed, the Figures show the sharing ratio has the highest values (approaching 0.8). These results can be easily linked to the diversity of pieces introduced by the mobility of nodes and the sharing opportunities it raises. One can conclude that in mobile environments only the scope of the neighborhood is to be limited. Furthermore, it is useless to provide any supplementary effort for the diversification because with the instability of paths, sending pieces to faraway peers in MANETs only increases the overhead and worsens the performances.

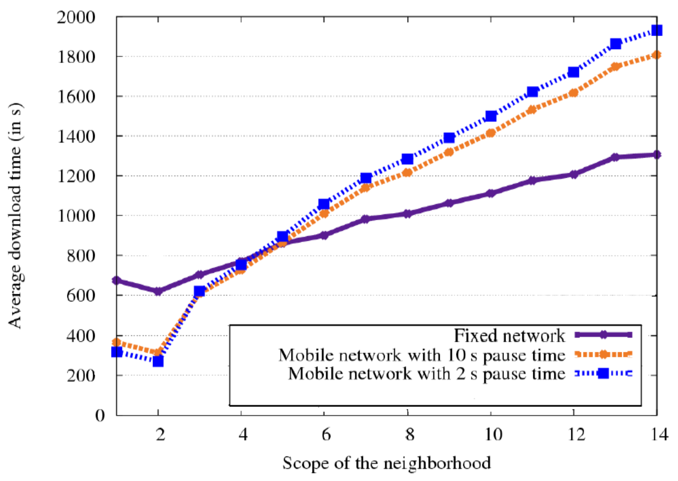


Figure 3.12: Download time Vs. Scope of the neighborhood(Mobility of nodes - NS-2 Simulations)

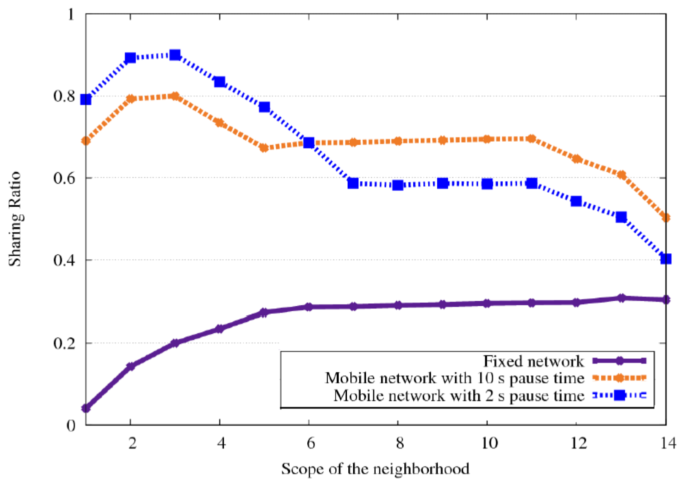


Figure 3.13: Sharing ratio Vs. Scope of the neighborhood (Mobility of nodes - NS-2 Simulations)

3.6 Conclusions and open issues

In this chapter, we studied the performance of BitTorrent in wireless ad hoc networks. Our main findings were that limiting the scope of sharing decreases dramatically the routing overhead and as a consequence, it shortens considerably the download time. However, we showed that, for sparse Torrents, this limitation disconnects the sharing overlay. In this case, some of the peers will be isolated from other and will never get all pieces of the file. Moreover, when considering the sharing opportunities, limiting the sharing area of peers decreases the diversity of pieces in the network and hence, one obtains low sharing ratios. Nevertheless, when nodes start moving, nodes exchange original pieces when they meet. In this mobile case, one obtains both better download time and sharing ratios. In fact, mobility boosts the diversity of pieces in the network.

In chapter 4, we design BitHoc, a BitTorrent variant that adapts to the constraints of wireless ad hoc networks. This protocol offers solutions to routing overhead, overlay connectivity and piece diversity problems. Moreover, BitHoc adapts the BitTorrent mechanisms to different densities of interest, to cross traffic and to the mobility of nodes.

Chapter 4

A P2P Content Sharing Protocol for Wireless Multi-hop Networks

Contents

Whereas efficient content retrieval and localization techniques for wireless ad hoc networks have been widely studied in the literature [Klemm 2003][Das 2004], the data plane of the content sharing problem is still in its first steps. The majority of previous studies focus on the particular case of a single sharing session running over a fixed and dense wireless network. They consider that all nodes are interested in sharing the same content and they take the BitTorrent protocol [Bit 2010b] as a reference. For instance, [Michiardi 2007] aims to ameliorate the global download time by reducing the routing overhead. The proposed idea is to make peers only concentrate on their nearby neighbors. However, one can easily notice from the study done in Chapter 3 that in the case of a limited scope of sharing the replication burden is unequally distributed among peers and that there is a poor transmission parallelism in the network. Moreover, the lack of piece diversity in the network generates low sharing opportunities. This is contradictory to the goals of BitTorrent and does not respect the constraints imposed by wireless ad hoc networks. Instead, we proposed in [Sbai 2009a][Sbai 2008] to replicate pieces of the content across the network at low rate to increase the diversity of information and improve the parallelism. Although these policies register better download times and point to some new directions, they are limited to some specific cases that need to be generalized to illustrate clearly the relationship between content replication, user performance, fairness and overhead on the underlying network.

On one side there is a need to diversify the content in the network to improve user perceived quality and enforce fairness, and on the other side, this diversification is costly because of the multi-hop routing. In our preliminary study [Sbai 2009a], we investigated the optimal balance between sharing and diversification efforts. The study done in [Sbai 2009a] is carried out by NS-2 simulations [NS- 2010] and is limited to the case of dense P2P networks with one file to share. Our main observation was that leechers (peers downloading the content) should concentrate their sharing effort on their physical neighborhood whereas seeds of the content should take into consideration the diversification of the content pieces across the network. The diversification area by one seed must be taken wider than the sharing area but no more than the distance allowed by a multi-hop TCP communication.

In this chapter, which is an extended version of our paper [Sbai 2010], we generalize this first result to the case of sparse P2P networks and multiple shared files in parallel. Our evaluation is based on extensive experiments over the ORBIT platform [ORB 2010]. In chapter 3, we considered the case of a single Torrent in the network while varying the density of peers and studied the impact of limiting the scope of sharing on the download time and the connectivity of the sharing overlay. We noticed that the routing overhead is decreased at the expense of the guarantee of finishing the download. Indeed, in the sparsest cases, peers will be isolated and cannot finish downloading the content. Therefore, in a first part of this chapter, we propose to organize peers in a minimum spanning tree which guarantees the connectivity of the overlay and limited routing distances between peers. The NS-2 simulations [NS- 2010] and the experimentations on the ORBIT platform [ORB 2010] both show an improvement in the download time and the completion ratio when this strategy is deployed.

Furthermore, in chapter 3, we studied the impact of limiting the scope of sharing on the diversity of pieces in the network. In fact, reducing the neighborhood over the spanning tree impacts negatively this piece diversity. Pieces of the content propagate in one way from the initial seed to the edge peers resulting in low sharing ratios. In a second part of this chapter, we show that by adding a diversification effort to seeds of the content, we improve considerably the sharing ratio while preserving the low routing overhead of the limited P2P neighborhood. Again this will be confirmed by NS-2 simulations and ORBIT experimentations. Finally, we study the tuning of the diversification effort. Mainly, we design an algorithm to adapt the scope of the diversification area of a seed to the changes in the network settings, to congestion and to mobility of nodes. On one hand, we show that when many concurrent Torrents run in the same network, the gain brought by the diversification is negligible since the network is very congested. On the other hand, we show that there is no need for diversification in a mobile scenario since nodes will exchange different pieces of the content while moving. Simulations and experimentations show that our algorithm adapts the diversification scope in both these cases.

The remainder of this chapter is organized as follows. Section 4.1 shows the benefits of using our minimum spanning tree strategy in case of sparse Torrents. Section 4.2 investigates the gain obtained by diversifying the pieces of the content in the network. We propose in Section 4.3 our neighborhood selection and choke algorithms. In Section 4.4, we propose an adaptive algorithm that selects automatically the optimal scope of diversification in case of multiple Torrents and in case of different mobility speeds. The conclusions and the perspectives of this paper are dressed in Section 4.5.

4.1 Reducing routing overhead and keeping the overlay connected

In Chapter 3, we took the well-known BitTorrent protocol as a baseline and have shown through experiments that the best neighborhood selection strategy, which guarantees the best download time, is the strategy that limits the sharing scope of the classical Internet version of BitTorrent. However, this amelioration of the download time is at the expense of fewer peers that complete the download, particularly in the case of very low Torrent densities. In fact, limiting the sharing to nearby peers engenders less routing overhead. Unfortunately, when the Torrent is very sparse, peers that are far from each other will be isolated in separate islands and will never finish the download. The neighboring sharing area should be extended in this latter case to ensure connectivity of peers. A tradeoff then emerges between reducing the sharing area to reduce overhead and increasing it to ensure connectivity.

In this section, we define a new neighborhood selection strategy that considers that the peers interested in sharing the same content are organized in a minimum spanning tree in terms of the number of hops. The construction of the minimum spanning tree is done by the membership management protocol that we have proposed in [Sbai 2009b] and we describe in details in chapter 5. According to this strategy, a peer selects all peers located at one logical hop in the tree as its neighbors and adds all peers located at a routing distance (layer 3 distance) shorter than the number of routing hops to 1-hop neighbors in the spanning tree. We choose the minimum spanning tree for two main reasons. First, this tree is a structure that limits the number of hops between peers, which translates to less routing overhead. Second, this tree has the advantage of being a structure that connects all the peers together; hence all peers can complete the download of the file. We evaluate the gain brought by constructing the P2P neighborhood over this minimum spanning tree connecting peers through experiments.

In Figure 4.2, we plot the completion ratio as a function of the Torrent density for our tree-based methodology compared to the limited routing scope strategy. As the minimum spanning tree guarantees the connectivity of the overlay, the completion ratio is equal to 100 % for all Torrent densities. However, the limited sharing scope method, as we described earlier, has small completion ratios for low Torrent densities. Moreover, a minimum spanning tree, apart from ensuring the connectivity of the overlay, is a logical structure, faithful to network topology that minimizes the distances between peers. By applying the spanning tree strategy, one can obtain the best download time as Figure 4.1 shows. This strategy indeed adapts the neigh-

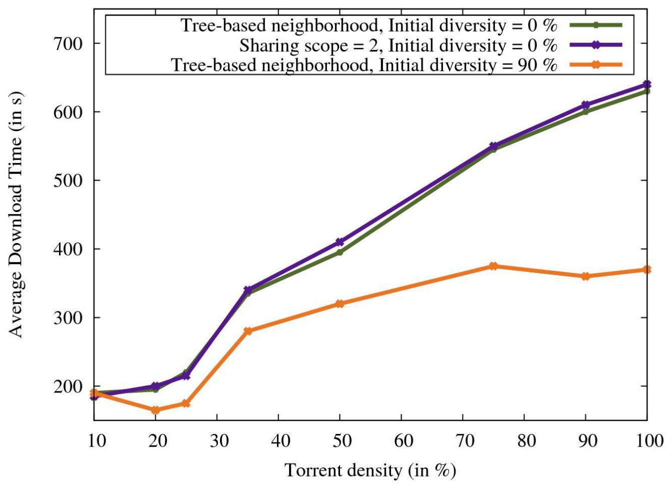


Figure 4.1: Download time Vs. Torrent density (Tree based neighborhood- ORBIT experiments)

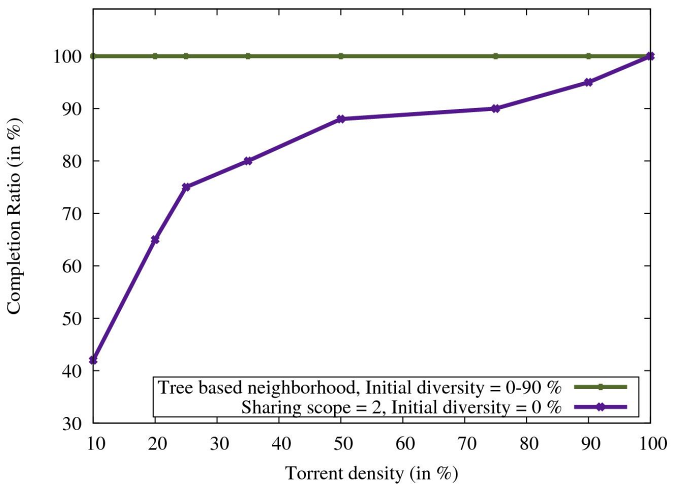


Figure 4.2: Completion ratio Vs. Torrent density (Tree based neighborhood- ORBIT experiments)

borhood selection to the density of the overlay. It is almost equivalent to a sharing scope equal to 2 in the dense cases but it has the further advantage that it connects far away peers in the sparse cases, hence ensuring that all peers get the file.

4.2 Impact of piece diversity on the tree-based strategy

In Chapter 3, we showed when some pieces of the content are distributed initially in the network, better sharing opportunities between peers are obtained. The same result is verified for the tree-based neighborhood selection mechanism we introduced in the previous section. We plot in Figures 4.1 and 4.3 respectively the download time and the sharing ratio as a function of the Torrent density for different initial diversities of pieces. They show that the tree-based neighborhood strategy yields both the best download time and sharing ratios for high initial diversities of pieces.

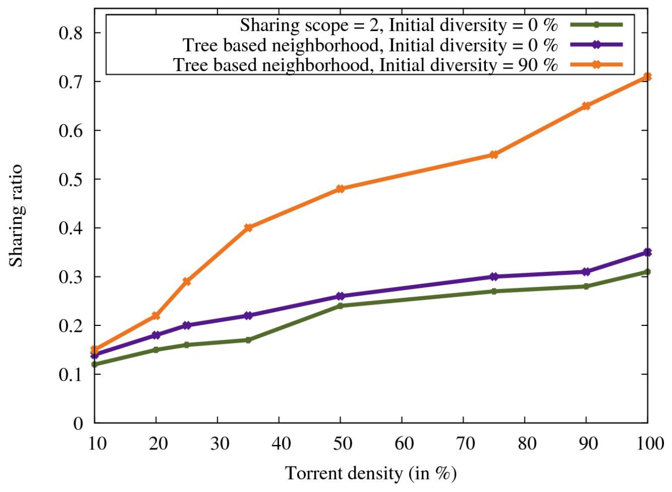


Figure 4.3: Sharing ratio Vs. Torrent density (Tree based neighborhood- ORBIT experiments)

So, there is a tradeoff between increasing the piece diversity by sending pieces to far nodes and reducing the routing overhead by limiting the sharing scope. In the following section, we prove that this is possible to solve this trade-off by decoupling the sharing effort from the piece diversification effort. Mainly, we introduce a new algorithm to increase the diversity of pieces in the network at a limited routing cost, and in parallel we limit the sharing scope following the tree-based neighborhood strategy in order not to overload the network. In this way, we can do better than the simple small scope case by having a better diversity of pieces, and hence more parallel transmissions and better reciprocity. At the same time, we are better than the simple large scope case in terms of sharing, because we can diversify pieces in the network to improve sharing, without suffering from the routing overhead problem.

4.3 Our choke algorithm and neighbor selection strategy

In the two previous sections, we presented two problems related to content sharing in wireless ad hoc networks. We now face the following dilemma. On one hand, decreasing the scope of sharing ameliorates the download time but leads to very weak parallelism in the network due to the lack of piece diversity. On the other hand, increasing the scope of sharing increases the diversity of pieces in the network, but at the cost of more routing overhead and worse download time. In this paragraph, we present our solution to this dilemma. Our objective is to come up with a joint solution for the routing overhead and piece diversity problems. In designing our new neighbor selection strategy and choke algorithm, we took into consideration the following points:

- Data transfers between distant peers suffer from very poor performances in wireless ad hoc networks. Hence, a leecher has no incentives to send pieces of the content to far nodes, as they will not be able to serve him back with a high throughput. Moreover, leechers that are far from the initial seed have less original pieces to reciprocate them with their nearer leechers. Considering this, we decided that in our effective neighbor selection strategy, only seeds send pieces to far peers. Indeed, a seed is a volunteer peer that serves others without expecting any return from them. The leechers have more incentives to concentrate on peers located in their close neighborhood, provided that there are original pieces to share with them.

- If all seeds send pieces to far nodes at the same time, the routing overhead will be large again and performance will decrease. In our solution, we subdivide the piece diversification effort among seeds both in space and time.

- If a seed cannot send a complete piece to the selected peer during the choking slot, the gain in diversity will be null since the smallest unit that a peer can share with others is the piece. In our solution, we limit the scope of diversification of seeds to the number of hops allowing the transfer of a complete piece. Pieces are spread in other parts of the network by other peers becoming seeds and deciding to stay in the torrent.

Our new neighbor selection strategy and choke algorithm can be summarized as follows:

- **Leecher behavior:** In the leecher state, peers concentrate on their nearby neighborhood. This neighborhood is constructed following the tree-based strategy as it is proved to be the best regarding the routing overhead and connectivity of the overlay. A leecher maintains 4 simultaneous active outgoing connections. The first 3 connections are dedicated to best uploaders among peers in the sharing scope and the fourth connection consists in an optimistic unchoke allowing to discover new upload capacities and the bootstrap of the sharing. The fourth peer is chosen randomly among leechers within the sharing area. The selection is done at the end of each choke time slot. Except the limitation of the sharing scope using the spanning tree, this is globally the classical BitTorrent algorithm for leechers.

- **Seed behavior:** In the seed state, peers dedicate their first 3 connections to serve leechers within their tree-based sharing areas. These are the connections dedicated to injecting the content in the network by starting from the small sharing area. The fourth connection of a seed is mainly dedicated to diversify pieces in an area wider than the sharing area (See Figure 4.4). This area is called the diversification area of the seed and contains all peers not belonging to its sharing area and that are located at a routing distance lower than the diversification scope. The routing scope of diversification is determined dynamically by observing the range of piece transmissions (see Section [AdaptiveAlg]). In the following paragraph, we study the optimal way to choose a leecher in the diversification area and we describe how the fourth connection is used when there is more than one seed within the same diversification area.

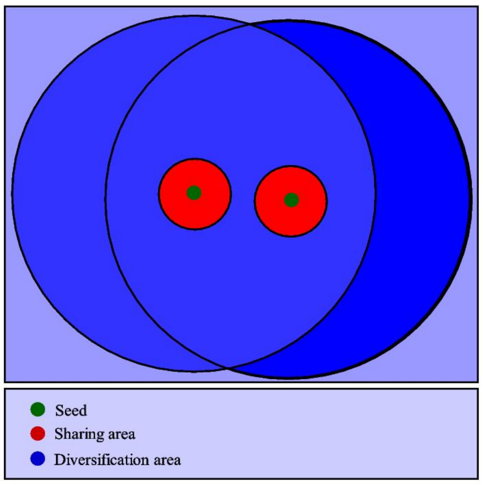


Figure 4.4: sharing and diversification areas

4.3.1 Dividing the diversification effort between seeds

Sending pieces to far nodes engenders a big routing overhead. Hence, the diversification effort must be divided between all seeds, both in space and in time. In our solution, each seed is responsible for its own diversification area and does not have to serve the whole network. Moreover, when there are many seeds within the diversification scopes of each other (for example when other peers finish the download and decide to stay), our solution reduces the routing overhead of the fourth connection of each of them, which is dedicated to diversification, by the number of seeds in its diversification area. This is done as follows. The seed pauses for a number of slots equal to the number of seeds in its diversification area between every two diversification time slots. During the pause, the seed can serve leechers in its tree-based sharing area. This scheduling is repeated periodically and follows the evolution of the number of seeds. In this way, the total diversification overhead is kept constant as there are more and more seeds in the network. Figure 4.5 illustrates the scheduling of the fourth connection when two seeds exist in the same diversification area.

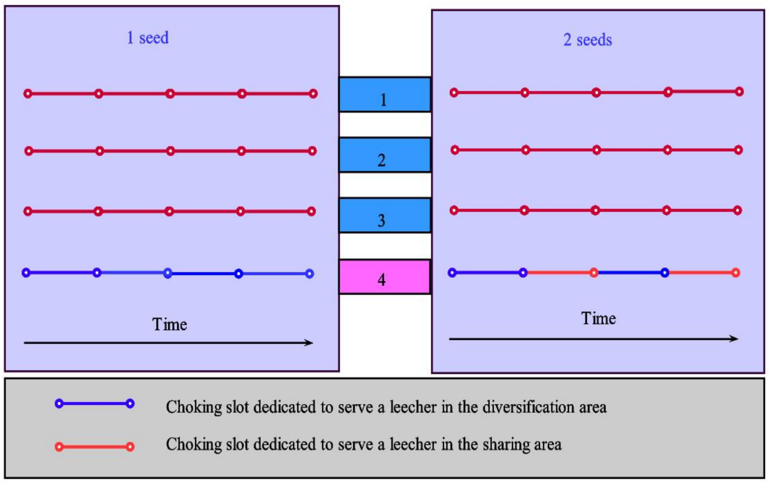

1 seed
2 seeds
1
2
3
4
Time
Time
Choking slot dedicated to serve a leecher in the diversification area
Choking slot dedicated to serve a leecher in the sharing area

Figure 4.5: Scheduling the fourth connection of a seed

To validate this choking strategy of the seed, we run some experiments where seeds decide with a probability p whether the current slot is a diversification one. This is done for the four outgoing TCP connections. The objective is to prove that p equal to $\frac{1}{4}$ is enough to have a good diversification of pieces while keeping a low level of routing overhead. Two versions have been tested: The first version does not adapt p to the number of seeds in the same diversification area. The second version starts with an initial probability p and then divides it with number of seeds in the same diversification area. Figure 4.6 plots the average download time as a function of the diversification probability p for both versions. It shows that adapting the diversification probability to the number of seed yields lower download times than keeping it independent of this number. A starting probability equal to $\frac{1}{4}$ is the lowest value in the optimal plateau. Hence, we validate our choice of having one out of four connections dedicated for the diversification effort and our scheduling mechanism that stops diversifying for a number of slots equal to the number of seeds.

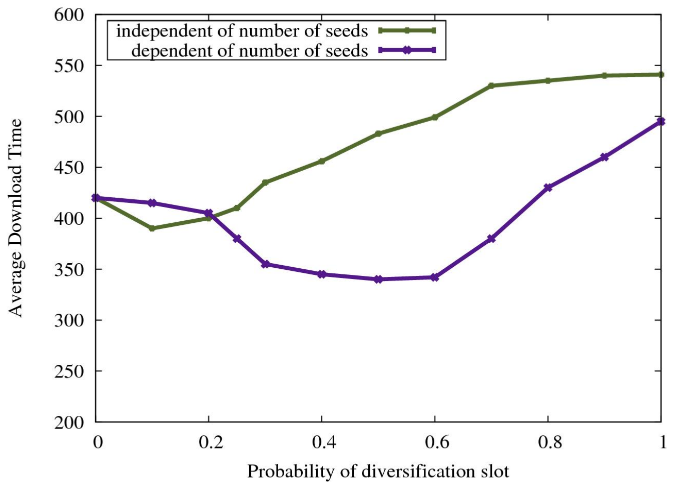


Figure 4.6: Scheduling diversification (Probabilistic study- ORBIT experiments)

4.3.2 Optimal diversification-neighbor selection strategy

In this paragraph, we look for the best strategy used by seeds to select leechers in their diversification areas at time slots. The goal is to maximize the sharing ratio while minimizing the average download time. As a first step, we suppose that all nodes of the network are interested in sharing the same content. Let's note the sharing scope (obtained by the tree-based strategy) of a seed as S_s and its diversification scope as S_d. Searching the optimal strategy, we define a parametric general probability distribution to tune the lecher selection and we study, through experiments and by varying the parameter of the distribution, the impact of the different strategies on the torrent performances. We model the probability to select a peer located at h hops from a seed in its diversification scope ($S_s < h \leq S_d$) as follows:

$$p(h) = \frac{h^{\alpha}}{\sum\limits_{l=S_s+1}^{S_d} N_l l^{\alpha}} \tag{4.1}$$

Where N_l is the number of peers located at l hops and α is a parameter of the probability distribution. The sum of this probability function over all peers in a diversification area is clearly equal to 1. By setting α to 0, we can obtain the uniform probability distribution where peers are selected with the same probability independently of their location. For large positive values of α, the probability to select the farthest peers becomes close to 1, and that to select peers near to the seed almost null. For large negative values of α, the opposite occurs; the seed diversifies pieces over peers close to it. This parameter α then covers a large set of strategies, and its optimal value should point us to the optimal leecher selection strategy to use for diversification purposes. Next, we seek this optimal value using extensive experiments. Figure 4.7 plots both the download time and the sharing ratio as a function of the parameter α. For large negative values of α, the download time is maximal and tends to the one obtained without diversifying the pieces. For large positive values of α, one can obtain a better performance since there is the introduction of some diversity of pieces in the network but the concentration is only on lechers located at the edge of the diversification area. This is below the optimal because of the routing overhead and an inefficient spatial distribution of pieces. Our main observation is that a value of α equal to 0 gives the best performance. Indeed, for this optimal value, seeds distribute pieces uniformly in the network and then boosts fair sharing among peers and transmission parallelism while having a reasonable average routing overhead.

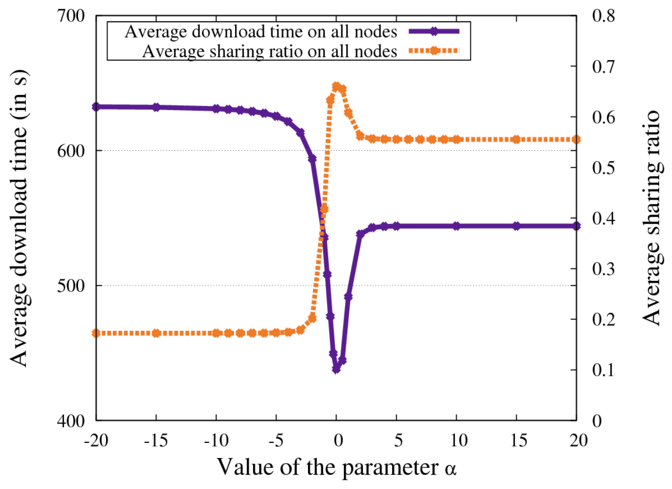


Figure 4.7: Finding the best diversification strategy (Spatial distribution - ORBIT experiments)

To validate the same results for the case of sparse Torrents, we plot in figures 4.8 and 4.9 respectively the download time and the sharing ratio as a function of the Torrent density for different values of the parameter α. The results show that a value of α equal to 0 gives the best performance both in sharing and download time perspectives. The out-performance of a uniform selection of leechers increases with the Torrent density since the routing overhead increases.

4.4 Adaptive diversification scope algorithm

The results shown up to now have been obtained for a diversification scope set to 10. In this section, we study the impact of this scope both on download time and sharing opportunities. Figure 4.10 plots the average download time over all nodes in case of a Torrent density equal to 100% as a function of the scope of diversification. It shows that for small values of this scope, there is not enough diversity introduced into the network. Hence, the download time worsens. For large diversification scopes, again the download time worsen for the simple reason that TCP becomes

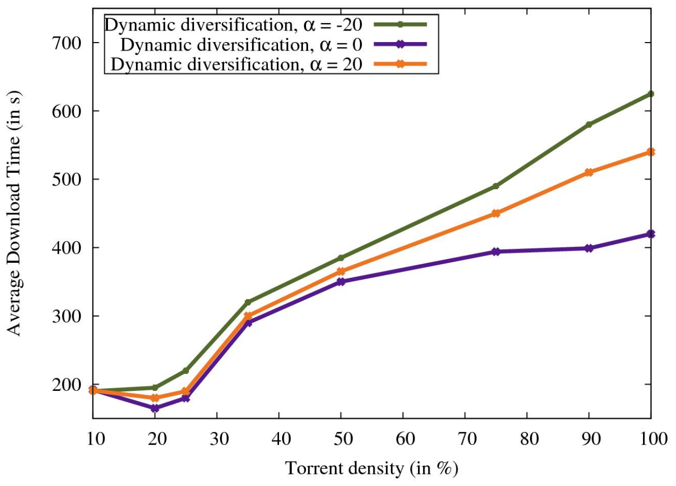


Figure 4.8: Download Time Vs. Torrent density (ORBIT experiments)

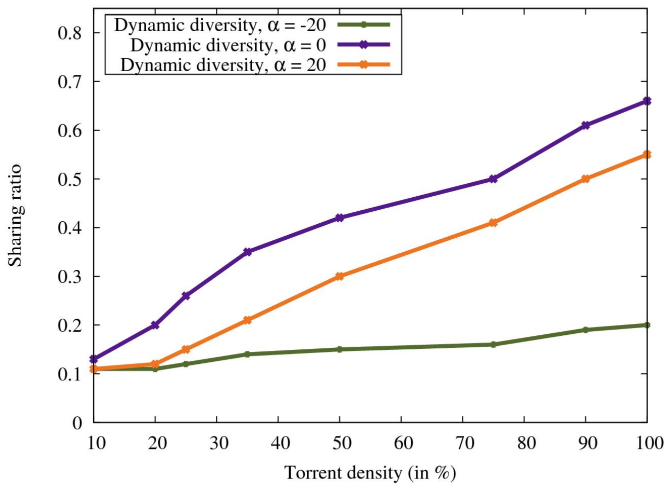


Figure 4.9: Sharing ratio Vs. Torrent density (ORBIT experiments)

unable to send entire reusable pieces far away from the seed. It is clear that in our settings a diversification scope around 10 routing hops away from each seed leads to best performances. This should be the largest scope where entire content piece could be sent.

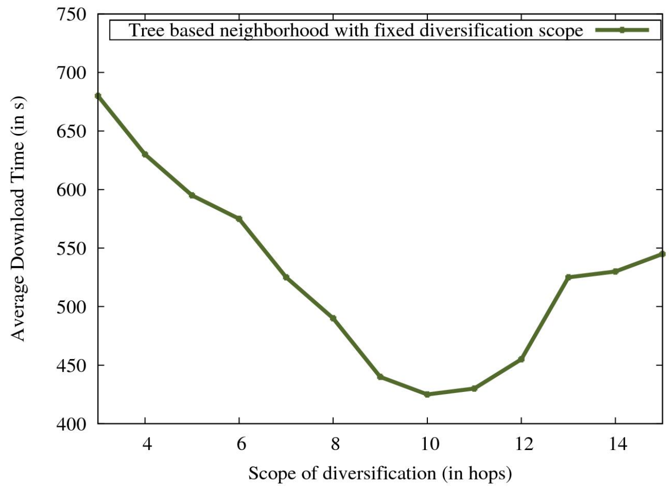


Figure 4.10: Download Time Vs. Scope of Diversification (Searching the best diversification scope - ORBIT experiments)

As we want our protocol to be general and adaptive, we propose to adjust the scope of the diversification area automatically as a function of network conditions. We measure in each diversification slot the number of pieces sent to the selected peer. If no complete piece has been sent, this means that we have reached the range of pieces and that one needs to decrease the diversification scope. And if one or more pieces can be sent, this means that one can increase the diversification scope, hopefully reaching farther peers. The algorithm that we propose to adjust this diversification scope is the following:

- At the beginning of the sharing session, the diversification scope (DS) is set to an average value equal to $\frac{3+h_m}{2}$, where h_m is the maximum number of hops in the network. It is a first guess of the diversification scope. It is taken equal to an average value since the scope of diversification can range from 3 hops to h_m hops.

- Each time a peer sends diversification packets to a node located at h hops ($h \leq DS$), it updates the value of DS depending on whether it can send a complete piece to the destination or not.

  - In the case where one or more complete pieces can be sent, DS is increased to $\beta DS + (1-\beta)(DS+1)$, where β is an empirical value chosen in our experiments equal to $\frac{3}{4}$.

– In case no complete piece can be sent, DS is decreased to $\beta DS + (1 - \beta)(h - 1)$, where β is the same empirical value chosen in our experiments equal to $\frac{3}{4}$.

The objective of any adaptation of this type is to absorb transitory network congestion while allowing fast convergence to the appropriate diversification scope. Figure 4.11 draws both the diversification scope adapted to the choking scope during a sharing session of 400s. It shows that the diversification scope converges automatically to the optimal scope (10 hops in our experiments).

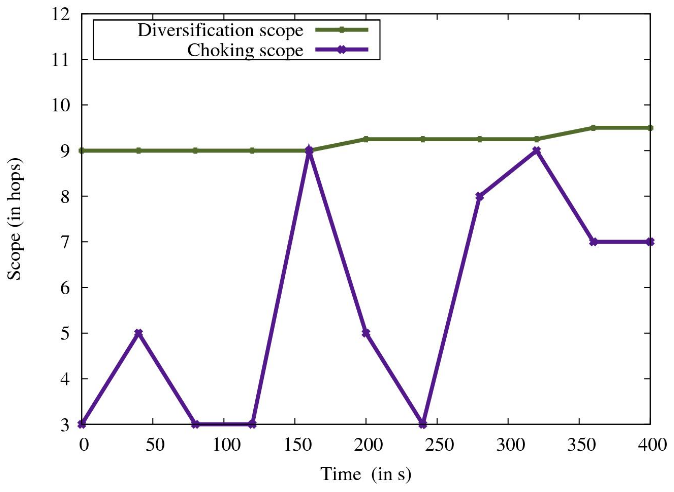


Figure 4.11: Diversification scope and choking scope (Adapting the diversification scope- ORBIT experiments)

To evaluate the gain obtained by diversifying the pieces and applying our dynamic scope of diversifications, we compare in figures 4.12 and 4.13 the tree-based neighborhood with and without initial diversity of pieces to a tree-based neighborhood with our adaptive choke algorithm. These figures show that our protocol performs better then the narrow neighborhood without initial diversity from both download time and sharing perspectives. Furthermore, they prove that diversifying pieces artificially is almost equivalent to having an initial uniform distribution of pieces in the network. Hence, our protocol diversifies the pieces while yielding the lower routing overheads.

4.4.1 Adapting the diversification scope to network congestion

In this paragraph, we study the impact of having many Torrents together in the same network. Our aim is to evaluate the impact of the inter-Torrent congestion on the neighbourhood selection strategy, mainly on the diversification effort. We vary the number of Torrents concurring in the network from 1 to 5 setting each Torrent's density to 50%. Figure 4.14 plots the download time as function of the number of

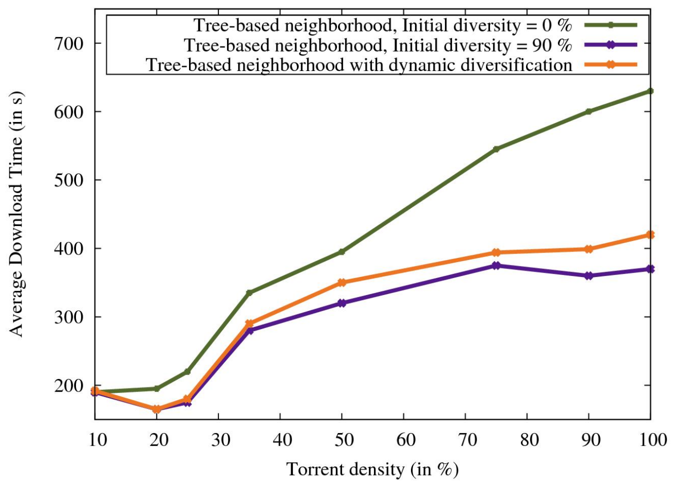


Figure 4.12: Efficiency of the diversification mechanism (Download time)(Dynamic diversification Vs. Initial diversification- ORBIT experiments)

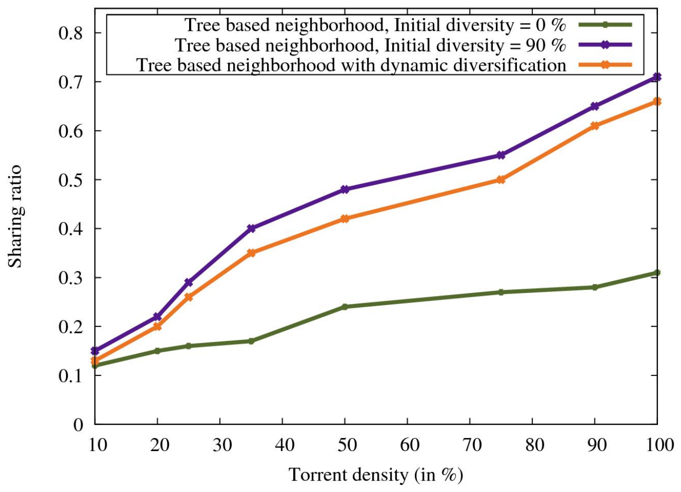


Figure 4.13: Efficiency of the diversification mechanism (Sharing ratio)(Dynamic diversification Vs. Initial diversification- ORBIT experiments)

Torrents for the spanning tree strategy with our adaptive scope of diversification compared to that with a fixed diversification scope. One can easily notice that the adaptive version has better performance than the fixed scope version mainly one the number of Torrents increases. One can notice that when there is a few number of Torrents (1 or 2), the diversification yields a decrease in the download time. However, for a larger number of Torrents, the download time worsens and it is better not to diversify. This is because when the network gets congested by other Torrents, pieces can no longer be sent over several hops as in the case of one Torrent. Insisting on sending them wastes resources and impairs the download time. It is better to artificially decrease the diversification area of seeds in this case. Furthermore, all the nodes will be busy sending or receiving some pieces and the network will be fully used by the different Torrents, so there will be no gain from enforcing parallel network utilization by means of piece diversity. Hence, our new adaptive strategy is aware of the inter-Torrent congestion problem and can easily absorb other transitory congestions or troubles in the path.

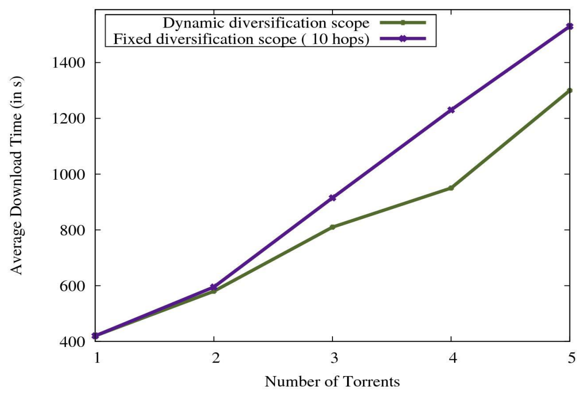


Figure 4.14: Download time Vs. Number of Torrents (Adaption to network congestion-ORBIT experiments)

Figure 4.15 plots the diversification scope of our algorithm in the steady state (after convergence) as a function of the number of Torrents. Clearly, the diversification scope has a decreasing trend with the number of Torrents. In particular, we notice that for the case of one Torrent, our adaptive algorithm converges to a $DS = 10$, which has been shown in Figure 4.10 to be the best value to use. For two Torrents, it is better to shrink slightly the diversification area to have less congestion and routing overhead. If one continues increasing the number of Torrents, the diversification scope will converge to its minimum value 2, which is equivalent to no diversification. As a conclusion, we propose to build the neighborhood using the spanning tree strategy with an adaptive scope of diversification. This strategy provides the best performances in both the single Torrent and the multiple Torrent cases.

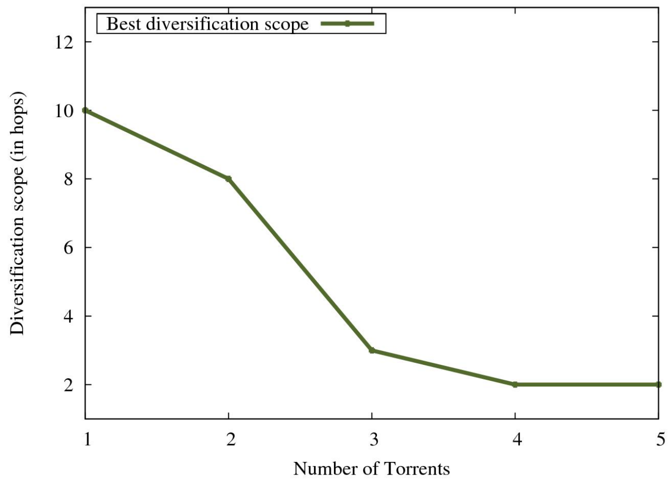


Figure 4.15: Diversification scope Vs. Number of Torrents (Dynamic diversification scope - ORBIT experiments)

4.4.2 Adapting the diversification scope to mobility of nodes

In the previous paragraphs, we supposed that the network is fixed. In case of mobility of nodes, two main factors must be considered. On one hand, the mobility increases naturally the diversity of pieces since the neighborhood of a peer is changing while moving. In this case, one can hope that there will be enough sharing opportunities and hence will be no need for sending pieces to far away nodes to boost diversity. On the other hand, as long paths suffer from high failure rates in mobile ad hoc networks, our adaptive diversification algorithm should converge to the case of no diversification when mobility increases. To validate these claims, we simulate in NS-2 [NS- 2010] a mobile network scenario where nodes move following the well-known Random Waypoint [Hyytia 2006] model. In this model, each node moves along a zigzag line from one waypoint to another. The speed of nodes ranges from 2m/s to 10 m/s. For the pause time, different values were considered; we show here the results for 20s. The dynamic minimum spanning tree is computed by our membership management protocol described in [Sbai 2009b]. The download time and sharing ratio in case of a 100% dense Torrent for different strategies (without diversification, fixed diversification scope and dynamic diversification scope) are summarized in figures 4.16 and 4.17. Clearly and as expected, the version without diversification has better and better performance when the speed of mobility increases. In fact, mobility increases the diversity of pieces in the network. On contrary, a fixed diversification scope strategy obtains bad performances for high mobility speed this is mainly due to instability of long paths and the fact that there is no need to inject extra diversity in the network. Our adaptive algorithm finds the good diversification that suits the current mobility of nodes and then obtains the best performance in all cases in both download time and sharing ratio perspectives.

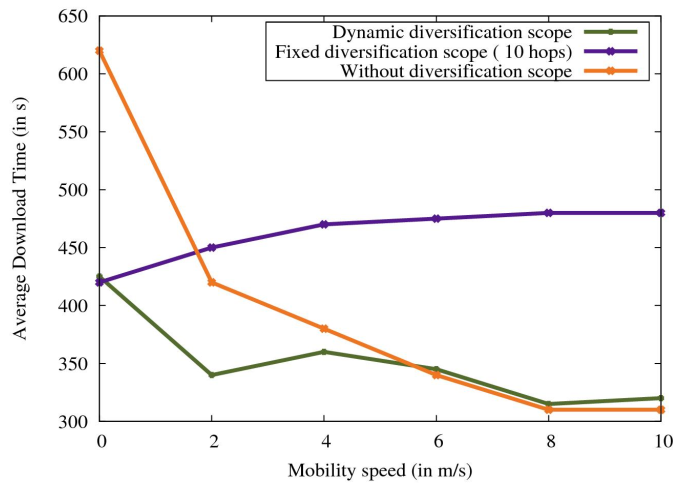


Figure 4.16: Download time Vs. Mobility speed (Adaption to mobility of nodes - NS-2 simulations)

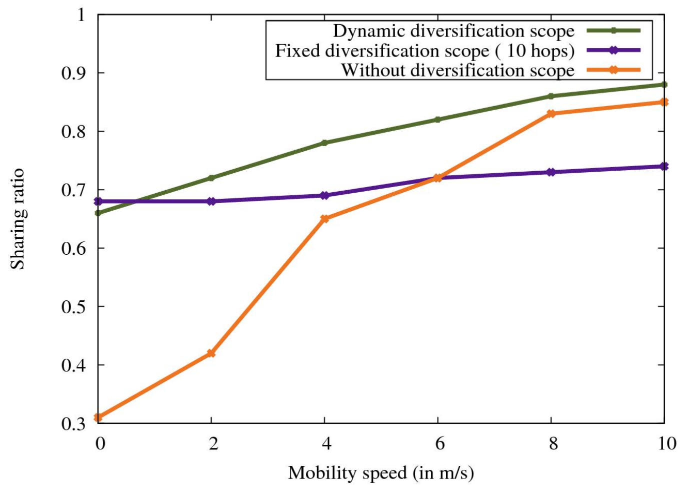


Figure 4.17: Sharing ratio Vs. Mobility speed (Adaption to mobility of nodes - NS-2 simulations)

4.5 Conclusions and perspectives

Spontaneous multi-hop wireless networks offer good opportunities to share content among communities of users. In this chapter, we proposed a general, stand-alone and efficient content sharing protocol for wireless ad hoc networks. The provided solution is a BitTorrent variant that adapts itself to the constrained nature of ad hoc networks and to the underlying topology. Considering the spectrum from sparse to dense Torrents, our first objective was to design a neighbourhood selection mechanism that reduces the routing overhead and hence the resources consumption, while guaranteeing the connectivity of the sharing overlay. Furthermore, we added a diversification mechanism that increases the sharing opportunities among the nodes and improves further the download time. We concluded by considering the realistic scenario of multiple simultaneous Torrents and designing an adaptive diversification algorithm that takes into consideration the network load caused by other Torrents. Extensive simulations and ORBIT experimentations were carried out to support the study and prove the outperformance of our solution compared to a standard BitTorrent-like solution. We mainly gain in download time and record the best sharing ratios. As a future work, one can study if the synergy between Torrents can help ameliorating the performance of content sharing. In fact, considering inter-Torrent incentives could help further increasing sharing opportunities in the case of congested networks.

In Chapter 6, we show how we designed and implemented BitHoc a standalone content sharing application to be used in real networks of mobile devices using their WI-FI ad hoc mode.

Chapter 5

A Membership Management Protocol

Contents

The wide spread of mobile devices (Laptops, PDAs, Smartphones, etc) encourages users to connect directly to each other to build ad hoc communities. These devices, forming spontaneous wireless multi-hop networks thanks to the use of ad hoc routing protocols, can support several services such as content sharing, multimedia streaming, instant messaging, chat conferencing, etc. The infrastructureless nature of mobile ad hoc networks (called MANETs for short) rules out the possibility of deploying services based on dedicated central servers. And even if a node volunteers to play the role of a central server, the global service will not scale and will suffer from bad performances due to interruptions caused by the mobility of nodes, network splits and bandwidth scarcity. Furthermore, MANET nodes are end users having, usually, modest resources. Hence, a single node cannot handle the global load of the service. A decentralized approach like the peer-to-peer one (P2P) is a good candidate solution to be adopted in such environments. Users of a P2P architecture, called peers, organize themselves in a collaborative overlay network by connecting to each other via logical links across the other nodes of the MANET. For example, a P2P content sharing application like BitTorrent [Bit 2010b] distributes the data-transfer load among all the peers interested in the same content. Peers who receive some content pieces are responsible of disseminating them to the rest of the P2P network. Other P2P applications can be Instant messaging, video streaming, social networks, etc.

The problem is that although P2P applications are designed to be completely decentralized, most of them rely on central servers in some of their functionalities. They generally use servers for discovering and updating the information on the members of their overlay. In fact, a P2P application needs to be permanently informed about the set of peers interested in the same service in order to adjust its overlay and account for the arrival and departure of peers. For instance, in BitTorrent [Bit 2010b], each peer is asked to contact periodically a central rendezvous server called Tracker to get up-to-date information about the members of the sharing session. Like this, the peer can choose the other peers with whom to exchange pieces of the content. In general, actual architectures like BitTorrent and instant messaging mainly focus on distributing the data plane but they keep the membership management plane centralized. Thus, they keep the control on the service they provide while fully profiting from the advantages of the P2P semantic in distributing the load of the data plane. The presence of a centralized component in some Internet P2P applications makes it difficult for them to run in MANETs. First, this raises the concern of the single point of failure of the service. In fact, unlike an Internet server, a MANET node has limited resources and cannot handle frequent solicitations. Subsequently, it becomes a bottleneck of the service and overwhelms the underlying network, known for its scarce and shared resources. Moreover, the mobility of nodes, the shadowing, the churn and the network splits can be major factors of interruption of the communications with the central server node. One can imagine the scenario where the network is partitioned into two completely disconnected parts. This means an interruption of the service in the part of the network that does not contain the membership server.

In this chapter, we study the membership management issue in MANET and propose a standalone distributed membership management protocol that emulates the central server and answers the needs of a variety of P2P services when they are deployed in these networks. Our protocol allows P2P applications designed for the Internet to migrate to MANETs without any significant changes in their service overlays. They can use our protocol to construct and share common knowledge about their overlay members equivalent to what is provided by central servers in the Internet.

Our protocol overcomes the limitations of the central server solution and takes into account the constraints of a MANET. It provides a distributed solution by relying on a collaborative approach: peers organize themselves in a shared tree dedicated for disseminating membership information. Events like new arrivals or departures of peers are announced on the tree so that each peer can keep permanently an up-to-date list of the members of the P2P service and of any related information. Using ad hoc routing information, peers construct and adapt their logical links in the membership tree according to the current topology of the network. Their goal is to minimize the length of the tree (in terms of number of wireless hops) so as to reduce the membership traffic and the overhead on the underlying network. The membership tree can be seen as a distributed minimum spanning tree connecting peers of the service. We propose fully decentralized mechanisms that allow peers to adapt the membership tree structure to the frequent changes of the underlying network caused by nodes' mobility. Moreover, our protocol addresses the partitioning issue in MANETs. We achieve this by the help of a new and simple mechanism that allows peers to benefit from the information provided by the underlying routing protocol for discovering peers from separate overlays providing the same service. This makes two or more separate trees for membership management belonging to the same service (e.g. same content in BitTorrent) merge together to form a new covering tree.

In the literature, many works have been conducted to implement P2P applications in MANETs. We refer to the background and motivations chapter for a brief description of these implementations. The majority of them does not study independently the membership management issue but rather focus on the data plane. In fact, the cost of the membership management is often ignored compared to the cost of the data traffic. More importantly, these works do not provide a solution for the network splits in MANETs that are caused by nodes' mobility and the finite range of the wireless. If not handled correctly, these splits may lead to an interruption of the service and an inefficient use of resources.

To validate the benefits of our protocol compared to classical solutions, we added a module to the NS-2 network simulator [NS- 2010] and conducted extensive simulations. The performance of a membership management solution can be measured in terms of the volume of signaling traffic it generates and the level of freshness of the knowledge about the members of the P2P service it allows. As there is a tradeoff between increasing the freshness of membership information and diminishing the cost of the management, we define appropriate metrics to measure the efficiency of the compared solutions in both regards. The comparison of our protocol to client/server, flooding and multicast-based solutions shows that it achieves indeed lower network overhead while ensuring better membership information freshness.

The work presented in this chapter has been published in the proceedings of the International Conference on Mobility Technology, Applications and Systems (Mobility 2009) [Sbai 2009b]. This chapter is an extended version of this paper and is organized as follows. Section 5.1 overviews the related work. Section 5.2 gives a short description of the graph concepts behind our membership management protocol. Section 5.3 explains the design of our protocol and includes a detailed presentation of its algorithms. Section 5.4 evaluates the performance of our protocol compared to other solutions. Section 5.5 summarizes the chapter and gives some idea on future work in the membership management direction.

5.1 Related work in membership management

In this section, we overview the main existing solutions that are relevant to our membership management problem. First, we describe the efforts done to manage the membership of P2P systems in the Internet. Then, we study P2P multicast overlays in MANET for the purpose of underlining the similarities and the differences that exist between membership management and multicast.

5.1.1 Membership Management in the Internet

Many membership management techniques have been proposed for the Internet. They can be subdivided into two categories: those decoupling the P2P data plane from the membership management and those coupling them together. One can mention here the client-server architecture used by BitTorrent to track peers as a solution that decouples the two functionalities. In BitTorrent, each peer contacts periodically a central rendezvous server named Tracker in order to update its list of peers. In parallel and to distribute the server functionalities, mechanisms based on Distributed Hash Tables (DHTs) have been also introduced to supply P2P applications with membership information without relying on one single server. For example, P2PSIP [P2P 2010] organizes nodes into a structured DHT-based overlay where ordinary peers having abundant capacities can become servers. Ordinary peers locate servers by DHT lookup functions. DHT-based solutions are efficient in the Internet since the graph of communication is totally meshed and the bandwidth is abundant. In a MANET, these properties do not hold. The network may split into separate clusters and nodes serving as DHT servers remain the bottleneck.

Other P2P protocols do not consider quality of service criteria when constructing their overlays and so they use the same structure to do both peer discovery and data dissemination. Content-based routing P2P networks [Gnu 2010] are examples of these techniques. In general, when quality of service is a concern, it is better to decouple the membership management from the data plane to allow for more efficient overlay construction.

Some other works (e.g. [Boyd 2005]) address the scalability problem by deploying gossiping techniques. The solution proposed is to contact a random sub-set of peers and to exchange with them known information on other peers. This technique generates random graphs over which peers exchange their knowledge about the service overlay. It is an acceptable option when the knowledge of a sub-set of peers is sufficient for a good P2P service and when the communication between faraway nodes is not constrained by physical connectivity. Otherwise, the overhead on the underlying network will be very important and the P2P application will suffer from bad performances.

5.1.2 P2P Multicast overlays in MANETs

The problem of constructing a P2P multicast protocol for MANET has some common challenges with our problem. Indeed, multicast protocols require a membership management component to track the MANET nodes interested in the session. The problem is that multicast protocols often aim at optimizing the data transfer plane and neglect the signaling plane. The energy spent on signaling is compensated by the efficiency of the data plane itself. In our case, we only focus on the dissemination of the membership information itself, which could be seen as only having the control packets of a P2P multicast overlay without the data. Moreover, some existing multicast protocols are centralized or require global knowledge which we want to avoid [Gui 2003]. We add to that to the fact that there is no multicast-based solution for the problem of network splitting.

To optimize the data transfer plane, multicast protocols proposed for MANET were constructed following two approaches: protocols based on meshed overlays and protocols based on tree overlays. Meshed overlays are non-structured; they represent random graphs linking nodes of the network. This kind of overlays offers more connectivity and more robustness by maintaining redundant paths between nodes. Nevertheless, the meshed topology is not efficient in MANET due to the overhead caused by duplicated transmissions of packets on redundant paths. Unlike meshed overlays, the tree topologies are very efficient in the MANET environment as they result in low load on the network by avoiding path redundancies. But they are less robust and require specific mechanisms to adapt to the frequent changes in the physical network. Our protocol adapts a minimal-cost tree structure while making it adaptive and resilient to network splits. Here are two examples of overlay multicast protocols recently proposed for MANETs:

- **PAST-DM** stands for Progressively Adapted Sub-Tree in Dynamic Mesh [Gui 2003]. Peers in PAST-DM first organize themselves in a mesh network and then each of them, knowing the topology of this mesh, computes in a centralized way a minimum spanning tree. Each peer discovers its neighbors in the meshed graph by broadcasting messages in a limited scope. This discovery is done periodically in order to adapt the mesh to the underlying topology. Neighbors in the mesh are linked through unicast tunnels in order to exchange link-state information allowing the computation of the spanning tree. When a peer leaves the overlay, the information on its departure propagates via the unicast tunnels until it reaches all the members of the overlay. The periodic exchange of link-state information is costly and the computation method is suboptimal since it must be done periodically by each node.

- **MOST** (Multicast Overlay Spanning Tree Protocol) [Rodolakis 2007] is an overlay multicast protocol based on the construction of a minimum spanning tree. This protocol requires imperatively the use of the OLSR routing protocol. In fact, each peer uses the topology information provided by OLSR in order to compute an optimal spanning tree. This tree is recomputed periodically to adapt to the changes in the topology and in the members of the overlay. Peers flood periodically JOIN messages including the addresses of the multicast groups they belong to. Each peer maintains a peer list per multicast group. If it does not receive any JOIN message from one of the peers during a specific period of time, it deletes it from the lists of peers of its multicast groups. Here also the cost of flooding JOIN messages periodically is very important and the solution requires the use of OLSR as the underlying routing protocol.

5.2 Graph theory background

In this section, we present some graph theory concepts that we use in the design of our membership management protocol. Let $G(V, E)$ be a graph. V and E are respectively the set of vertices and the set of edges of the graph.

- A **cycle** is a subset of edges that forms a path such that the first node of the path corresponds to the last one.
- A **cut** is a partition of the vertices of the graph into two disjoint sets S and T. Any edge $e(u, v)$ in E with u in S and v in T is a cut edge.
- A **tree** is a graph in which any two vertices are connected by exactly one path. Given a connected undirected graph, a **spanning tree** of this graph is a subgraph which is a tree and which connects all the vertices together. A single graph can have many different spanning trees.
- One can assign a weight to each edge of a graph. The cost of a spanning tree can be computed as the sum of the weights of the edges of that spanning tree. A **minimum spanning tree** is then a spanning tree whose weight is less or equal than the weight of every other spanning tree. More generally, any undirected graph (not necessarily connected) has a minimum spanning forest, which is a union of minimum spanning trees for its connected components.

The following properties of a minimum spanning tree have been profitable for the development of our protocol:

- **Cycle property:** For any cycle C in the graph, if the weight of an edge e of C is larger than the weights of other edges of C, then this edge e cannot belong to a minimum spanning tree.
- **Cut property:** For any cut C in the graph, if the weight of an edge e of C is smaller than the weights of other edges of C, then this edge e belongs to all minimum spanning trees of the graph.

5.3 The membership management protocol

In this section, we describe our protocol for P2P membership management in MANET. Our protocol constructs a spanning tree to be used for the exchange of membership information among peers in the P2P network. We want this tree to match the topology of the underlying network in order to minimize the cost of the dissemination of membership information among peers and to ensure the freshness of the lists of members maintained by each peer. As optimality is needed, we propose to construct a minimum spanning tree in terms of number of hops, covering all peers of the underlying routing graph (some MANET nodes might not be P2P members). This guarantees a minimum cost of the membership information dissemination in terms of number of hops, transmissions and power. We introduce efficient and distributed mechanisms to track the intermittent connections and disconnections of the MANET nodes. Moreover, because nodes are continuously moving and so tree weights are subject to frequent changes, our protocol needs to restructure the tree when needed to maintain its optimality property. Centralized algorithms, as the well known Kruskal algorithm [Kru 2010], are to be discarded because on one side they require global knowledge and on the other side each peer computes its own spanning tree. Since the minimum spanning tree is not unique, peers might then calculate different spanning trees. Unfortunately, this may disconnect the service overlay. Our protocol is based on a completely distributed approach which guarantees the uniqueness of the tree by making all decisions locally. Optimality is ensured by satisfying the cycle and cut proprieties described in Section 5.2. Another important factor that we consider in the construction of our adaptive tree is network partitioning. We add to our protocol a specific technique to merge separate trees belonging to the same P2P network when communication opportunities occur. The following paragraphs describe our protocol and explain its core algorithms.

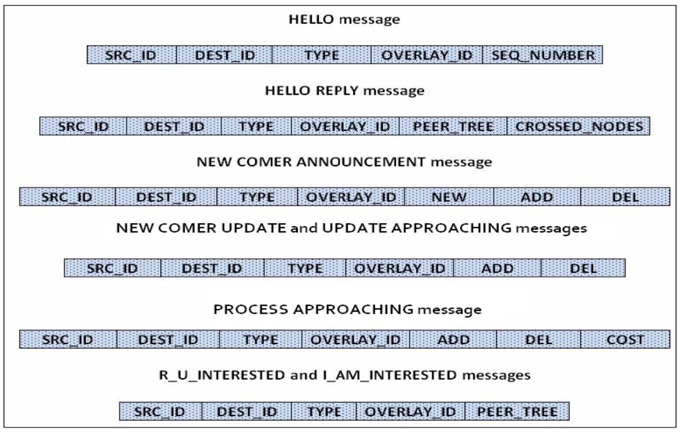

HELLO message
SRC_ID | DEST_ID | TYPE | OVERLAY_ID | SEQ_NUMBER
HELLO REPLY message
SRC_ID | DEST_ID | TYPE | OVERLAY_ID | PEER_TREE | CROSSED_NODES
NEW COMER ANNOUNCEMENT message
SRC_ID | DEST_ID | TYPE | OVERLAY_ID | NEW | ADD | DEL
NEW COMER UPDATE and UPDATE APPROACHING messages
SRC_ID | DEST_ID | TYPE | OVERLAY_ID | ADD | DEL
PROCESS APPROACHING message
SRC_ID | DEST_ID | TYPE | OVERLAY_ID | ADD | DEL | COST
R_U_INTERESTED and I_AM_INTERESTED messages
SRC_ID | DEST_ID | TYPE | OVERLAY_ID | PEER_TREE

Figure 5.1: Packets' format

5.3.1 Joining the membership tree

We suppose that each P2P service has a unique identifier (e.g. file ID in the case of BitTorrent). Knowing this identifier, a node that becomes interested in the service initiates a join procedure. This procedure can be divided into two phases: discovering the nearest peer and disseminating the new arrival information to all other

Table 5.1: Description of packet fields

Field name	Field description
SRC_ID	Identifier of the peer sending the message
DEST_ID	Identifier of the peer that will handle the message
TYPE	Type of the message (e.g. HELLO)
OVERLAY_ID	Identifier of the service overlay
SEQ_NUMBER	Sequence number of the message
PEER_TREE	A string representing the tree
CROSSED_NODES	Nodes tagged for partitioning awareness
NEW	Identifier of the node joining the overlay
ADD	List of logical links to be added to the tree
DEL	List of logical links to be removed from the tree
COST	Weight of the most costly logical link in a cycle

peers. If needed, this triggers the update of the membership tree to maintain its optimality.

5.3.1.1 Discovering the nearest peer

In order to discover a first attachment point to the membership tree, we propose to use a simple flooding technique with a controlled scope. The new member floods a HELLO message in its one hop neighborhood ($TTL = 1$) and then waits for HELLO_REPLY messages, sent in unicast, from any member of the P2P network located at one hop. In case there is no answer, the new member increments exponentially the value of the TTL of the HELLO message and waits again for at least one HELLO_REPLY. If the maximum TTL is reached and no answer is received, the node considers that the service is not provided in the network and that it is up to it to build a new membership tree. If an answer is received, the new member gets in the HELLO_REPLY message a copy of the current tree with the list of members and other useful information as the canonical names and descriptions. We underline that in our method a peer does not need to know the cost of the edges of the tree, it only needs to know which peer is connected to which other peer. The format of HELLO and HELLO_REPLY is depicted in Table 5.3 and Figure 5.1.

Using its current routing table, the new arriving peer compares the costs in number of hops to connect to other peers in the tree. This comparison allows it to identify the closest peer to it. The new peer should then connect to the spanning tree by attaching itself to this closest peer as required by the cut property described in Section 5.2. This property requires that the connection to add is the one having the lowest cost in the cut formed on one side by the old tree and on the other side by the new peer. In practice, after identifying this closest peer, the new arriving peer sends in unicast a simple CONNECT_ME message to it. Receiving this message, the nearest peer triggers a new arrival information dissemination phase on the old tree. This phase is coupled with an adaptation of the tree, to be described next.

5.3.1.2 New arrival information dissemination and tree adaptation

When a member of the tree receives a CONNECT_ME message from a new joining peer, it adds this peer as a child node. This modification of the tree is then disseminated to the other peers to trigger any necessary modification that keeps the tree optimal. The new parent sends to all its neighbors in the tree, except the new arriving peer, a NEW_COMER_ANNOUNCEMENT message containing the identifier of the new peer and the modifications it has made on the tree. We refer to Figure 5.1 and Table 5.3 for details on this message. Every peer that receives the NEW_COMER_ANNOUNCEMENT message updates its knowledge about the tree and verifies whether it can modify some of its logical links to improve its connectivity to the tree next to this new arrival. This modification is described later in this section. After making these local decisions, it informs its neighbors in the tree, except the peer which has sent the NEW_COMER_ANNOUNCEMENT message. It does that by sending a new version of the NEW_COMER_ANNOUNCEMENT message adding, eventually, its own changes. Upstream peers that have already seen the NEW_COMER_ANNOUNCEMENT message are informed by the modifications through a NEW_COMER_UPDATE message, which contains the logical links that the peer has added or removed from the tree. The NEW_COMER_UPDATE message differs from the NEW_COMER_ANNOUNCEMENT message by the fact that it does not trigger any modification of the tree. A peer receiving this former message just updates its knowledge about the tree. In this way, all peers are aware of the new arrival and the tree is restructured in parallel. The decision that a peer must make when it receives a NEW_COMER_ANNOUNCEMENT message is based on a simple verification of the cycle property described in Section 5.2. The cycle to consider is the one formed by the logical links on the path of the current tree starting from the intermediate peer making the decision to the newly joining peer, and by adding the direct logical link between both peers. If any optimization is possible, it will result in cutting the logical link through which the peer received the NEW_COMER_ANNOUNCEMENT message and adding the logical link to the newly joining peer. This way the cost of the tree is always kept minimal. One can notice that all the decisions are made locally and in a distributed manner without compromising global optimality.

5.3.2 Adapting the membership tree to mobility of nodes

Due to the mobility of MANET nodes, the distances between the peers of the membership tree vary in time. If the spanning tree is not adapted to these movements, it will quickly lose its optimality property. One can distinguish four possible movements of peers:

- Two peers that are neighbors in the tree can get closer. In this case, the cost of the spanning tree becomes smaller but it remains a minimum spanning tree. This movement has no impact on the structure of the tree.

- Two peers that are not neighbors in the spanning tree get farther from each other. In this case, the weight of the tree does not change and there is no decision to be taken.
- Two peers that are neighbors in the spanning tree get farther from each other. The weight of the current tree increases which means there might exist a better tree to be identified.
- Two peers that are not neighbors in the spanning tree get closer to each other. In this case, the weight of the current spanning tree does not change but there might be another spanning tree with a smaller weight. This movement requires, eventually, an adaptation of the spanning tree seeking for the existence of an optimal one.

In the following paragraphs, we describe how we adapt the membership tree in response to the two latter movements impacting its optimality.

5.3.2.1 Two neighbors in the tree get farther from each other

If two peers that are neighbors in the spanning tree get farther due to mobility, this leads to two possible situations. The first situation is that one of these peers or maybe both will get closer to other peers of the tree. Here, the tree adaptation can be done by applying the approaching adaptation procedure which we describe in paragraph 5.3.2.2. The second situation is that no one of these two peers gets nearer to other peers, in this case no better spanning tree can be found and no adaptation of the tree is needed. Hence, the problem raised by neighbors getting farther from each other can be always transformed into a simple approaching problem and solved by the solution we come up for the latter one.

5.3.2.2 Two peers that are not neighbors and that get closer to each other

Let $P1$ and $P2$ be two peers that are not neighbors in the spanning tree. Suppose that the cost of the physical direct path between these two peers becomes smaller due to the mobility of nodes. Take the cycle formed by the tree path from $P1$ to $P2$, and by adding the network link that connects directly $P1$ to $P2$. If there is a logical link L in this cycle such that $cost(L) > cost(P1, P2)$, the cycle property described in Section 5.2 indicates that the logical link $(P1, P2)$ must belong to the minimum spanning tree. So the actual tree should be adapted in such a way to replace the logical link L by the logical link $(P1, P2)$. Each peer in the tree tracks continuously other peers and verifies if another peer, which is not its neighbor in the current tree, becomes closer than its farthest logical neighbor. Here, it forms the cycle between it and this peer and initiates a procedure of identification of the most costly link in the cycle. Between the two peers, the one having the lowest identifier initiates the procedure. This is done by circulating a PROCESS_APPROACHING message on all logical links of the cycle. Each peer in the cycle adds its logical link to the next peer in the cycle as being the link to be removed (DEL field in the message) if this link is more costly than the link it finds in the message. It also updates the field COST in the message because peers only know the costs of their own logical links. This procedure is repeated until the message returns to its original sender which then can decide which link to be removed and which link to be added. This modification is then disseminated to all peers. Figure 5.2 plots a membership management tree that adapts to two peers getting closer due to mobility. In fact, the path between peer 6 and 7 is now cheaper (2 hops) than the path between the peer 1 and 7 (3 hops). Our mechanism detects automatically that the link $(1, 3)$ has the highest cost in the cycle $(1, 3, 6, 7, 1)$ and then it is the link to be cut. To ensure the connectivity and the optimality, the link $(6, 7)$ belongs now to the membership tree. The peer sends an UPDATE_APPROACHING message to its neighbors in the tree which forward it to their neighbors and so on. The messages are described in Figure 5.1 and Table 5.3.

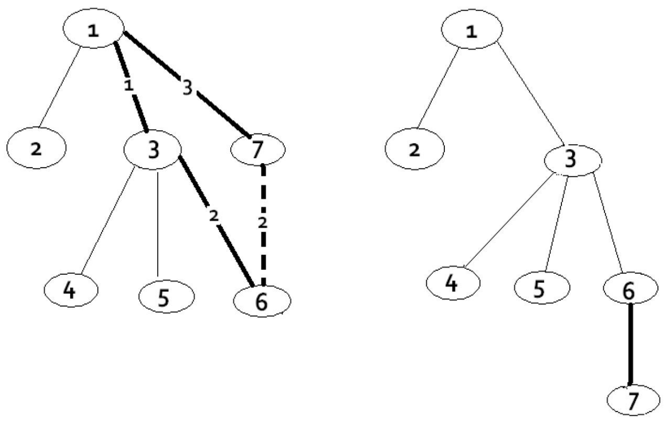


Figure 5.2: Adapting the tree after node 6 and 7 get closer due to mobility

5.3.3 Leaving the membership tree

Adapting the membership tree after the departure of peers is very important for the efficiency and the uniqueness of the tree and for the freshness of the membership information. We ask every peer that decides to leave the P2P service to inform its logical neighbors in the spanning tree by sending an I_AM_LEAVING message to them. The format of the I_AM_LEAVING message is described in Figure 5.1 and Table 5.3. Excepting for leafs, the departure of peer will result in the decomposition of the tree into two separated sub-trees. The first sub-tree represents the children of the leaving peer and the second one its parents. To reconnect the tree, the child of the leaving peer having the highest identifier connects to its parent and becomes the parent for the remaining children. This way, a new spanning tree is formed. The problem is that this new tree may not be optimal. The optimal tree is reached by having the peers apply the normal approaching adaptation procedure described earlier in paragraph 5.3.2.2. Note how this procedure is important for our solution to always rewire the tree in a way to ensure its optimality. All modifications made on the old tree are disseminated to all peers of the tree together with the identifier of the departing peer. Sometimes a peer can leave the service overlay improperly; in this case, it is the duty of the first neighbor detecting this to trigger the tree adaption procedure. We plot in Figure 5.3 a membership tree before and after the departure of a peer (peer 3 here).

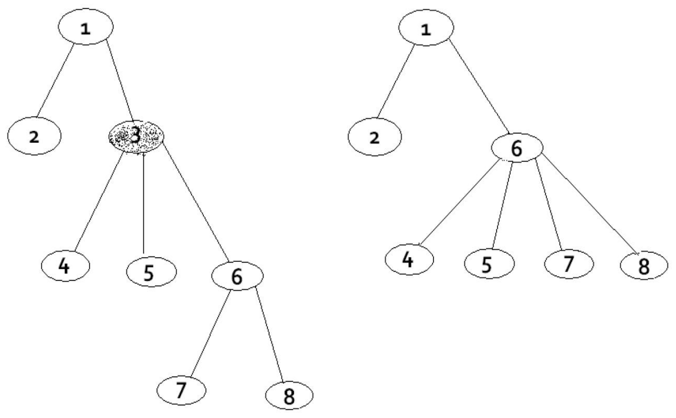


Figure 5.3: The membership tree before and after the departure of peer 3

5.3.4 Network split awareness

Due to the high dynamicity and mobility of MANETs, the network can split into different disconnected clusters. These clusters can merge again into one or more larger clusters. So one can imagine the scenario where one P2P network is split into two or more networks because of the underlying network splitting. Another scenario is that two or more membership trees constructed separately in different clusters but offering the same P2P service meet together. At the first merging opportunity, it is very important to connect together the different partitions of a membership tree in one large and efficient tree. That is why we add to our membership protocol a mechanism allowing peers to discover others coming from other partitions when they get close to them.

After connecting to the current minimum spanning tree, each peer observes its routing table and tags network nodes that are not interested in the same service. Then, using its routing table, it tracks continuously the appearance of non tagged nodes in its neighborhood. We choose the neighborhood of a node in the P2P network to be equal to the maximum number of hops to one of its direct neighbors on the spanning tree. A new node not tagged and not belonging to the same membership tree is a good candidate to be asked whether it belongs to the same service and comes from another cluster. The peer sends to this newly detected node a message R_U_INTERESTED in order to ask whether it is interested in the service. If it receives no answer from that node then it tags it as a not interested node. The tagging information is disseminated to all the peers in order to reduce the number of R_U_INTERESTED messages. When a node receives an R_U_INTERESTED message and if it is interested in the service, it answers the source by sending an I_M_INTERESTED message. In this case, the two trees maintained by the two peers need to be merged together. To ensure efficient merging, the peers of the smallest tree apply the join procedure in order to connect to the biggest tree. The formats of the messages R_U_INTERESTED and I_M_INTERESTED are described in Table 5.3 and Figure 5.1.

5.3.5 Global knowledge is not required

Until this point, we have considered that each peer must have a global knowledge about the members of the service overlay. In some cases for scalability reasons, this is a very strong assumption since data exchanges with very far nodes suffer from very bad performances. Hence, each node in this case can restrict its knowledge about the overlay to some scope. For example, for our BitHoc service described in Chapter 4, one can limit the scope of the knowledge of a peer to a number of hops slightly higher than the scope of diversification.

In this paragraph, we show that when applying the mechanisms described above while considering limited scopes of knowledge about the members of the P2P service, one still construct global minimum spanning tree and the overlay partitioning problem is still handled. First, when a new peer wants to join the overlay. It will be informed by its nearest peers about their local knowledge of the membership overlay and then it will construct its own local knowledge of the overlay. It will after connect to the nearest peer among those in its local knowledge. This is equivalent to connecting to the nearest peer in the global overlay. In the case of two peers non belonging to the global tree that get nearer, these two nodes must be in the local knowledge about the overlay of each other otherwise this rapprochement is not important to be considered. The case of separate overlays does not change since the tagging-querying mechanism we deploy to merge the same overlays together still work in the case of local knowledge.

This limited knowledge about the overlay will significantly reduce the load on the network since information on arrivals and departures will propagate only in the local scope interested by these changes.

5.4 Performance evaluation

In this section, we study the performance of our protocol by comparing it to classical membership management approaches. The validation is based on extensive simulations run with our implementation of the different membership management methods in the NS-2 network simulator [NS- 2010]. First, we describe the metrics used in the evaluation of the different approaches. Then, we describe the simulations scenarios and the membership management techniques implemented for comparison.

Finally, we discuss the compared results and conclude on the performance of our protocol.

5.4.1 Performance metrics

As we described earlier, a good membership management protocol must minimize the overhead of the peer discovery and update operations on the underlying network. Thus, it should limit the number of packets sent for this purpose. Furthermore, the information hold by peers in this protocol must be consistent. Hence, at any moment, it should reflect the real members of the P2P service. Consequently, there is a tradeoff between minimizing the cost of the management and having a good adaption to the changes in the members of the overlay. The performance metrics that we define below take into consideration these aspects.

Let P_t be the set of peers interested in the service at instant t and let p_t be the cardinality of this set of peers. When a peer is running a membership management approach, it maintains at the instant t a set of peers N_t that corresponds to its view of the members of the P2P service. Let n_t be the cardinality of this set. Due to peers that leave the P2P network or generally errors in the membership management, among these n_t peers, there are t_t peers belonging to P_t and f_t peers not belonging to it.

During a specific measurement time (namely the simulation time for us here), peers exchange messages between them in order to discover the interested peers and update their knowledge about them. Let C be the cost in number of hops over paths crossed by the exchanged messages during a fixed period of time. The importance of this cost C varies with the method used for membership management. However, this cost does not consider the freshness of the information maintained by the peers. Thus, it is not enough to decide whether a method is appropriate or not. In fact, one can spend a very low cost and have a lot of pollution in its knowledge about the peers. That is why we propose another cost metric C_f that accounts for the freshness of information. This cost is also a global metric computed during a fixed time. After each small period T_s measured in seconds, one takes a snapshot of the P2P network and measures the value of p_t and computes $\hat{n_t}$, $\hat{t_t}$ and $\hat{f_t}$, the average values of n_t, t_t and f_t over peers interested in the P2P application. At the end of the simulation time, one computes $\tilde{p}$, $\tilde{n}$, $\tilde{t}$ and $\tilde{f}$ the average values of p_t, $\hat{n_t}$, $\hat{t_t}$ and $\hat{f_t}$ over all measured samples. The cost corrected by freshness of information C_f is the ordinary cost to which we add two terms. The first term accounts for the cost that the P2P network should pay to discover the $\tilde{p}-\tilde{t}$ missing members. This term can be easily calculated considering that the members of the P2P application have paid on average $\frac{C}{\tilde{n}}$ to discover one peer. Hence, the term modeling the missing information cost will be equal to $\frac{C(\tilde{p}-\tilde{t})}{\tilde{n}}$. The second term to be added to the ordinary cost is a term accounting for the pollution existing in the knowledge of the peers. We consider that one pays the same cost to discover an interested peer or to remove an idle one. That is why we take this term equal to $\frac{C\tilde{f}}{\tilde{n}}$. The following formula computes the cost metric corrected by the freshness of information:

$$C_f = C(1 + \frac{\tilde{p} - \tilde{t}}{\tilde{n}} + \frac{\tilde{f}}{\tilde{n}}) \tag{5.1}$$

5.4.2 Simulations scenario

To ensure the validation of our membership management protocol against other possible methodologies, we implement all approaches in the NS-2 [NS- 2010] network simulator. The membership management module communicates with the existing NS-2 modules that implement MANET routing, MAC and physical layers, etc.

To conduct our simulations, we consider a MANET composed of 50 nodes moving inside a bounded area of width 100m and length 500m following the Random Waypoint mobility model [Hyytia 2006]. The speed and the pause time of each node are taken equal to 2m/s and 30s respectively. The nodes connect to each other using the IEEE 802.11 MAC layer [802 2010] with the RTS/CTS-Data/ACK mechanism enabled. The range of transmission is fixed to 50m and the data rate is set to 1 Mb/s. For ad hoc routing, we use the proactive OLSR protocol [Jacquet 2001]. The mobility model corresponds to a random pedestrian mobility, simulating then a community of people moving around in an open-space area and wanting to share an experience using a P2P service.

To simulate a dynamic membership, we suppose that a node has two states: the first state is an idle state where it is not interested in the P2P service; the second state is the one where it becomes interested in the P2P service. The membership of a node follows then an ON/OFF process until the end of the simulation (see Figure 5.4). The durations of the ON and OFF states follow an exponential distribution of parameter λ and μ respectively.

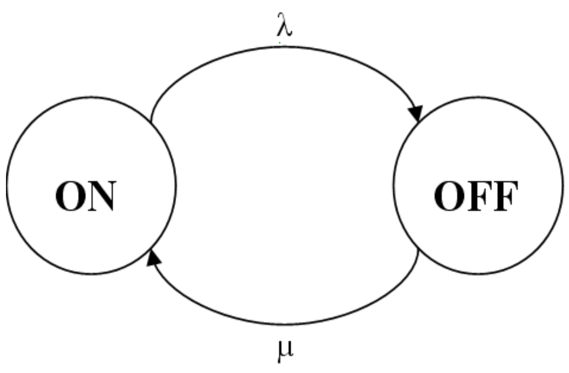


Figure 5.4: ON/OFF process

We define the density d of the P2P overlay as the number of nodes interested in the P2P network divided by the total number of MANET nodes. One can easily show that this density is on average equal to:

$$d = \frac{1}{1 + \frac{\lambda}{\mu}} \tag{5.2}$$

When not stated the density is taken equal to 50 % by assuming that both λ and μ are equal to 500s. The simulation duration is set to 3600s. The sampling period T_s used to compute the corrected cost is chosen equal to 10s.

With this scenario, we can test the adaption capacity of different protocols to different dynamic aspects of the service overlay and the underlying network topology. In fact, peers are constantly interested and disinterested in the P2P service following the ON/OFF process. In this way, one can test the mechanisms designed to take into consideration the arrivals and the departures of peers and validate how the different approaches for membership management update their knowledge about the members of the P2P service. Furthermore, having a mobile network, we can evaluate whether our mechanism yields the minimum cost by adapting its membership tree to topological changes. Moreover, we can study the impact of mobility on the stability of the other approaches and measure the cost they engender in such a dynamic environment.

5.4.3 Approaches used for comparison

We compare, through extensive simulations, our protocol to four classical methods for membership management: a client/server method, a flooding-based method, a multicast-based method and a non-adaptive tree method.

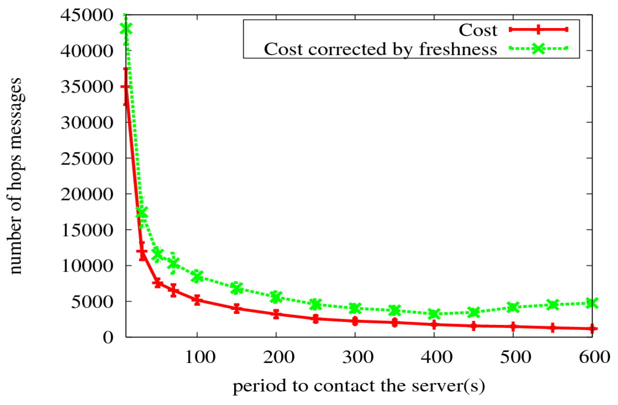


Figure 5.5: Real and corrected cost Vs. Period

- **Client/server method:** The classical client/server method supposes that peers contact periodically a central server to update their knowledge about the members of the P2P application. In our simulations, a random selected node plays the role of the server. Figure 5.5 plots the real cost in number of hops-messages as a function of the period at which the peers contact the server. It shows that this cost is proportional to the inverse of the contact period. The more the period of contacting the server is large, the less peers send messages for the membership management. The same figure also plots the cost corrected by the freshness of the membership information. One can notice that this cost is higher than the ordinary cost. Contrary to the real cost, this former does not continuously diminish when the period of contacting the server increases. In fact, the freshness of information decreases with the increase in the contact period, which is accounted for in our corrected cost metric. In the following simulations, the contact period is set to 400s which according to the figure yields the best performances for the client/server method.

- **Flooding-based method:** Peers advertise their arrivals and their departures to other interested peers by physically flooding the network. Two types of sequenced messages are used for this purpose: peer-joining and peer-leaving messages. When receiving such a message, each node in the MANET forwards it to its physical neighbors if it is seen for the first time, otherwise it is discarded. For this purpose, every message has a unique sequence number allowing to timestamp it. During the broadcast, if a MANET node is interested in the P2P service, it updates its membership information.

- **Multicast-based method (PAST-DM):** Multicast protocols developed for MANET share with P2P services the objective of updating their knowledge on the members of the overlay when the nodes join and leave the multicast group. To compare with the membership management of multicast protocols, we use the PAST-DM protocol [Gui 2003] known for its efficiency in MANET. We refer to the related work section of this chapter for a detailed description of this protocol and of its membership management mechanism. In our simulations, we implement for PAST-DM the exchange of link state messages, the JOIN/LEAVE messages and the messages to discover peers in the near neighborhood. The period to exchange the link state messages is set to 30s in order to match the value of the pause time for nodes' mobility.

- **Non-adaptive tree method:** This method is very similar to our membership management protocol. It implements the same algorithms but it does not adapt the constructed spanning tree to the topology and dynamicity of the network. In fact, a peer joining the service does not connect itself to the nearest peer but to the first responding peer and the constructed tree is not adapted to the topological changes of the underlying network. Hence, the constructed tree is a sub-optimal spanning tree. Our aim is to prove the need for topology awareness and the gain obtained from it.

5.4.4 Comparative study

In this paragraph, we present simulation results for the different membership management approaches. The objective is to compare them considering the different metrics that we have defined earlier in this section.

5.4.4.1 Real cost of the membership management

We begin our comparison by analyzing the impact of the overlay density on the real cost of the membership management. This cost measures the number of hops crossed by messages sent for the purpose of discovering peers and updating the membership information. The results for the four methods are presented in Figure 5.6. They show that the cost of the flooding-based method increases linearly with the number of interested nodes in the network and is quite high. This is due to the fact that information about arrivals and departures of peers is flooded in the entire network, creating a large number of redundant messages. In contrast, the cost of our protocol increases slowly with the overlay density while staying low. One can explain this behavior by the fact that the expanded-range technique used by our protocol for discovering peers guarantees a low cost in dense overlays. Moreover, update messages circulate along shortest paths of the minimum spanning tree without generating redundant messages.

Although PAST-DM implements a controlled flooding technique to overcome the limitations of classical flooding technique, its periodic updates increase dramatically its membership management cost as the overlay density increases. Finally, unlike our protocol, the non-adaptive tree method does not scale when the P2P network grows because of the sub-optimality of the costs of the tree branches. The client/server method yields a real cost slightly higher than our protocol for the 400 s contact period, which has been shown to be the optimal in our earlier simulations. However, one cannot decide on the best approach considering only the real cost because it does not reflect the consistency of the membership information.

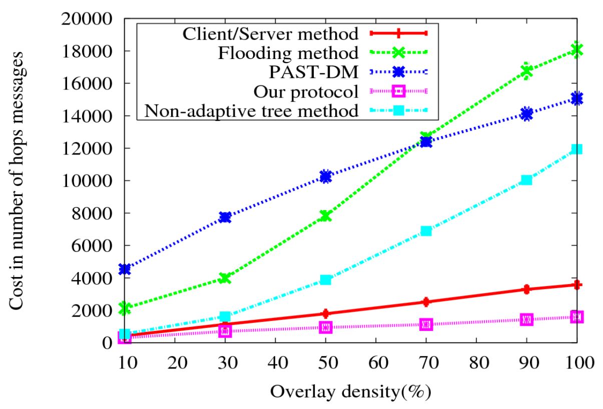


Figure 5.6: Real cost Vs. Overlay density

5.4.4.2 Freshness of the membership information

As a second step, we examine the freshness of membership information of the different methods. We plot in Figure 5.7 the ratio of the corrected cost to the ordinary cost as a function of the overlay density. As the cost corrected by the freshness of information takes into consideration the consistency of the knowledge about the overlay, this ratio measures the freshness of the information hold by peers. The higher this ratio is, the more out of date the peers are. A ratio equals to 1 means that peers have correct knowledge of the P2P service members over time. One can notice that our protocol and the non-adaptive tree method achieve a ratio value very close to 1. In fact, in these two methods, triggered updates and the tagging technique allow immediate information dissemination among peers. As expected, the client/server mechanism achieves a quite high ratio, even in sparse overlays. This confirms the idea that the client/server method does not scale in wireless environments. Concerning PAST-DM, update information is gradually propagated in the network through iterative exchanges between peers. Hence in dense overlays, where neighbors are physically close to each other, information needs several periods to reach all the peers. This explains the high cost ratio and the increasing trend seen in Figure 5.6. The flooding method congests seriously the network and requires a lot of transmissions to update the membership information. This congestion yields important packet losses resulting in a bad freshness of the membership information.

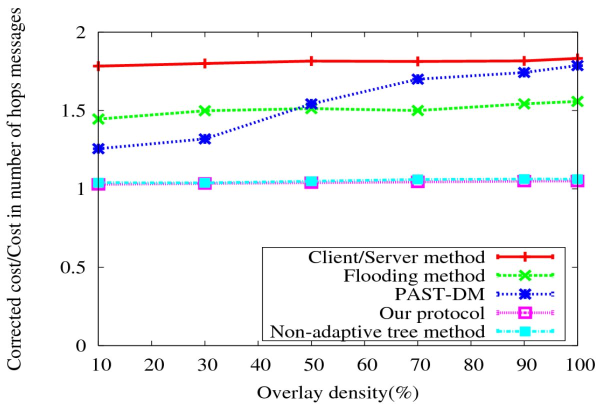


Figure 5.7: Ratio of corrected cost to cost Vs. Overlay density

5.4.4.3 Cost corrected by the freshness of information

By comparing the different methods using directly the cost corrected by the freshness of the membership information, one can decide which one is better than the others in terms of both network overhead and freshness of information. This metric adds a virtual cost that corresponds to the extra-effort that a protocol must provide to have the best freshness of information. In Figure 5.8, we plot this metric as a function of the overlay density. The figure shows that our protocol outperforms the other methods as it achieves the lowest network overhead while keeping a very high level of freshness of the membership information. Unlike our protocol, the non-adaptive tree method has good freshness of information but pays a much higher cost for the overlay construction as we have seen in Figure 5.6 and Figure 5.8. The flooding-based method and PAST-DM achieve higher corrected costs as they have both higher network overhead and bad freshness of information.

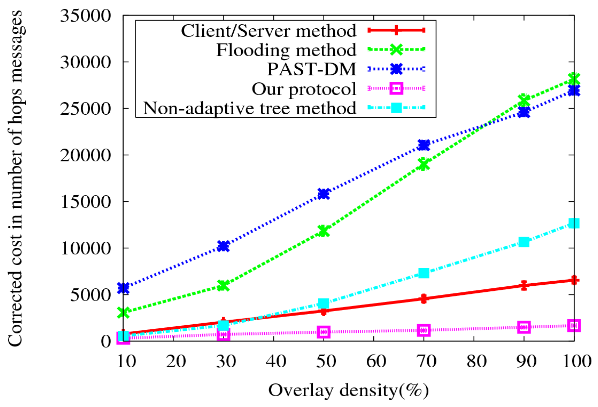


Figure 5.8: Corrected cost Vs. Overlay density

5.4.4.4 Network split awareness

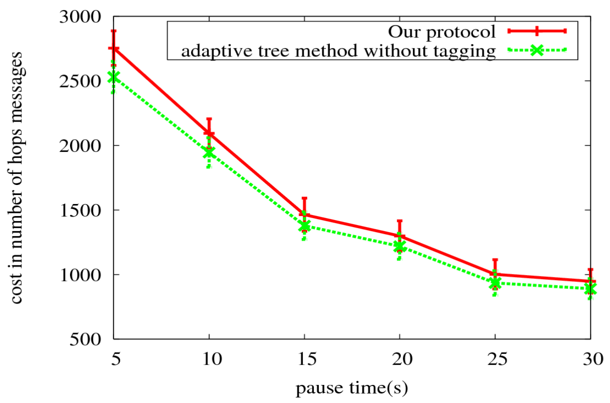


Figure 5.9: Split awareness: Cost Vs Pause time

The last set of simulations aims to study the efficiency of our solution for overlay splits versus the frequency of topology changes by varying the pause time of mobile nodes from 5 to 30s. A low value of pause time means frequent topology changes and more probable network splits. The density of the overlay is set to 50%. We evaluate the capacity of our protocol in handling network splits by simulating it in two modes: a mode that enables the split awareness mechanism and a second mode that disables it.

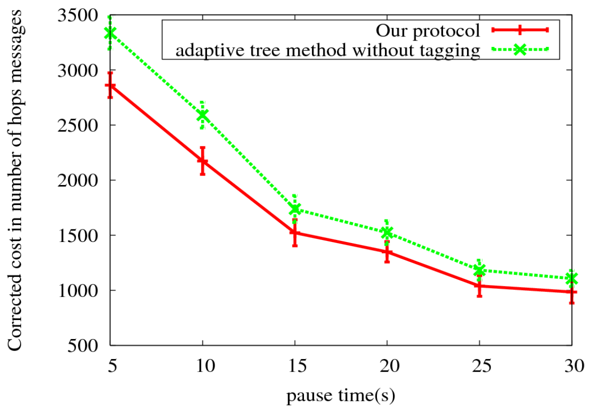


Figure 5.10: Split awareness: Corrected cost Vs Pause time

Figure 5.9 shows that the extra cost for handling network splits is relatively small even for low values of the pause time. In fact, the tagging mechanism is only working in the local neighborhood of peer nodes and peers joining again a part of the overlay quickly get the whole membership information from the nearest peer. However, Figure 5.10 shows that the corrected cost of the split-unaware variant of the protocol becomes higher than the corrected cost of the split-aware variant thanks to a better freshness of information. Hence, we conclude that our protocol provides an efficient and low-cost solution for MANET partitioning problem.

5.5 Conclusions and perspectives

P2P membership management is a hard and costly task in MANET. In this chapter, we presented a scalable, robust and network friendly protocol to construct an adaptive topology-aware tree allowing peers to discover each other and to keep themselves informed about the arrivals and departures of other peers. The proposed protocol is a standalone service that can be used by any application requiring the sharing of up-to-date information among a group of users. Moreover, our protocol minimizes the number of exchanged messages and copes with node mobility and network partitioning, which makes it very useful for applications to know where peers are located in the network and how far they are from each other. The simulations show that our protocol outperforms classical solutions in terms of network load and freshness of information. In the next chapters, we continue our design of a content sharing architecture fo wireless multi-hop networks. The membership management protocol presented in this chapter will be used to discover and update the members of the

P2P network. As a future work, one can profit from the generality and the simplicity to study the gain from using it in other peer-to-peer services like instant messaging, video streaming, social networks, etc.

Chapter 6

BitHoc: Implementation on real mobile handhelds

Contents

Content sharing is currently a universal concern among computer users and has recently become an important requirement for mobile handhelds. In fact, nowadays, the market proposes a variety of mobile devices offering user-friendly interfaces, long-life battery autonomy, sufficient computational power and efficient wireless connectivity. This tremendous advancement triggers the necessity of supporting the very fashionable desktop applications in such mobile environments. Indeed, thanks to the efficient wireless connectivity offered by mobile devices (PDAs, smartphones...), users are frequently brought to locate and share content of interest (data, photos, videos, etc) with other members of the same spontaneous community. With current technology, they are mainly using point-to-point basic connections, which can be considered as an efficient solution when the number of users interested in the sharing session is very small. However, in the case of a large community (for example, mobile handheld users assisting to a conference and willing to share some papers), one is facing the following problem: Increasing the number parallel point-to-point communications may decrease the global ad hoc network capacity, while increasing dramatically the download time. The multi-hop point-to-point communication over long paths is also a serious issue. Therefore, there is a strong need to organize the communication overlay among nodes in a way to distribute fairly the burden of data sharing among the set of participants while aiming to decrease the global download time. P2P file sharing solutions are good candidates for such infrastructureless networks as they are based on multi-sourcing which balances resource consumption among peers and reduces the dependency on any central entity. But unfortunately, P2P content sharing applications developed for the Internet cannot directly be plugged and used into mobile devices. Indeed, on one hand, these solutions are not adapted to the constraints of multi-hop wireless networks. For example, it is known that in a resource constrained environment, the choice of the peers to whom to connect cannot be done independently of information on the underlying dynamic topology. Moreover, centralized peer management approaches like the tracker used in BitTorrent do not perform well in such environment as the tracker can be either far away or even invisible by some peers because of disconnections. Furthermore, computer users rely on Internet search engines and dedicated desktop applications to look for the content they are willing to share. This approach becomes obsolete in the case of a spontaneous MANET based community and thus, a dedicated distributed content discovery approach must be provided. One has to add the fact that from a technical point of view, limited Software Development Kits (SDK) proposed for mobile handhelds represent only a sub-set of classical SDK(s) used for desktops which leads to incompatibility problems.

This chapter presents BitHoc, an open-source standalone software solution for content sharing in MANETs that overcomes the aforementioned challenges. This software is downloadable from our web site [Bit 2010a] and has been presented in different demonstration sessions of international conferences [Krifa 2009b][Krifa 2009a]. The architecture of BitHoc consists of three main components: a content discovery service (BitHoc Search Engine), a membership management service (distributed BitHoc Tracker) and a content sharing service (BitHoc Client). BitHoc tracker agents installed in nodes connect to each other in order to form and manage the per-content sharing overlay needed by the BitHoc client. This emulates the central tracker of standard BitTorrent. They also construct the content discovery overlay used by the BitHoc search engine to enable content publishing and discovery. To connect to a sharing overlay, a node needs to retrieve a torrent file (a meta-data file) by connecting to the discovery overlay. The members of this overlay manage a distributed database that contains for each torrent the list of nodes uploading it and a short description of the related content. To ensure content sharing, the BitHoc client decides, using routing information available at layer 3, of the structure of the data exchange overlay seen by him (with whom to exchange data). It also manages the scheduling of data pieces among devices. Our main contribution here is to adapt the peer neighbor and piece selection strategies of BitTorrent to account for the topology of the network and for the scarcity and shared nature of resources. The algorithms used in these three components of BitHoc are inspired from those presented in chapters 5 and 4.

We developed our software on mobile devices having Windows Mobile 6 [WM6 2010] as an operating system and equipped with WIFI adapters. As permanent topology information is required by BitHoc, we chose to use the proactive routing protocol OLSR [MOL 2010]. For the evaluation, we measured the performance of BitHoc with regard to the download time and sharing ratio metrics. Our test-bed allows us to experiment with the different features of our solution and to compare our version of the BitTorrent's algorithms to the ordinary Internet one when deployed in a wireless ad hoc environment. The performance analysis based on experimentation shows that BitHoc outperforms the standard version, which is in compliance with what has been shown throughout this thesis.

The remainder of this chapter is organized as follows. Section 6.1 analyses the objectives of BitHoc. Section 6.2 describes its architecture. Section 6.3 presents the test-bed we have used to test and evaluate our solution and we show some results.

6.1 Requirements Analysis

We designed our package in such a way to be standalone. So, the user has just to install the software to start publishing and discovering contents and sharing them later with those having the same interest. The wireless nodes not interested by the same content collaborate by forwarding packets at the routing level. Through BitHoc, we provide solutions to the following problems:

- In the classical version of BitTorrent [Bit 2010b], peers periodically contact a central rendezvous point called Tracker to obtain fresh information about the peers interested in a specific content and to update their information on the progress of the download. This membership information is dynamic since peers can join and leave the content sharing overlay (called torrent) at any time during the session. Because of the inappropriateness and the large overhead of client/server architectures in wireless ad hoc networks, it is important to introduce a distributed Trackerless solution to manage the membership of the sharing session. The BitHoc tracker component of our architecture is designed for this purpose and is inspired from the membership management protocol we presented in details in Chapter 5.

- The classical version of BitTorrent [Bit 2010b] supposes that the cost of sending data packets to peers is in somehow independent of their locations. In an ad hoc network, performance metrics like achievable throughput, delay, and energy consumption strongly depend on the number of hops to the peer node. So, it is clearly suboptimal and even unrealistic to deal with peers without considering the underlying topology. Furthermore, when applying the classical BitTorrent incentives in a wireless multi-hop network, nodes fail to reciprocate data fairly among them. The content dissemination scheme is close to a wave transferring data from the initial seed to the farthest peers. Through new peer selection and content piece scheduling strategies, our solution is topology-aware and ensures fair sharing. These strategies are described in details in Chapter 4.

- To join a sharing session, a user should find and download the Torrent file related to that session. In the Internet, peers usually find their torrent files by the help of search engines which mainly look for the files in different central servers. This method does not apply in a mobile ad hoc environment as MANETs. The BitHoc search engine overcomes this challenge by maintaining a distributed torrent file database thanks to the overlay constructed by the BitHoc Tracker.

6.2 Architecture of BitHoc

In this section, we describe the architecture of the BitHoc application. Figure 6.1 depicts the principal components of this architecture and the interactions between them. We illustrate these interactions through three typical usage scenarios:

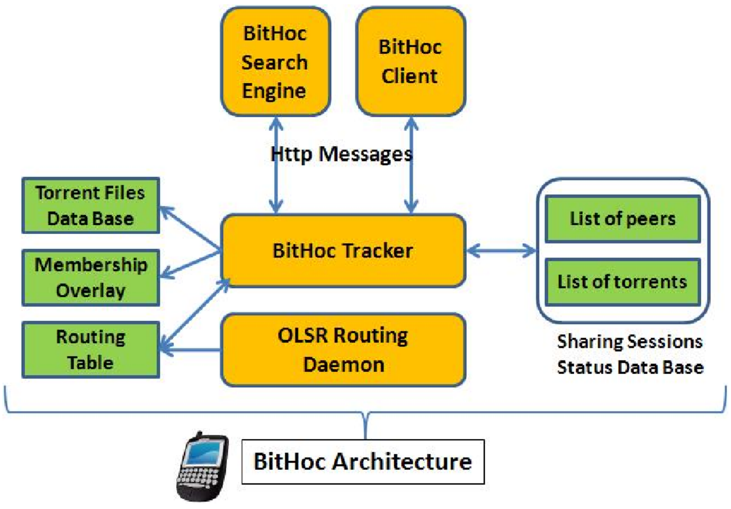
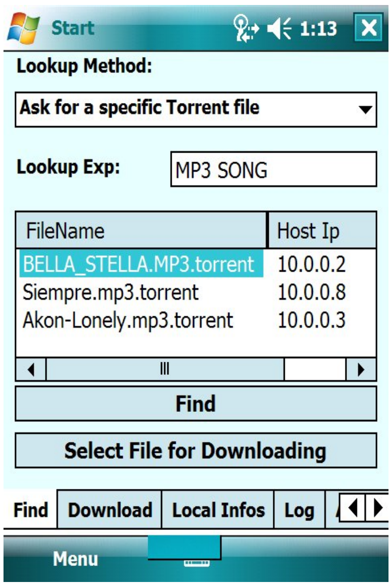


Figure 6.1: Architecture of BitHoc

6.2.1 Content publishing and discovery

A user willing to share some content with the members of his community needs to indicate to the BitHoc client the location of the content in the mobile device file system. First, the client creates a meta-info file (Torrent file) that identifies in a unique manner a sharing session for this specific content. After that, the user publishes (locally) the new torrent file and a short text description of the related content using the BitHoc Search Engine service, which will update the local Torrent file database maintained in the underlying BitHoc Tacker via HTTP messages. A remote user, willing to share the same content, has to use the BitHoc search engine to find and download the Torrent file. He specifies for that the name of the content or some keywords related to its description. The request is sent via HTTP messages to its local tracker which looks for the closest match in its local database. If there are no matches, it forwards the HTTP request to the other trackers in the discovery overlay. Then, it presents the received results through an ergonomic user interface (see Figure 6.2). Based on the details of received answers (fitness to the search, number of peers involved in the sharing session, number of seeders, and number of lechers, etc), the user can choose the torrent file to download, then start sharing the content using the BitHoc Client.



Lookup Method: Ask for a specific Torrent file

Lookup Exp: MP3 SONG

FileName	Host Ip
BELLA_STELLA.MP3.torrent	10.0.0.2
Siempre.mp3.torrent	10.0.0.8
Akon-Lonely.mp3.torrent	10.0.0.3

Find

Select File for Downloading

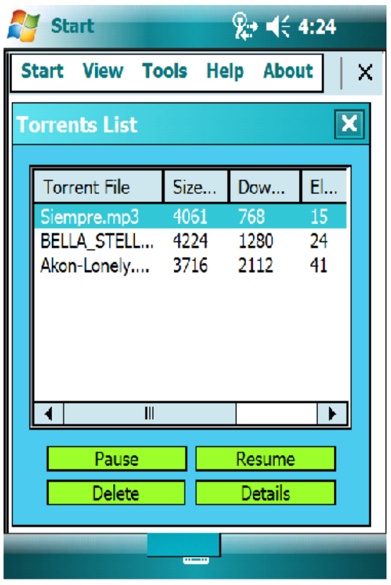
Figure 6.2: Search Engine screen shot

6.2.2 Membership management

When a peer wants to join or leave the sharing session, the BitHoc client informs the BitHoc Tracker about this event using a specific HTTP message. This local agent disseminates this modification to the other BitHoc Tracker agents in other nodes in order to update their knowledge about the global membership information. The communications between Tracker agents are established in an event-driven fashion and use HTTP messages. Each tracker holds a HTTP server accepting HTTP requests from other agents and from the local BitTorrent client. The BitHoc Tracker component receives from the routing daemon up-to-date routing entries. In our testbed, the dynamics of the ad hoc network are captured by the OLSR routing protocol [MOL 2010]. Each time the number of hops toward a given peer changes, the routing daemon fires an event, which will be caught by the BitHoc Tracker and forwarded internally to the BitHoc client. This way we are sure the peer selection strategy always uses the updated number of hops to other peers. The parameters of the communications among tracker agents like HTTP listening ports and IP addresses can easily be configured by users via an ergonomic GUI. In addition to these functionalities, the BitHoc Tracker allows the user to monitor in real-time the status of the overlay (Contents it shares, members of the session, current topology of the ad hoc network). He can even decide to keep traces about all the events in a file. For this, he just needs to activate the tracing option provided by the application.

6.2.3 Content sharing

Before starting a new sharing session, the user can choose between two versions of BitTorrent algorithms: The classical version [Bit 2010b] and our version adapted to mobile ad hoc networks described in Chapter 4. The BitHoc client offers a Wizard allowing the user to configure the parameters of BitTorrent (communication ports, choking slot duration, minimum and maximum number of peers, etc). Once the torrent file is obtained, the BitHoc client can start the sharing session where it can either play the role of a leecher or a seed. It contacts periodically the local BitHoc tracker to get the current list of members of the same content sharing session (torrent). Using this list and the routing table, it manages the connections with the interested peers. Briefly a client implementing our algorithms exchanges pieces with close peers and only seeds distribute pieces across the network. Note that we allow the user to pause or resume the download while conserving the session context. He can also monitor in real time the status of the session (downloaded bytes, uploaded bytes, numbers of leechers, number of seeders, elapsed time, etc). Furthermore, the BitHoc client keeps in a log file statistics on the content sharing session and provides different levels of event traces. It also manages the storage of the downloaded contents and their classification. Figure 6.3 shows a screenshot of the BitHoc client.



Torrent File	Size...	Dow...	El...
Siempre.mp3	4061	768	15
BELLA_STELL...	4224	1280	24
Akon-Lonely,...	3716	2112	41

Figure 6.3: BitHoc Client screen shot

6.3 Experimentation and results

6.3.1 Test-bed description

Our wireless ad hoc network experimental environment consists of 14 mobile devices including 7 PDAs (HP iPAQ 214) and 7 smartphones (HP iPAQ 614c). Each handheld is equipped with an IEEE802.11b wireless card. The characteristics of the two types of devices are detailed in Table 6.3.1. The ad hoc connectivity is maintained thanks to OLSR daemons run by the different devices. In our experiments, we constructed several network topologies containing a maximum of 6 hops. The objective of the realized swarm was to download a 4 MB MP-3 content. All PDAs were supposed to participate to the sharing of the file. The original seed of the content was chosen randomly among the set of the 14 PDAs.

Table 6.1: Characteristics of mobile handhelds

	PDA	**Smartphone**
Name	HP iPAQ 214	HP iPAQ 614c
Processor speed	624 MHz	520 MHz
RAM	128 MB	128 MB
Operating system	Windows Mobile 6	Windows Mobile 6

6.3.2 Experimentation results

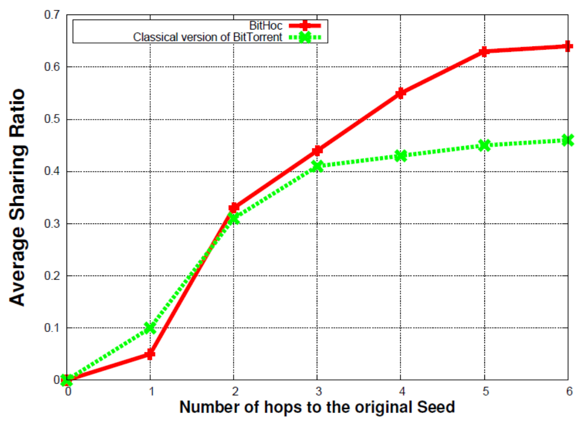


Figure 6.4: Sharing ratio

The metrics tracked during our experiments are the download time and the average sharing ratio of nodes. We define R_h as the sharing ratio of peers located at h hops from the original seed. It measures the level of reciprocity between downloads and uploads. In the ideal case, the ratio should be close to 1. The two versions of BitTorrent (The legacy one and ours) have been tested and the results are presented

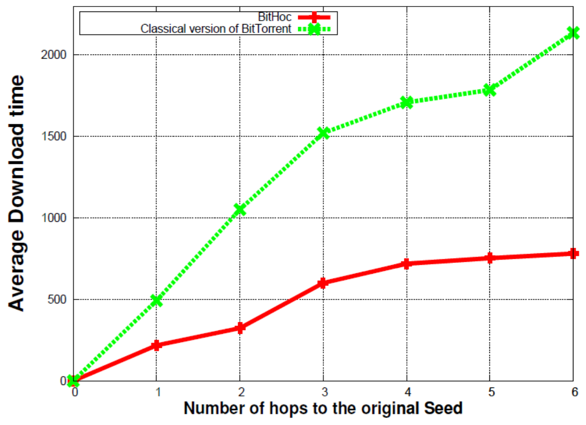


Figure 6.5: Download time

in Figures 6.4 and 6.5. Figure 6.4 shows a dramatic increase of sharing opportunities when our adapted version is deployed. The routing overhead generated by the classical version makes any gain obtained by important diversification of pieces negligible. Our method finds the good equilibrium between sharing and diversification. Figure 6.5 shows that BitHoc outperforms the classical version of BitTorrent in terms of download time. It is in accordance with our research results presented in chapter 4. More information about our experiments and our GPL licensed open-source code can be found on the BitHoc web site [Bit 2010a].

6.4 Conclusions and perspectives

In this Chapter, we described our solution for content sharing in wireless ad hoc networks. The implementation that we have proposed contains three components: a distributed membership management service, a content search engine and an optimized content sharing service. The design of these services takes into consideration the constraints of mobile environments and the user needs. With the help of a real test-bed composed of PDAs and Smartphones, we were able to validate our solution in a real scenario and to compare it to other classical approaches. The experiments show that our application outperforms the classical approach and highlight the utility of its features. In particular, we were able to reduce by a factor of 2 to 3 the download time and to increase dramatically the sharing ratio. As a perspective, we plan to implement the application on other types of devices and do extensive experiments with real users.

Chapter 7

Conclusions and perspectives

P2P networks and wireless ad hoc networks share some common characteristics such as decentralization and auto-organization. Starting from these similarities and encouraged by the advances in mobile technologies, we studied in this thesis how to deploy content sharing applications designed for the Internet in wireless ad hoc networks. By evaluating the performance of the well-known Internet version of BitTorrent, we showed that its neighborhood selection mechanism and its choke algorithm are not adapted to the wireless environment. In fact, selecting peers independently of their locations in the network is proven in this thesis to generate a lot of routing overhead. One can imagine that pieces of the content transit at the network layer of peers without profiting from them at the application level. Furthermore, connections to far away nodes suffer from very low throughputs in wireless networks. The first intuitive solution we had was to limit the scope of the neighborhood to some routing hops and hence reducing the routing overhead. By reducing the sharing scope, we were able to diminish the download time of peers compared to the Internet version of BitTorrent. However, whenever only a few nodes are interested in the content, having a narrow neighborhood disconnects the sharing overlay and some peers, isolated from the rest of the P2P network, never finish the download. Hence, there is a tradeoff between decreasing the routing overhead and ensuring the connectivity of the overlay.

In our study of BitTorrent in wireless ad hoc networks, we considered another important factor which was sharing opportunities. As the goal of BitTorrent is to distribute fairly the burden of data transfers among the set of peers and to ensure that a peer contributes to the P2P network as much as it receives from others, we measured in our work to which extent the Internet version of BittTorrent realizes these fair sharing objectives. Through our experiments, we discovered that low sharing ratios are recorded for all sharing scopes. Mainly, the lowest levels of sharing are obtained for very narrow neighborhoods since pieces of the file will propagate from the original source of the content to edge peers. Hence, this generates a very low diversity of pieces and as a result, peers do not have original pieces to reciprocate. So, unlike in the case of download time, decreasing the sharing scope is not the right solution to increase sharing opportunities. However, in case of a mobile network, our simulations show that mobility of nodes shuffles the pieces in the network. Consequently, one obtains a higher diversity of pieces and better sharing ratios. Hence, in this case, we show that it is sufficient to reduce the sharing scope to some routing hops. The gain in diversity obtained by communicating with far away peers is not important compared to that obtained by the mobility. Furthermore, in a mobile network, paths to far away nodes are instable and have very low throughputs.

From this study of the performance of BitTorrent in the wireless ad hoc environment, we concluded that it is important to reduce the routing overhead while keeping the connectivity of the overlay and that unless the network is mobile, one needs to add some connections to far away peers to increase the diversity of pieces. To answer these requirements, we designed a content sharing protocol adapted to the constraints of wireless ad hoc networks. To minimize the routing overhead and ensuring the connectivity of the overlay, our protocol organizes peers in a minimum spanning tree in terms of the number of routing hops and selects the neighborhood of peer as being the peers located at a routing distance lower than number of hops to peers located at one hop on this tree. Using this tree strategy, we reduced considerably the download time of peers and ensured that 100 % of peers completed the download.

Knowing that reducing the scope following the spanning tree decreases the sharing opportunities, we modified the choke algorithm of seeds in order to send some pieces to far away peers. The fourth connection of a seed is sometimes used to diversify the content in the network. This diversification effort is divided between seeds in time and space in such a way to keep the global diversification load constant during the sharing session. Knowing that the scope of pieces varies with the level of the congestion of the network and with the mobility of nodes, we limit the range of the diversification connections to the range of the transfer of a complete piece. As this range varies during time, our protocol adapts the range of diversification to the observed range of pieces using a specific algorithm. On one hand, as the gain obtained from the diversification in a congested network is negligible, our adaptive algorithm ensures that the diversification scope shrinks when the number of Torrents in the network increases. On the other hand, the mobility of nodes decreases the range of pieces and then, our protocol diminishes the scope of diversification. This is a searched behavior since mobility increases the diversity of pieces and there is no need to communicate with far away nodes. As a consequence, our protocol adapts the diversification effort to cross traffic and mobility of nodes.

The data transfer plane of a P2P content sharing application can not work without the help of a membership management protocol. The latter protocol discovers the peers interested in the content and updates their list during the sharing session. In the Internet, the membership is ensured thanks to dedicated servers. For instance, peers in the case of BitTorrent contact a central rendezvous point, called Tracker, to obtain up-to-date information on the members of the overlay. Unfortunately, the client/server model is not adapted to wireless ad hoc networks. In fact, even if one of the peers devotes its upload capacity to play the role of a server, the membership management service will never scale since the peer server remains its bottleneck and is its single point of failure. In a wireless network, the server can be isolated from the rest of peers due network partitioning caused by the mobility. In this thesis, we designed a fully distributed and efficient membership management mechanism, which is adapted to the nature of mobile ad hoc networks. Our protocol organizes peers in a minimum spanning tree in terms of the number of hops. This tree which is also used for the neighborhood selection mechanism is continuously adapted to the underlying topology of the networks and to arrivals and departures of nodes. Compared to classical methods such as flooding and multicast, our mechanism yields the lowest overhead while having the best freshness of information on the members of the overlay. Another important feature of our protocol is network partitioning awareness. In fact, separate trees belonging to the same content can merge together when communication becomes possible between two separate partitions of the wireless ad hoc network.

In the case of high mobility speeds or intermittent connectivity networks, we have shown in this thesis that there is no need to diversify pieces of the content in the network and that a narrow neighborhood seems to be very sufficient and optimal. That is why solutions based on adapting the Content-Centric Networking paradigm to opportunistic wireless networks seem to be good candidate solutions in this case. The research community is offering protocols and studies for this particular cases.

To consolidate the NS-2 simulations and the ORBIT experiments, we implemented our protocols on real mobile handhelds and verified the obtained results. Moreover, the real implementation called BitHoc [Bit 2010a], is available on our web site and allows users to deploy our protocols by forming real spontaneous communities of their mobile devices. They are able to search for contents, to discover other users interested in the same contents and to efficiently upload/download files.

As a future work, we aim at implementing our protocols for other types of devices and experiment with real large scale communities of users. Furthermore, the membership management mechanism can be used for other applications like video streaming, instant messaging and social networking. We plan to test these applications on the top of our membership mechanism and to study their data planes. Many optimizations can be done on the data transfer planes of these P2P applications by observing the analogy with the work we did on BitTorrent. Moreover, one can go one step further with the study of the data plane of BitTorrent by deploying inter-Torrent incentives in the case of many Torrents in the same network. In deed, one can suppose that two peers can exchange pieces belonging to different contents and in this way, the entropy of pieces in the network is increased and the incentives for collaboration becomes more efficient. This study of inter-Torrent incentives and other optimizations are left as a future work.

Bibliography

[802 2010] *WWW pages of the IEEE 802.11 standards*, 2010. http://www.ieee802.org/11/. (Cited on page 85.)

[A. Balasubramanian 2007] B. N. Levine A. Balasubramanian and A. Venkataramani. *Dtn routing as a resource allocation problem.* In ACM SIGCOMM, 2007. (Cited on page 33.)

[A. Krifa 2008] C. Barakat A. Krifa and T. Spyropoulos. *An optimal joint scheduling and drop policy for delay tolerant networks.* In WoWMoM workshop on Autonomic and Opportunistic Communication (AOC), 2008. (Cited on page 33.)

[A. Lindgren 2003] A. Doria A. Lindgren and O. Schelen. *Probabilistic routing in intermittently connected networks.* In SIGMOBILE Mobile Computing and Communicatio Review, vol. 7, no. 3, 2003. (Cited on page 33.)

[A. Pentland 2004] R. Fletcher A. Pentland and A. Hasson. *Daknet: Rethinking connectivity in developing nations.* In IEEE Computer, 2004. (Cited on page 32.)

[Al-Hamra 2009] Anwar Al-Hamra, Nikitas Liogkas, Arnaud Legout and Chadi Barakat. *IEEE ICCCN conference, San Francisco.* In Swarming Overlay Construction Strategies, 2009. (Cited on page 15.)

[Alan Demers 1987] Carl Hauser Wes Irish John Larson Scott Shenker Howard Sturgis Dan Swinehart Alan Demers Dan Greene and Doug Terry. *Epidemic algorithms for replicated database maintenance.* In ACM Symposium on Principles of Distributed Computing, 1987. (Cited on page 33.)

[Basu 2002] P. Basu and T. Little. *Networked parking spaces: architecture and applications.* In IEEE Vehicular Technology Conference (VTC), 2002. (Cited on page 32.)

[BGP 1995] *Y. Rechtet and T.Li. A Border Gateway Protocol 4 (BGP-4)*, 1995. RFC 1771, IETF. (Cited on page 22.)

[Bharambe 2005] A. R. Bharambe, C. Herley and V.N. Padmanabhan. *Performance Evaluation Review, 33(1):398399.* In Some Observations on BitTorrent Performance, 2005. (Cited on page 6.)

[Bit 2010a] *WWW pages of BitHoc*, 2010. http://planete.inria.fr/bithoc/. (Cited on pages 94, 100 and 103.)

[Bit 2010b] *WWW pages of BitTorrent*, 2010. http://wiki.theory.org/bittorrentspecification. (Cited on pages 1, 6, 8, 13, 38, 53, 72, 95 and 98.)

[Boyd 2005] S. Boyd, A. Ghosh, B.Prabhakar and D.Shah. *Gossip Algorithms: Design, Analysis, and Applications.* In IEEE Infocom, 2005. (Cited on page 74.)

[Broch 1998] J. Broch, D.A. Maltz, D.B. Johnson Y.-C.Hu and J.Jetcheva. *A performance comparaison of multi-hop wireless ad hoc network routing protocols.* In Mobile Computing and networking, pp 85-97, 1998. (Cited on page 26.)

[Carzaniga 2003] A. Carzaniga and A. L. Wolf. *Forwarding in a content-based network.* In ACM SIGCOMM, 2003. (Cited on page 32.)

[Castro 2003a] M. Castro, P. Druschel, A.M. Kamarrec, A. Nandi, A. Rowstorn and A. Singh. *Splitstream: high-bandwidth multicast in a cooperative environment.* In ACM SOSP, 2003. (Cited on page 24.)

[Castro 2003b] M. Castro, P. Druschel, A.-M. Kermarrec and A. Rowstorn. *Scribe: a large-scale and decentralized application level multicast infrastructure.* In IEEE Journal on Selected Areas in Communication (JSAC), 2003. (Cited on page 24.)

[Cohen 2003] B. Cohen. *the First Workshop on Economics of Peer-to-Peer Systems, Berkeley, USA.* In Incentives build robustness in bittorrent, 2003. (Cited on page 15.)

[CTC 2010] *WWW-pages of Client-To-Client Protocol (CTCP)*, 2010. http://www.irchelp.org/irchelp/rfc/ctcpspec.html. (Cited on page 6.)

[Dabek 2001] F. Dabek, M.F. Kaashoek, D. Karger, R. Morris and I.Stoicachel. *Wide-area cooperative storage with CFS.* In ACM SOSP, 2001. (Cited on page 24.)

[Das 2004] S.M. Das, H. Pucha, and Y.C. Hu. *Ekta : an efficient peer-to-peer substrate for distributed applications in mobile ad-hoc networks*, 2004. Technical Report TR-ECE-04-04, Purdue University. (Cited on pages 1, 29 and 53.)

[DCC 2010] *WWW-pages of Direct Client-to-Client (DCC)*, 2010. http://en.wikipedia.org/wiki/Direct_Client-to-Client. (Cited on page 6.)

[DSR 1996] *D.B. Johnson and D.A. Maltz. Dynamic Source Routing in Ad Hoc Wireless Networks*, 1996. Kluwer Academic. (Cited on pages 19, 20, 23, 25, 26, 28 and 29.)

[DTN 2010] *Delay tolerant networking research group*, 2010. http://www.dtnrg.org. (Cited on page 32.)

[DVM 1988] *S. Deering, C.Partridge, and D. Waitzman, Distance vector multicast routing protocol.*, 1988. Technical Report 1075, IETF. (Cited on page 22.)

[eDo 2010] *WWW-pages of eDonkey*, 2010. `http://en.wikipedia.org/wiki/EDonkey_network`. (Cited on pages 6 and 8.)

[Erik Nordstrom] Per Gunningberg Erik Nordstrom and Christian Rohner. *A Search-based Network Architecture for Mobile Devices.* In Uppsala University, Sweden. (Cited on pages 32 and 34.)

[Fall 2003] K Fall. *A delay-tolerant network architecture for challenged internets.* In ACM SIGCOMM, 2003. (Cited on page 32.)

[Gnu 2010] *Gnutella Specification*, 2010. `http://dss.clip2.com/GnutellaProtocol04.pdf`. (Cited on pages 1, 6, 9, 23, 24, 26, 28 and 74.)

[Gui 2003] C. Gui and P.Mohapatra. *Efficient Overlay Multicast.* In WCNC, 2003. (Cited on pages 75 and 87.)

[Harjula 2004] E. Harjula, M. Ylianttila, J. Ala-Kurikka, J. Riekki and J. Sauvola. *the 3rd international conference on Mobile and Ubiquitous Multimedia (MUM 2004), pp: 63-69,Â College Park, Maryland, USA.* In Plug-and-Play Application Platform: Towards Mobile Peer-to-Peer, 2004. (Cited on pages 6, 7, 8, 9 and 10.)

[Hu 2004] Y.Charlie Hu, Saumitra M.Das and Himabindu Pucha. *Peer-to-Peer Overlay Abstractions in MANETs.* In Handbook on Theoritical and Algorithmic Aspect of sensors. pp-857-874., 2004. (Cited on page 26.)

[Hyytia 2006] Esa Hyytia, Pasi Lassila and Jorma Virtamo. *Spatial Node Distribution of the Random Waypoint Mobility Model with Applications.* In IEEE Transactions on Mobile Computing, 2006. (Cited on pages 49, 68 and 85.)

[ICQ 2010] *WWW-pages of ICQ*, 2010. `www.icq.com`. (Cited on page 6.)

[IRC 2010] *WWW-pages of IRC*, 2010. `http://www.irchelp.org/irchelp/new2irc.html`. (Cited on page 6.)

[Jacobson 2009] Van Jacobson, Diana K. Smetters, James D. Thornton, Michael F. Plass, Nicholas H. Briggs and Rebecca L. Braynard. *Networking named content.* In Proceedings of the 5th international conference on Emerging networking experiments and technologies, CoNEXT '09, pages 1–12, New York, NY, USA, 2009. ACM. (Cited on page 32.)

[Jacquet 2001] P. Jacquet, P. MÃ$\frac{1}{4}$hlethaler, T Clausen, A. Laouiti, A. Qayyum and L. Viennot. *Optimized Link State Protocol for Ad Hoc Networks.* In IEEE INMIC, 2001. (Cited on pages 19 and 85.)

[Jetcheva 2001] J.G. Jetcheva and D.B Johnson. *Adaptive demand-driven multicast routing in muti-hop wireless ad hoc networks.* In ACM MobiHoc, 2001. (Cited on page 23.)

[Kad 2010] *WWW-pages of Kademlia*, 2010. http://en.wikipedia.org/wiki/Kademlia. (Cited on page 16.)

[KaZ 2010] *WWW-pages of KaZaA*, 2010. http://en.wikipedia.org/wiki/Kazaa. (Cited on pages 6, 8 and 23.)

[Klemm 2003] A. Klemm, C. Lindermann and O. Waldhorst. *A special-purpose peer-to-peer file sharing system for mobile ad-hoc networks.* In IEEE VTC, 2003. (Cited on pages 1, 28 and 53.)

[Krifa 2009a] Amir Krifa, Mohamed Karim Sbai, Chadi Barakat and Thierry Turletti. *A content sharing application for wireless ad hoc networks, demo description.* In IEEE Percom, Texas, 2009. (Cited on page 94.)

[Krifa 2009b] Amir Krifa, Mohamed Karim Sbai, Chadi Barakat and Thierry Turletti. *A standalone content sharing application for spontaneous communities of mobile handhelds, demo description.* In ACM SIGCOMM MobiHeld Workshop, Barcelona, 2009. (Cited on page 94.)

[Kru 2010] *WWW pages of the Kruskal algorithm*, 2010. http://en.wikipedia.org/wiki/Kruskal's_algorithm. (Cited on page 77.)

[Lee 2001] S.-J. Lee, W. Su and M. Gerla. *On-demand multicast routing protocol in multihop wireless mobile networks.* In Kluwer Mobile Networks and applications, 2001. (Cited on page 23.)

[Legout 2006] A. Legout, G. Urvoy-Keller and P. Michiardi. *ACM SIGCOMM/USENIX IMC'2006, Rio de Janeiro, Brazil.* In Rarest First and Choke Algorithms Are Enough, 2006. (Cited on page 15.)

[Lenders 2007] Karlsson G. Lenders V. and M May. *Wireless ad hoc podcasting.* In IEEE Communications Society Conference on Sensor, Mesh, and Ad Hoc Communications and Networks (SECON), 2007. (Cited on page 32.)

[Li 2000] J. Li, J.Jannotti, D.S.J.D. Couto, D.R.Karger and R.Morris. *A scalable location service for geographic ad-hoc routing.* In ACM MobiCom, 2000. (Cited on page 30.)

[Lib 2010] *LibTorrent's web site*, 2010. http://libtorrent.rakshasa.no/. (Cited on page 40.)

[Mad 2010] *MadWifi Project Team*, 2010. http://madwifi.org. (Cited on page 40.)

[Mic 2010] *WWW-pages of Microsoft Messenger*, 2010. http://messenger.msn.com/. (Cited on page 6.)

[Michiardi 2007] P. Michiardi and G. Urvoy-Keller. *Performance analysis of cooperative content distribution for wireless ad hoc networks.* In WONS, Obergurgl, Austria., 2007. (Cited on page 53.)

[MOL 2010] *WWW pages of OLSR for Windows Mobile*, 2010. `http://www.grc.upv.es/software/introolsr.html`. (Cited on pages 94 and 97.)

[MOS 1994] *J.Moy. Multicast extensions to OSPF*, 1994. Technical Report 1584, IETF. (Cited on page 22.)

[Nap 2010] *WWW-pages of Napster*, 2010. `http://en.wikipedia.org/wiki/Napster`. (Cited on page 7.)

[NS- 2010] *WWW pages of NS-2: The Network Simulator*, 2010. `http://www.isi.edu/nsnam/ns/`. (Cited on pages 38, 39, 54, 68, 73, 83 and 85.)

[Olafur R. Helgason 2010] Sylvia T. Kouyoumdjieva Ljubica Pajevic Gunnar Karlsson Olafur R. Helgason Emre A. Yavuz. *A mobile peer-to-peer system for opportunistic content-centric networking.* In second ACM SIGCOMM workshop on Networking, systems, and applications on mobile handhelds, 2010. (Cited on pages 32 and 34.)

[Oliviera 2003] L.B. Oliviera, I.G. Siqueira and A.A. Loureiro. *Evaluation of ad-hoc routing protocols under a peer-to-peer application.* In IEEE WCNC, 2003. (Cited on pages 1, 26 and 28.)

[OOL 2010] *OOLSR project*, 2010. `http://hipercom.inria.fr/olsr/`. (Cited on page 40.)

[ORB 2010] *Rutgers ORBIT project team*, 2010. `http://www.orbit-lab.org`. (Cited on pages 38, 40 and 54.)

[OSP 1994] *J.Moy. OSPF version 2*, 1994. RFC 1583, IETF. (Cited on page 22.)

[P2P 2007] *Internet study*, 2007. `http://www.ipoque.com/resources/internet-studies/internet-study-2007`. (Cited on page 6.)

[P2P 2010] *WWW pages of P2P SIP*, 2010. `http://www.p2psip.org/`. (Cited on page 74.)

[Perkins 1994] Charles E. Perkins and Pravin Bhagwat. *Highly Dynamic Destination-Sequenced Distance-Vector Routing (DSDV) for mobile computers.* In SIGCOMM 1994: 234-244, 1994. (Cited on pages 18 and 26.)

[Perkins 1999] C.E. Perkins and E.M. Royer. *Ad-hoc on-demand distance vector routing.* In IEEE WMCSA, 1999. (Cited on pages 20, 23, 25, 26 and 28.)

[Ratnasamy 2001] S. Ratnasamy, P. Francis, M. Handley, R. Karp and S. shenker. *A scalable content-adressable networks.* In ACM SIGCOMM, 2001. (Cited on pages 9, 13, 23 and 26.)

[Ratnasamy 2002] S. Ratnasamy, B. Karp, L. Yin, F. Yu, D. Estrin, R. Govindan and S.Shenker. *GHT: a geographic hash table for data-centric storage in sensornets*. In 1st ACM WSNA, 2002. (Cited on page 31.)

[RIP 1988] *C.Hendrick. Rounting Information Protocol*, 1988. RFC 1058. (Cited on page 22.)

[Rodolakis 2007] G. Rodolakis, A. Naimi and A. Laouiti. *Multicast Overlay Spanning Tree Protocol for Ad Hoc Networks*. In WWIC, 2007. (Cited on page 75.)

[Rowstron 2001a] A. Rowstron and P. Druschel. *PAST: a large scale, persistent peer-to-peer storage utility*. In ACM SOSP, 2001. (Cited on page 24.)

[Rowstron 2001b] A. Rowstron and P. Druschel. *Pastry: scalable, distributed object location and routing for large-scale peer-to-peer systems*. In Middleware, 2001. (Cited on pages 12, 23, 26 and 29.)

[S. Jain 2004] K. Fall S. Jain and R. Patra. *Routing in a delay tolerant network*. In ACM SIGCOMM, 2004. (Cited on page 32.)

[Salhi 2008] Emna Salhi, Mohamed Karim Sbai and Chadi Barakat. *Neighborhood selection in mobile P2P networks*. In the 11th Algotel conference, Carry-Le-Rouet, France, 2008. (Cited on page 39.)

[Sbai 2008] Mohamed Karim Sbai, Chadi Barakat, Jaeyoung Choi, Anwar Al Hamra and Thierry Turletti. *Adapting BitTorrent to wireless ad hoc networks*. In Ad-Hoc Now Networks and Wireless conference, Sophia Antipolis, 2008. (Cited on pages 39 and 53.)

[Sbai 2009a] Mohamed Karim Sbai and Chadi Barakat. *Revisiting content sharing in wireless ad hoc networks*. In the fourth workshop on self-organizing systems (IWSOS), Zurich, 2009. (Cited on pages 39, 53 and 54.)

[Sbai 2009b] Mohamed Karim Sbai, Emna Salhi and Chadi Barakat. *A membership management protocol for mobile P2P networks*. In the Mobility Conference, Nice, 2009. (Cited on pages 55, 68 and 73.)

[Sbai 2010] Mohamed Karim Sbai, Emna Salhi and Chadi Barakat. *P2P content sharing in spontaneous multi-hop wireless networks*. In IEEE COMSNETS conference, Bengalore, 2010. (Cited on pages 39 and 54.)

[Sen 2004] S. Sen and J. Wang. *Analyzing peer-to-peer Traffic Across Large Networks*. In IEEE/ACM Transactions on Networking 12, 2, 219-232, 2004. (Cited on page 8.)

[SHA 1995] *Secure Hash Standard. Technical Report Publication 180-1*, 1995. Federal Information Processing standard (FIPS), NIST, U.S. Department of Commerce, Washington. (Cited on pages 13 and 30.)

[Stoica 2001] I. Stoica, R. Morris, D. Karger, M.F. Kaashoek and H. Balakrishnan. *Pastry: scalable, distributed object location and routing for large-scale peer-to-peer systems.* In ACM SIGCOMM, 2001. (Cited on pages 13, 23 and 26.)

[Su 2007] Scott J. Hui P. Crowcroft J. Diot C. Goel A. de Lara E. Lim M. H. Su J. and E. Up-ton. *Haggle: Seamless networking for mobile applications.* In UbiComp, 2007. (Cited on page 32.)

[Tang 2004] C. Tang, Z. Xu and S. Dwarkadas. *Peer-to-peer information retrieval using self-organizing semantic overlay networks.* In ACM SIGCOMM, 2004. (Cited on page 24.)

[Tap 2001] *B.Y. Zhao, J.D. Kubiatowicz, and A.D. Joseph. Tapestry: an infrastructure for fault-resilient wide-area location and routing*, 2001. Technical Report UCB//CSD-01-1141, U. C. Berkeley. (Cited on pages 13, 23 and 26.)

[Vahdat 2000] A. Vahdat and D. Becker. *Epidemic routing for partially connected ad hoc networks.* In Duke University, Tech. Rep. CS-200006, 2000. (Cited on page 33.)

[Werner Vogels 2002] Robbert van Renesse Werner Vogels and Ken Birman. *The power of epidemics: Robust communication for large-scale distributed systems.* In First Workshop on Hot Topics in Networks (HotNets-I), 2002. (Cited on page 33.)

[WM6 2010] *WWW pages of Windows Mobile 6*, 2010. `http://www.microsoft.com/windowsmobile/en-us/default.mspx`. (Cited on page 94.)

[X. Zhang 2006] J. Kurose X. Zhang G. Neglia and D. Towsley. *Performance modelling of epidemic routing.* In IFIP Networking, 2006. (Cited on page 33.)

[Xie 2002] J. Xie, R.R. Talpade, A. Mcauley and M. Liu. *AMRoute: ad hoc multicast routing protocol.* In Mob. Netw. Appl., 7(6):429-439, 2002. (Cited on page 23.)

Architecture for content sharing in wireless networks

Abstract: P2P networks utilize efficiently resources shared by end-users. Hence, the global service is not based on the existence of any central server but on the collaboration among the users, which connect to each others forming spontaneous overlays. For instance, BitTorrent is a P2P content sharing protocol where users are both content providers and requestors. When a peer receives some pieces of the content, it sends them to others following some specific algorithms that encourage fair collaboration among peers.

Some characteristics of P2P networks such as decentralization and auto-organization are also basic proprieties of mobile ad hoc networks (MANETs). In fact, a MANET is a spontaneous network of mobile nodes connecting to each other through wireless links. As the range of a wireless communication is limited, packets are routed through other nodes to reach the majority of destinations. Hence, MANETs must be self-organized and have to use efficient routing protocols.

In this thesis, we profited from the synergy between P2P networks and MANETs to design efficient content sharing applications for MANETs. Despite of the similarities between the two networks, Internet P2P applications can not be directly deployed in MANETs. In fact, they are not designed for environments where resources are limited and shared among nodes. Moreover, nodes are both routers and end-users. Hence, if the mechanisms managing the P2P overlay do not adapt to the dynamic underlying network, this will generate a big routing overhead. In this work, we design a content sharing protocol that takes into consideration the limitations of wireless networks and which is aware of the fundamental differences between the Internet and MANETs. The objective is to minimize the global download time of users by decreasing the network load and ensuring fair cooperation among peers. For this, our protocol adapts the algorithms of BitTorrent, mainly the neighborhood selection algorithm to the topology of the wireless network, the mobility of nodes and cross traffic.

In addition to considering the data plane, we design efficient algorithms to discover and update the members of a P2P overlay in MANETs. Our membership management protocol adapts to arrivals and departures of peers and changes in the topology due to the mobility of nodes. The objective is to minimize the membership traffic while keeping a good freshness of the information about the members.

Our protocols have been validated through extensive NS-2 simulations and real experiments over the ORBIT plateform. To ensure the feasibility of these protocols, we implemented BitHoc a content sharing application that implements our protocols on real mobile devices and did further experiments in spontaneous networks of handhelds.

Keywords: content sharing, P2P, wireless ad hoc, membership management