Objective FP7-ICT-2009-5-257448/D-2.9

Future Networks

Project 257448

"SAIL – Scalable and Adaptable Internet Solutions"

D-2.9
(D.A.9) Description of Overall Prototyping Use Cases, Scenarios and Integration Points

Date of preparation: **2012-06-11**
Revision: **1.0**
Start date of Project: **2010-08-01**
Duration: **2013-01-31**
Project Coordinator: **Thomas Edwall**
Ericsson AB

Document Properties

Document Number:	D-2.9
Document Title:	**Description of Overall Prototyping Use Cases, Scenarios and Integration Points**
Document Responsible:	Benoit Tremblay (EAB)
Document Editor:	Benoit Tremblay (EAB), Michael Soellner (ALUD)
Authors:	Ramon Aguero (UC) Pedro A. Aranda (TID) Anders E. Eriksson (EAB) Paulo Gonçalves (INRIA) Matthias Keller (UPB) Anders Lindgren (SICS) Bob Melander (EAB) Susana Perez Sanchez (Tecnalia) Petteri Pöyhönen (NSN) Peter Schefczik (ALUD) Azimeh Sefidcon (EAB) Ove Strandberg (NSN) Asanga Udugama (UHB) Bengt Ahlgren (SICS) Thomas Begin (INRIA) Stephen Farrel (TCD) Claudio Imbrenda(NEC) Dirk Kustcher (NEC) Olivier Mehani (NICTA) Börje Ohlman (EAB) Hareesh Puthalath (EAB) Suksant Sae Lor (HP) Fabian Schneider (NEC) Ayush Sharma (Fhg) Lucian Suciu (FT)
Target Dissemination Level:	PU
Status of the Document:	Final
Version:	1.0

Production Properties:

Reviewers:	Holger Karl (UPB), Hannu Flinck(NSN)

Document History:

Revision	Date	Issued by	Description
1.0	2012-06-11	Benoit Tremblay	Initial revision

Disclaimer:

This document has been produced in the context of the SAIL Project. The research leading to these results has received funding from the European Community's Seventh Framework Programme (FP7/2010–2013) under grant agreement n° 257448.

Abstract:

This document presents the overall prototyping scenarios, use cases and integration points in the SAIL project.

Three prototyping scenarios derived from the project overall use cases defined in D.A.1 [1] are described: *Event with Large Crowd*, *Dynamic Enterprise* and *Elastic NetInf Deployment*. These scenarios integrate the work from the different technical work packages NetInf, OConS and CloNe.

An overview on how the prototype components will be integrated and some means to achieve the integration is also documented.

Keywords:

SAIL, prototype, demonstration scenario, information-centric network, open connectivity services, cloud networking, future internet

Executive Summary

This document is a public deliverable of the Scalable Adaptive Internet Solutions (SAIL) EU-FP7 project [2]. It presents a global view of what SAIL intends to prototype during the project and eventually demonstrate during the course of the project in public (conference) or private events (project reviews, project meetings). It complements other technical documents presented by the different work packages (WPs).

Three demonstration scenarios are presented. The *Event with Large Crowd* scenario focuses around Network of Information (NetInf). It illustrates how mobile users during planned or impromptu large gathering of people can benefit from the NetInf solutions with the support Open Connectivity Services (OConS) mechanisms. It intends to demonstrate the general feasibility of the NetInf approach, the interoperability of the components developed by the different partners during the SAIL project and the specific benefits of the NetInf system for content distribution. The experimentation include use of real devices interacting with the NetInf system as well as emulator to study scalability of the solution and establish comparison with the current internet protocols used for content delivery. The *Event with Large Crowd* scenario is intended to be demonstrated at the Future Internet Assembly (FIA) in Dublin in February 2013.

The *Dynamic Enterprise* scenario focuses around Cloud Networking (CloNe) showing how Virtual Infrastructure (VI) for running applications can be dynamically deployed over multiple infrastructure service provider (data centre or network) domains. In collaboration with Open Connectivity Services (OConS), it also showcases elasticity of the flash network slice. The main testbed is deployed over multiple partner premises and uses different instantiations of cloud Operating System (OS) and network technologies. This allows to validate the applicability of the concepts and constructs used for the establishment of VI in different cloud provider environments. The *Dynamic Enterprise* scenario will be demonstrated at the Future Network & Mobile Summit (FuNeMS) in Berlin in July 2012.

The *Elastic NetInf Deployment* illustrates how NetInf can be deployed as an overlay on top of the Internet taking advantage of CloNe and providing a viable migration path. This scenario provides the required context to validate the CloNe interface from a NetInf management perspective. It illustrates how NetInf nodes can be deployed or removed on-demand from the network depending on the current load for content distribution. No public demonstration is currently planned for the Elastic NetInf Deployment. However, it is planned to be demonstrated at the SAIL project general meeting in January 2013.

For each of these scenarios, we set the objectives and expected benefits of the scenario, describe the demonstration setup and storyboard and give an overview of the different components involved and how they relate to each other. Some indications and milestones on how we plan to achieve these scenarios are also included.

In complement to the demonstration scenarios, we present a quick overview of other prototyping activities that occurred during the SAIL project to validate or experiment with specific concepts or architectural constructs but that have not been integrated in the main scenarios.

Contents

1 Introduction

In research, prototyping has always been used to experiment with new concepts and validate potential solutions to specific problems. In the Scalable Adaptive Internet Solutions (SAIL) project, a large share of the resources are dedicated to prototyping and experimentation of the proposed concepts.

The objective of this document is to describe what we intend to prototype during the project and eventually demonstrate during the course of the project in public (conference) or private events (project reviews, project meetings).

While each work package (WP) describes their prototypes and experimentation in specific deliverables [3–5], this document describes the prototyping and experimentation in a global perspective, showing how all prototyping components from the different partners and WPs fit together in a larger experimental testbed.

We have selected an incremental, bottom-up approach both at partner and WP level. The activities have been organised in two phases. In the first phase, prototyping activities have started from the bottom-up perspective allowing experimentation on specific concepts and architecture constructs. The second phase is characterised by an increased cooperation between the partners and across WPs to form cluster of prototypes and larger scale demonstration. This approach minimised the overall dependencies and risks and allows the project to adapt to theoretical achievement and progress concurrently and quickly. It also allowed to transfer experience gained in the first prototyping phase to the progress of the architectural work that continues until the end of the project.

1.1 Scenarios

The architecture and prototyping work have been focused around a set of scenarios and use cases that are defined in D.A.1 [1]. Two sets of scenarios have been elaborated at the beginning of the project. The first set includes project-wide scenarios that target the project stakeholders and the public. They are used to communicate how the solutions developed in SAIL can be used to answer problems envisioned in the future Internet. The second set of scenarios is used by the project participants to identify and document technical requirements and issues that need to be addressed in the proposed solutions. This second set of scenarios is related to the first set as they describe specific portions of the "non-technical" scenarios, pinpointing specific technical elements.

The prototype integration effort has focused mainly around two of the technical scenarios, the *Event with Large Crowd* and the *Dynamic Enterprise*. These scenarios have been selected because they illustrated well typical problems that are addressed by SAIL. They also require the collaboration from more than one WP, showing the benefit of integrating the proposed solutions.

The first demonstration scenario, *Event with Large Crowd*, is focused around Network of Information (NetInf). It illustrates how mobile users during planned or impromptu large gatherings of people can benefit from the NetInf solutions. It is mainly the result of the efforts from WP-B contributors but also integrates some Open Connectivity Services (OConS) mechanisms developed by WP-C.

The second demonstration scenario, *Dynamic Enterprise*, is focused around Cloud Networking (CloNe) showing how virtual infrastructure for running applications can be dynamically de-

ployed over multiple infrastructure service provider (data centre or network) domains. The demonstration scenario highlights some key technical elements such as VxDL, decomposition of virtual infrastructure over multiple providers, creation of Flash Network Slice (FNS), delegation, resource management, etc. OConS services are also used to illustrate elasticity of the FNS over the wide area network (WAN).

Finally, a third scenario, *Elastic NetInf Deployment*, has been added to showcase possible interactions of the first two scenarios. It illustrates how NetInf can benefit of CloNe. This demonstration scenario is the result of a closer collaboration between WP-B and WP-D contributors.

The main motivations for having multiple demonstration testbeds rather than a fully integrated one are:

- increased visibility of specific innovative features that would otherwise get little attention in too large a testbed;
- streamline the prototyping efforts towards the demonstration of specific elements, rather than spending a lot of effort on testbed aspects which do not provide additional research value or insights;
- reduced complexity and increased potential of successful integration by limiting the number of integration points;
- possibility to target more specific audience (e.g. Cloud and network operators or content providers) that might be interested in only a subset of the overall SAIL solution.

1.2 Structure of the Document

After this section that motivates the prototyping work in SAIL, recapitulates the scenarios and presents the document, the document presents (in Chapter 2, 3 and 4, respectively) the three main demonstration scenarios developed during the project. For each of those demonstrations scenarios, we:

- set the objectives and expected benefits of the scenario,
- describe the demonstration setup
- present the storyboard of the demonstration,
- give an overview of the architecture and integration points between prototyping components,
- describe in more detail each prototype component.

In Chapter 5, we present the results of other specific prototyping activities that occurred during the project but that were not integrated in the final demonstration scenarios.

We finally present in Chapter 6 the plan that have been established for achieving the integration of the prototype components in larger testbeds.

2 Event with Large Crowd

2.1 Objectives and Expected Benefits

When operators dimension deployments of cellular networks, they base the design on regular demands and load on the network during peak hours. There is, however, a limit to how much capacity can be allocated to a single location (in particular for radio communication where the available frequency spectrum is a limiting factor), and operators do not want to spend more money on deployments than is typically required. Nonetheless, there are situations where large crowds gather in a relatively small area during a relatively short period of time (on the order of tens of minutes to hours), creating a very high load on the cellular network. This is typically the case at large sports arenas during matches and other sports events, or during planned or impromptu large gathering of people in urban areas. However the Event with Large Crowd (EwLC) approach is feasible for other situations like high speed train network use, during rock concerts, etc. The main objective of the EwLC is to show improved service to end users during events with large crowds when NetInf functionality is available.

There are different demonstration objectives for this scenario:

Demonstrating general feasibility of the NetInf approach: It will be worthwhile to demonstrate the general feasibility of the NetInf approach, for example for showing that an URI/URL type of communication is adequate to be used in a EwLC scenario. Such a demonstration would require a fairly small network provided that it contains the crucial components, i.e., a user device with an application that is connected to other nodes and the infrastructure through a router collocated at a base station. Additional components, for example a Name Resolution System (NRS) node, can be part of a network setup if infrastructure services are to be shown.

Demonstrating interoperability of different partners' components: We want to show the ability of the different partner components to inter-operate based on the NetInf protocol. Such a demonstration would include multiple different partner components such as mobile node and application components from one partner, NetInf router and name resolution / rendezvous components from other partners. The network would not have to be especially big for that (in terms of number of nodes).

Demonstrating specific benefits of the NetInf system: If we want to demonstrate specific benefits of the NetInf system, for example better scalability (with respect to requests per network and time frame) for content distribution or better robustness (against disconnections in mobile scenarios), we need larger networks that can model specific scenarios in a more realistic way. Such a setup would preferably include well over 100 nodes in a local segment of a larger crowd and be suitable to model base station congestion, packet loss, mobility with changing contacts between peer nodes and an interest distribution that fits the assumptions about the nature of the scenario.

The objective of this demonstration scenario is to highlight the benefits NetInf can provide. It will show how the use of NetInf (instead of existing Internet protocols) can yield a better experience to the end user and at the same time provide benefits to operators and content providers by reducing

the load on network and server infrastructures. In addition, the demo will show how the prototyping activities of different partners can be integrated (using the NetInf protocol) to create a fully working and interoperating NetInf system.

2.2 Demonstration Setup

We will build the EwLC scenario by creating networks that resemble a locally gathered crowd that can exchange named data objects in a scalable fashion. Those objects can be created either locally or in other networks where they are made available through the global NetInf NRS and routing infrastructure (which is not part of the EwLC scenario).

The EwLC scenario occurs during some event that gathers a large crowd, the members of which are interested in accessing data from the network. This large gathering of people creates a demand on the network that is higher than what the network infrastructure is dimensioned for, causing the user experience to deteriorate. As the people in the crowd are in the same area for the same event, it is also reasonable to expect that they will have similar interests that drive their data access patterns (e.g., at a football match, it is likely that a large fraction of the crowd wants to view a replay of a goal). Thus, there are great potentials for using NetInf in this type of scenario as Named Data Objects (NDOs) can be accessed directly from nearby Mobile Nodes (MNs) (see Figure 2.1) that already have retrieved the information. Here the MNs assist each other and data is delivered directly between MNs without use of network resources, thus reducing the load of the network. When no near by MNs have a requested NDO, can the MNs access the NDO from the network, the NDO can be cached close to users in local access points (see Figure 2.2) or available elsewhere in the network (see Figure 2.3). Additionally, user-generated NDO can be distributed either via infrastructure caches or via local peer-to-peer communication techniques to minimize a mobile node's outbound bandwidth consumption .

There are 3 different locations from where the data the user wants to access in this EwLC can be located. The required NDO can reside in either another MN, in the on-path NEC NetInf Router Platform (NNRP) cache or in the off-path RendezVous Server (RVS) cache. The detailed message overview for these different data access categories are depicted in the subsequent figures. Figure 2.1 illustrates the case where the MN1 discovers its neighbor MN and first resolves the content locator (message 1). The MN2 responses with the locator (loc2) back to MN1. After this, MN2 fetches the content (*data1*) using the MN2's locator (loc2) (see messages 3 and 4T). Figure 2.2 depicts the second case where the actually requested *data1* (message 1) is available at an upstream NNRP. The GET message is received at the NNRP and the cached *data1* is delivered with message 2. Here *data1* is delivered directly and no locator reply message needed.

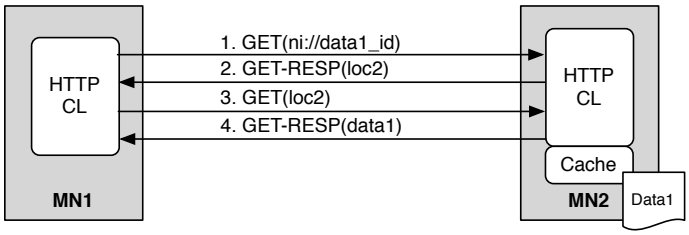


Figure 2.1: Example of how MN1 First Discovers a Content Copy and Then Requests the Content Directly From a MN2

In the third case, see Figure 2.3, a MN tries to retrieve *data1* by quering the network by sending a GET message towards the RVS indicating interest in the *data1* with the *ni://data1_id* reference. The NNRP on the path towards the RVS intercepts the GET message and looks for potentially

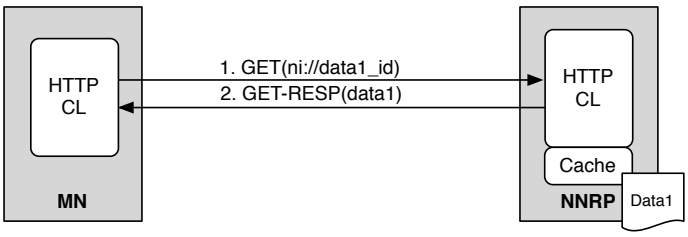


Figure 2.2: Example of how MN's Request is Served Directly From Next-hop Access Router

cached identical data as in the request. However in case there is no copy available in the NNRP, the GET message is forwarded towards the RVS as message 2 in the figure. The RVS replies (message 3) with the actual data and the downpath NNRP caches the *data1* in its local cache (action 4). The final delivery of the data is from the NNRP to the MN with message 5.

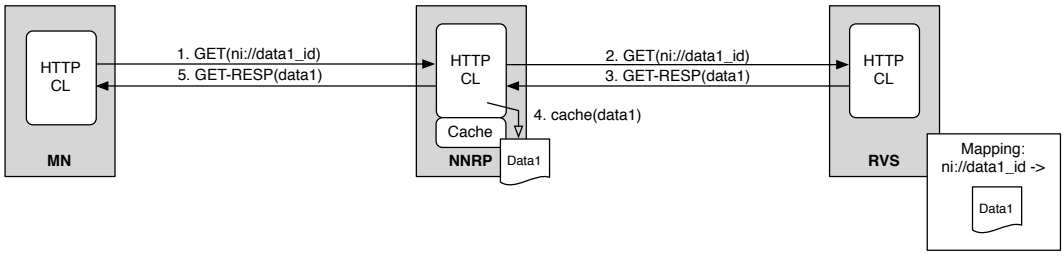


Figure 2.3: Example of how MN's Request is Served From the Infrastructure Cache

When there is a copy available already in the first cache, see Figure 2.4, the GET request gets intercepted and the RVS is not utilized. Here the MN tries to retrieve *data1* by quering the network by sending a GET message towards the RVS indicating interest in the *data1* with the *ni://data1_id* reference. The NNRP on the path towards the RVS intercepts the GET message and finds a copy of the request data in its cache. The data is thus directly delivered from the NNRP to the MN with message 2. Thus the message sequence is the same as depicted in Figure 2.2.

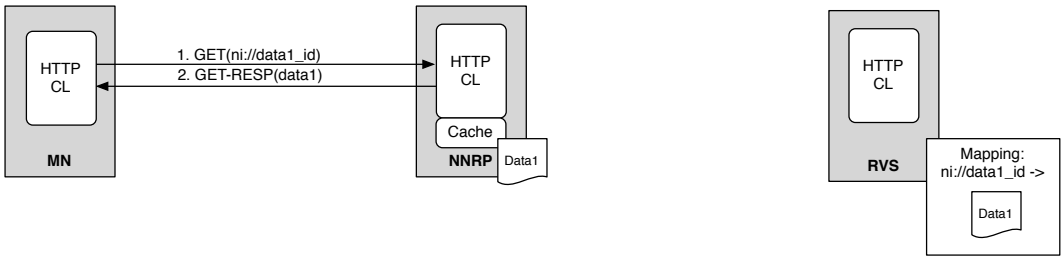


Figure 2.4: Example of how MN's Request is Served From an on-Path Cache

Figure 2.5 depicts a generic, rather simple set-up that could be used to demonstrate the general feasibility and component interoperability. In order to demonstrate the more interesting aspects of NetInf communication, we are setting up an emulation system that allows us to emulate a larger network in real time, with the ability to set-up arbitrary topologies, mobility patterns, load on bottleneck nodes, interest distribution and interest requests sending rates in a controlled and reproducible way. The demo setup presented in Figure 2.5 is realized with the emulation system as shown in Figure 2.6. The emulation system is implemented using NS3[1] and Linux

[1]NS3 is a real-time network simulator tool, capable of simulating diverse network technologies and environmental conditions.

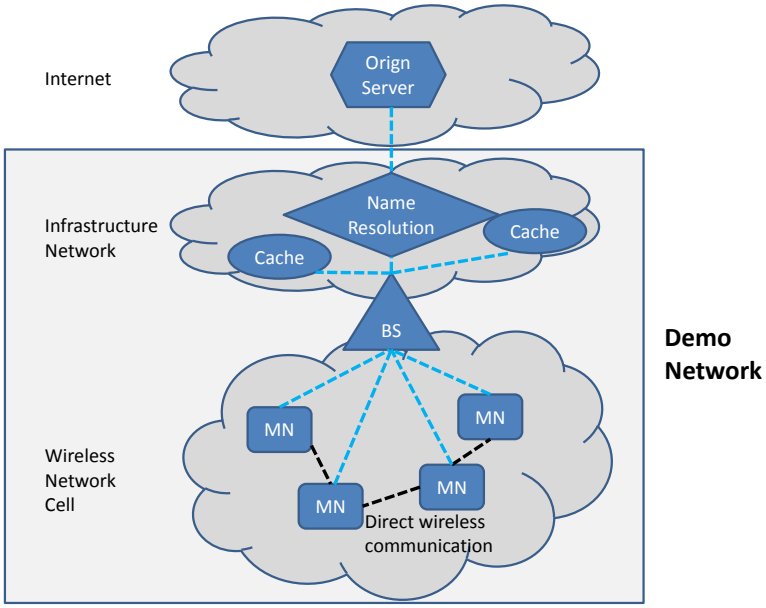


Figure 2.5: EwLC Demo Network Set-up

Container (LXC)[2]. Each LXC container runs a stripped down Ubuntu Linux installation and has 3 virtual network interfaces: one for administration purposes and two for virtualised access points/networks; the latter are used for 3G access and WiFi access (Ad-Hoc mode). NS3 provides real time simulated 3G and WiFi connectivity between the nodes and simulates node mobility; reachability of nodes with the WiFi is therefore not fixed, and varies over time.

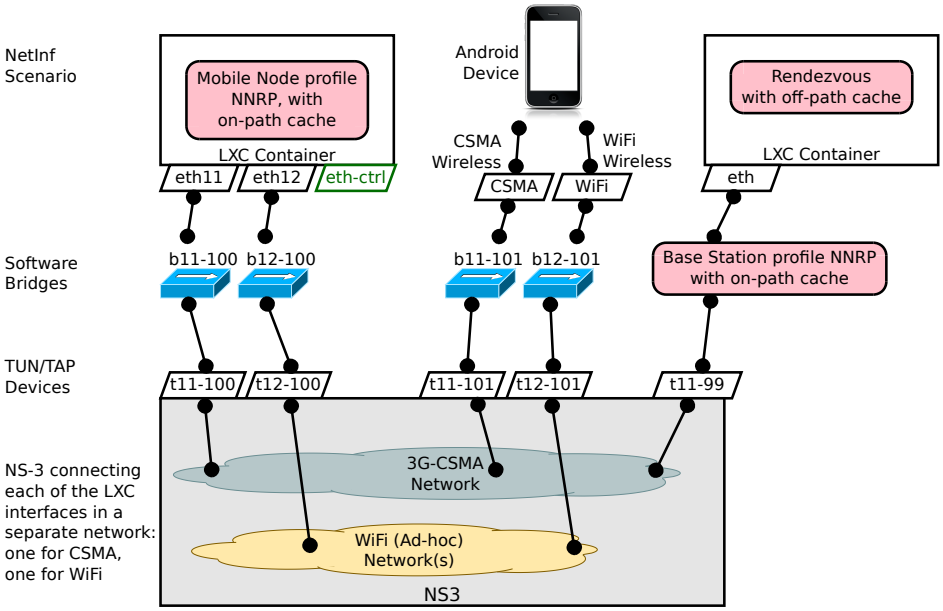


Figure 2.6: EwLC Emulator Set-up

Virtualised nodes run NetInf software natively. Any NetInf implementation that runs on Linux, like NNRP, Rendezvous or others (see Section 2.5), can then be run in the framework out of the

[2]LXC (Linux Containers) is a lightweight operating system-level virtualization method for running multiple isolated Linux systems (containers) on a single control host.

box. It is also possible to attach physical interfaces to the virtual bridge, e.g., for devices or systems which cannot be run easily in a Linux system such as Android devices.

2.3 Demonstration Storyboard

This section outlines the plan for the event with a large crowd demonstration, and describes the different steps that will be executed during the demonstration. The purpose of the demonstration is to highlight the benefits of NetInf in an event with a large crowd scenario, such as the stadium scenario described in [6]. The purpose is also to show how the demonstration objectives described in Section 2.1 are met as follows.

The general feasibility of the NetInf approach will be demonstrated by means of both a real NetInf network and by means of the network emulator described in Section 2.2. The real network consists of a small number of end user devices (Android phones) communicating directly with each other or using an infrastructure network including NetInf routers and a Name Resolution System. The network emulator will be used to demonstrate how the NetInf network can be scaled up with adequate functionality and performance. The code used in the real network will to a large extent also be used in the emulator. The intention is to demonstrate that adequate end user experience can be achieved with NetInf mechanisms while off-loading cellular links.

Interoperability of different partners' components will be demonstrated by the interoperation of the end-user device functions being developed by one partner with NetInf router and Name resolution functions being developed by other partners. This interoperation will be demonstrated both in the real network, and using the network emulator.

Specific benefits of the NetInf system will be demonstrated using the network emulator. The following aspects could potentially be demonstrated:

- end-user experience in terms of response latency
- off-loading of cellular links
- scalability (with respect to requests per node and time frame)
- robustness against intermittent connectivity

Exactly which aspects to demonstrate is for further study.

The emulator demonstration consists of two different profiles: a "legacy" profile and a "NetInf" profile. In both cases the clients (nodes in the crowd) try to fetch objects at random from a predefined list.

In the "legacy" profile, the objects are located on a centralized server and all clients request the objects over the 3G connection. This is a model of the current internet.

In the "NetInf" profile, each client holds one of the objects, and tries to fetch objects from neighbours through the ad hoc WiFi link. If that fails, the object is fetched through the 3G connection. The object is in any case cached, and served to any neighbour who might request it.

The demonstration of the real network will show dissemination of user-generated content based on a selection of the mechanisms shown in Figure 2.1 through Figure 2.4 in Section 2.2. In addition, the demonstration will show how user-generated content is registered with the Name Resolution System and disseminated to caches in infrastructure nodes and in end user devices. The detailed sequence of events in the demonstration has not been decided yet, but the following sequence can be regarded as a draft proposal. First, a user in the large crowd generates content by recording e.g. a video, and registers it with the Name Resolution System. Other users in the large crowd then retrieve the content. A first user sends a request to the NRS to resolve the name of the object into an object locator, and then retrieves the object using this locator. This retrieval results in a

population of caches, both in NetInf routers, and in the device of the first user. A second user then retrieves the object via a cache hit in a NetInf router. A third user retrieves the object via a cache hit in a neighbour device. In the demonstration, end user devices will use WiFi and Bluetooth to communicate device-to-device, and WiFi to communicate with the NetInf infrastructure network via wireless access points.

2.4 Architecture

2.4.1 Overview

Figure 2.7 depicts a sample (simplified) network topology for the EwLC scenario as well as individual components that are required to realize the scenario.

Crowd: The EwLC scenario consists of mobile nodes that are connected to the network infrastructure (via a base station) and that have local communication capabilities. Depending on radio link properties, node position etc., a node can communicate to one or multiple other nodes at the same time. Communication sessions can be short-lived, as nodes can loose contact etc., so that, in general, the crowd provides communication opportunities, but with unpredictable performance. Each node is also connected to the infrastructure, but we assume that, for large crowds, the link to the corresponding base station is overloaded, i.e., heavily congested and not always usable.

Infrastructure: The infrastructure provides access to the Internet and to operator-provided caching and name resolution resources. We assume that operators provide caches and other NetInf components in their access and core network. One of the components is a rendezvous server (RVZ) that manages a network of caches in an operator network.

Internet: The Internet provides additional NetInf nodes, especially sources for named data objects that are requested in the Crowd network.

Figure 2.8 zooms into a mobile node in the EwLC scenario and depicts how OConS service for Connectivity Management and Multipath communication can be leveraged:

Connectivity Management: A NetInf mobile node in the EwLC has several network attachments and communication opportunities over time. We assume that a node has at least two network interfaces – in general one for infrastructure connectivity (via the base station) and one for local communication. A NetInf protocol stack on such a node has to find out about the available interfaces, configure them so that they can be used in a specific setting. Once an interface is operational, it still needs to be managed. For example, for infrastructure connectivity, suitable base stations have to be selected based on cost, available capacity etc. and associations to base stations have to be maintained. For local radio interfaces, we assume that nodes will experience opportunistic connectivity and be able to communicate to zero, one or multiple nodes at a time. This opportunistic connectivity needs to be managed, i.e., contacts have to be detected and signaled accordingly so that NetInf Convergence Layer links can be set up.

Disruption Tolerance: In scenarios such as the *event with large crowd* scenario connectivity between nodes may be short-lived and intermittent. For the local communication interface, nodes cannot rely on being able to communicate with all other nodes in the crowd. Instead, connectivity will depend on occasional contacts to a subset of the nodes. In order to decide which nodes to query for named data objects and which nodes to publish objects to, NetInf

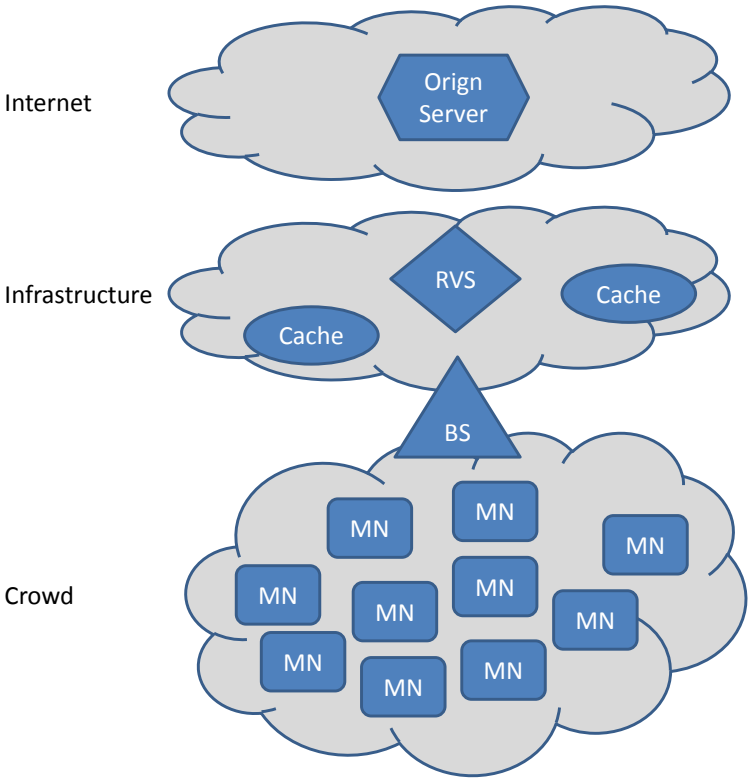


Figure 2.7: Functions in the EwLC Scenario

can inter-work with underlying DTN routing services, which will be described in more detail in Section 2.4.2.1.

Multipath Communication: The different connectivity options enable path diversity that can be leveraged by nodes to use the network's capacity more efficiently. For example, in the hop-by-hop NetInf message forwarding model, requests can be duplicated, aggregated, filtered at intermediate nodes, depending on their corresponding forwarding strategy. A single node with multiple interfaces thus has to decide how a request should be forwarded best. If the requests are duplicated over the available interfaces but will be aggregated again some hops away, bandwidth and processing resources would be wasted. A multipath management function should assist the NetInf protocol layer in performing those decisions (see Section 2.4.2.2 for details).

In the following, we describe how connectivity and multipath communication management functions as Open Connectivity Services are planned to be integrated into the NetInf EwLC scenario.

2.4.2 Integration with OConS

A number of mechanisms of OConS together with the OConS framework improves the performance of NetInf in retrieving content in the context of the flash crowd (EwLC) scenario. One of these mechanisms is the support for DTN routing based on social routines, while the other relates to multi-path content retrieval with NetInf. These components are described in the following sections.

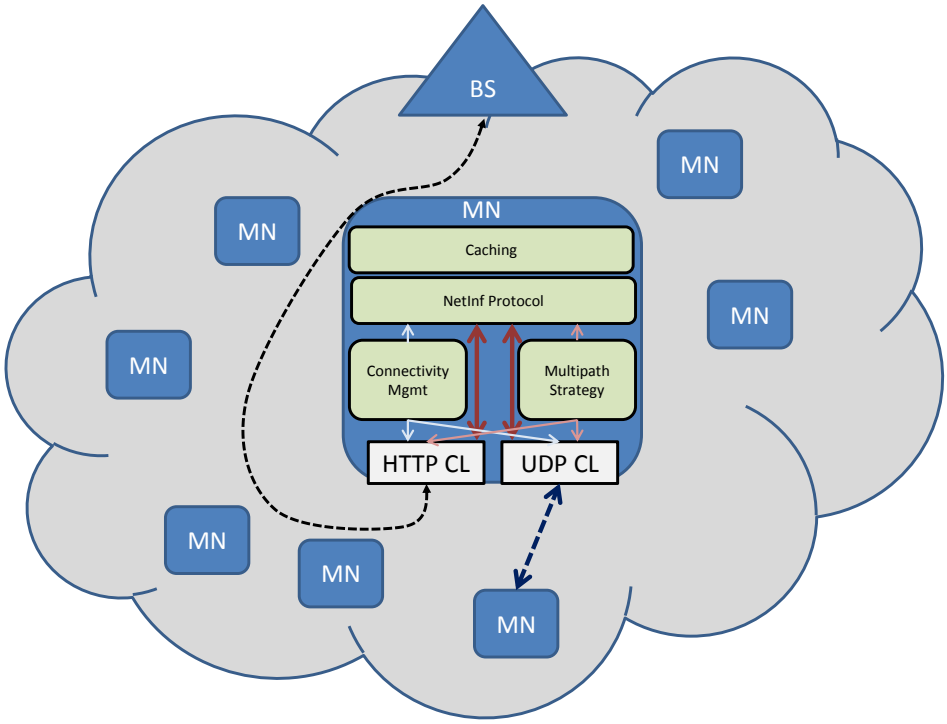


Figure 2.8: Multipath and Connectivity Management in the EwLC Scenario

2.4.2.1 DTN Routing Based on Social Routines to Support NetInf

Not every single node in the EwLC scenario is permanently connected to every other node in the group (there are no end-to-end paths permanently established), and so, a Disruption/Delay Tolerant Network (DTN) environment can be formed and exploited for sharing the content demanded. If some people start moving away from the crowd, they could still spread information to new neighbours, not physically present in the flash event. We propose a novel protocol, (HUman Routines used for Routing (HURRy)), which incorporates the contact duration to the information retrieved from historical encounters among neighbours, so that more meaningful routing decisions can be made, compared to previous approaches (e.g. Probabilistic Routing Protocol based on Historical EncounTers (PRoPHET)). [7] describes how PRoPHET bases the estimation of contact probabilities on the inter-contact times, deriving a proportional increment or decrement with each new encounter registered. This approach has proven to be efficient for certain generic scenarios, but might result insufficient for some human-based topologies, where people's social behaviour influences the network deployment. Some experiments in this direction (like [8]) highlighted the importance of, for instance, distinguishing between short and long contacts and deriving more elaborated mathematical relations in order to optimally prioritize the available routes to a destination. Accordingly, HURRy introduces a novel mathematical equation that evaluates the quality of each neighbour and estimates contact probabilities according to this rating system. Some simulation results, as compared with the performance of PRoPHET for scenarios where the contact duration shows high variability are summarised in [9].

A simple idea to demonstrate how OConS can support NetInf in a flash crowd consists of a DTN node (A) triggering a request to retrieve a video, previously recorded by another DTN node (B) which is not present in the crowd anymore. Let us assume that some of the people participating in the EwLC scenario previously know each other, and even share some connections over any of the

common social networks on the Internet. They normally use their mobile phones while interacting with their social profiles, which link them together with third groups of friends and the like. If that information history about previous social interactions can be exploited by a DTN routing mechanism during the EwLC, node A may be able to get a video file recorded by the non-present node B, due to having some friends in common (even if they were not aware of their short degree of separation). The short range connections established in the EwLC can be considered together with previous social behaviour of present nodes, and incorporated to the DTN routing. Considering the contact routine of the nodes involved can support the application service by selecting the best route to a content file, since the mechanism itself incorporates some knowledge about network nodes, which is not actually available from the local topology awareness.

2.4.2.2 Multi-path Content Delivery for NetInf

The participants of the flash crowd perform downloads of content based on spontaneous decisions that are related to the interests of these participants. These participants use NetInf-based computing devices and these devices are capable of maintaining simultaneous multiple attachments to the networks. The OConS mechanisms together with its framework enable these NetInf nodes to retrieve content over the multiple attachments in the best possible manner (Figure 2.9).

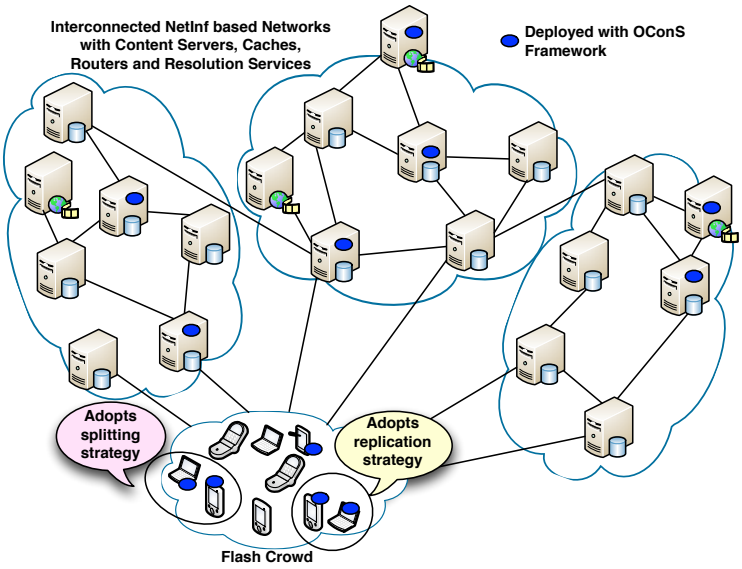


Figure 2.9: Multi-path Content Delivery in Flash Crowd

NetInf has the built-in capability to perform retrieval of content over multiple attachments or paths. The OConS mechanism together with the components of its framework steers this built-in capability of NetInf in retrieving content. The Decision Making Entity (DE) of the OConS framework located in NetInf based user devices makes decisions related to selecting the best multi-path strategy to use. There are 3 types of strategies.

- Splitting - This strategy relates to distributing the NetInf GET messages between the multiple paths
- Replication - This strategy replicates the GET messages to the multiple paths
- Distribution - This strategy distributes the GET messages of different content requests to the multiple paths

The identification of the multiple paths and the strategy to be adopted by the DEs are based on the information provided by the Information Management Entity (IE) of OConS. The IEs located in the network as well as the user devices, have a number of information collection modules such as the module to collect information about currently active network attachments. The DE use this information to evaluate a set of rules held in the DE to obtain a set of actions. These actions specify the strategy to adopt. Once the decisions on the strategy and the multiple paths to use are made, this information is given to the Execution and Enforcement Entity (EE) in the NetInf based device to implement the content retrieval strategy. This process is a continuous process where decisions are re-evaluated based on new information and re-implemented.

The sequence of actions related to the scenario involves a number of steps starting from the request of the content to the final receipt of the content on the NetInf based devices.

- The content retrieval application requests for the content by providing NDO names;
- These names are resolved by the NRS to one or more locators;
- This location information together with the other information provided by the IEs are used by the DEs to select the multi path strategy to use to retrieve the content

Each of the networks that a flash crowd participant's device connects to has different characteristics. These will determine what multi-path strategy is selected. For example, one such characteristic is the congestion in the attached networks. When different networks have different congestion levels due to activities of the flash crowd participants, the OConS mechanisms select a content splitting strategy and will fine-tune the strategy based on the changing congestion information.

2.5 Prototype Components

This section describes the different prototype components involved in the scenario.

2.5.1 Rendezvous Component

The rendezvous function is the control point of the publish/subscribe-based ICN architecture used in the prototype scenario. In essence, the function is responsible for matching interests of a content provider (publisher) and a content consumer (subscriber). In the scenario, when mobile nodes are not able to find content in their vicinity, they do need infrastructure services to find and locate the content and this is rendezvous' main role. Rendezvous clients are both publishers and subscribers. Publishers use the NetInf protocol to publish their content in the rendezvous system and subscribers query or fetch the available content using the NetInf protocol as well. The function itself is implemented using Java and HTTP server functionality is based on Apache HTTP components (http://hc.apache.org/). In the prototype scenario, the rendezvous is located at the infrastructure side and it can operate in two different modes: i) query mode and ii) request routing mode. In the former case, the rendezvous instance receiving a query serves it without contacting another rendezvous instance. In the latter case, rendezvous instances automatically find the instance responsible for the requested content. Depending on the configuration, there might be one or many rendezvous instances in the infrastructure nodes.

2.5.2 Netinf Enabled Android Phones

Ericsson's contribution to the Event with Large crowd prototype is a Mobile NetInf terminal with an Android implementation of NetInf node functionality.

The mobile terminal has the following functionality:

- Retrieve Netinf objects from other Netinf enabled devices using Bluetooth, WiFi or 3G connection.
- Cache Netinf Objects.
- Name resolution of locally stored objects. This is used both when local applications request objects and when remote nodes send get requests to the node.
- Serve locally cached Netinf objects on request from other NetInf-enabled devices.

The functionalities described above use interfaces as developed and described in the SAIL project.

The mobile terminal can interact with other mobile terminals without using any other network infrastructure components but can also interact with NSN RVS and NEC NNRP (Figure 2.10).

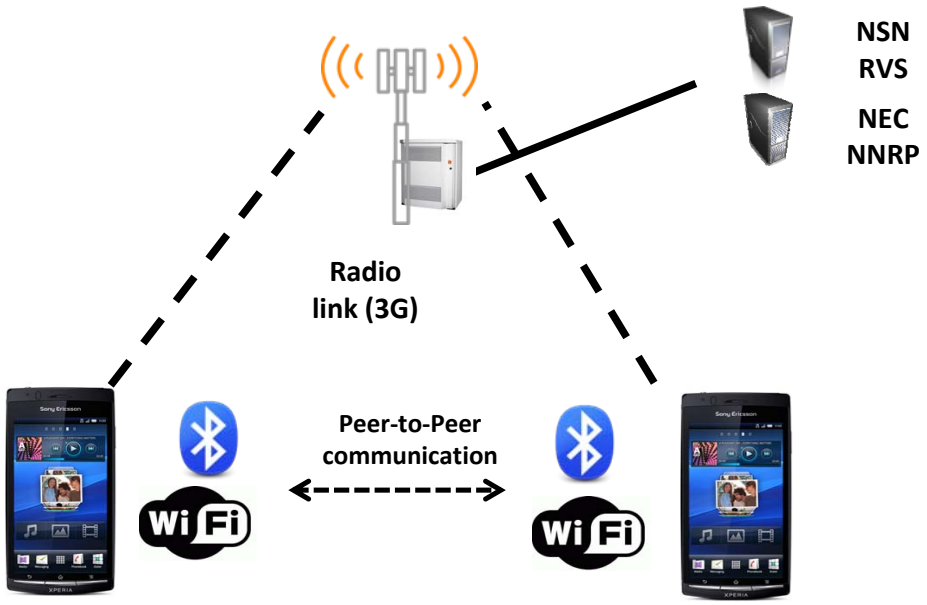


Figure 2.10: Mobile Nodes Interaction

The implementation is done in Java. Apart from code developed by Ericsson some code from the open source project open NetInf created by University of Paderborn has also been used.

The functionality described above is implemented and to a large extent also tested and working.

2.5.3 NEC NetInf Router Platform (NNRP)

The NNRP prototype aims to implement the full SAIL NetInf specification. NNRP is a modular and extensible infrastructure for easy development of ICN networks and testbeds. NNRP is implemented in standard C, for POSIX-compatible operating systems (mostly tested on Linux). Many modules are implemented and a basic ICN functionality is already in place. The goal for the end of summer 2012 is to be able to fully implement the SAIL NetInf specification, including Convergence Layers, naming scheme, routing, and discovery of neighbours.

The planned scenario for a demo is an implementation of the large crowd scenario, as per D.B.1 [10].

The architecture of NNRP, like many other routing platforms, is centred on chains of modules which operate on the passing messages (Figure 2.11). Chains and chain configurations are defined entirely in the configuration file.

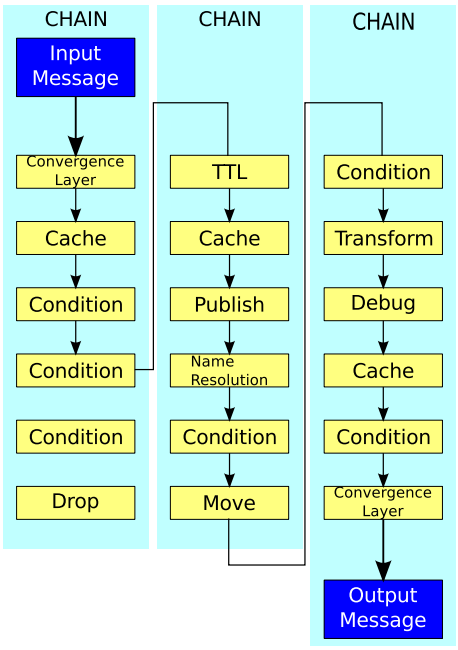


Figure 2.11: Detailed View of NNRP

Incoming messages are injected in chains by the appropriate convergence layer modules, filtered by the modules, moved around through different chains and eventually forwarded to other nodes. This great flexibility allows NNRP to be used on any node in the demo setup.

2.5.4 Multi-Path with OConS for NetInf

The NetInf architecture considers the use of multiple paths to retrieve content natively. Though NetInf supports the use of multiple paths simultaneously, no formal mechanisms have been defined to utilize them in the best possible manner. The prototype developed here demonstrates the use of a number of multi-path strategies for NetInf through the functionality of the OConS framework.

The architecture of the prototype consist of a number of components that interact with each other to select and implement the multi-path strategies. These components extend the operation of the NetInf architecture to perform the retrieval of content of multiple paths.

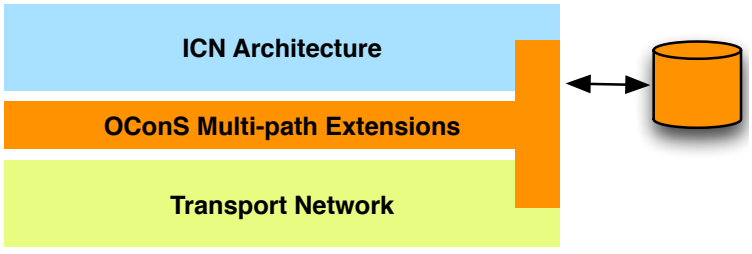


Figure 2.12: Protocol Stack (Generic Architecture)

Figure 2.12 shows how the extensions are positioned in terms of the layers of the protocol stacks of any Information Centric Networking (ICN) architecture. These extensions, operating as the Forwarding module and the Convergence Layer (CL)s in the case of NetInf and as a shim layer with other ICN architectures perform the setting up of the forwarding strategy to retrieve content over multiple paths.

The initial prototype uses a simple ICN implementation that is extended to include the OConS multi-path extensions. The OConS functionality in this prototype is located in the user device where the DE functionality selects the best multi-path strategy to use based on the information provided by the IE and enforced by the EE located in the user device. Figure 2.13 shows how the test-bed is setup with this initial prototype to perform multi-path content retrieval.

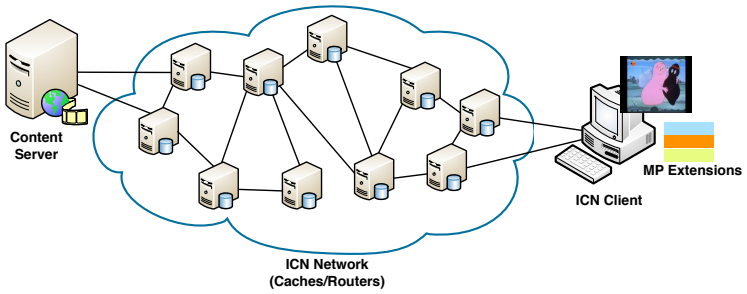


Figure 2.13: Realization of Extensions

The user device of the prototype uses the splitting strategy to retrieve content. The IE provides information related to current attachments to the networks and the content request expiration information, for the DE to fine tune how the splitting strategy is operated. The splitting strategy uses a Additive Increase/Multiplicative Decrease (AIMD) mechanism to distribute the content requests to the multiple paths. A video streaming application is used to demonstrate the operation of the splitting strategy.

2.5.5 DTN Routing Based on Human Routines with OConS

Figure 2.14 shows the targeted scenario for the demonstration of this prototype. The realization of the described mechanism is aimed to be released for Android phones, but could also run on laptops. The OConS implementation of a DTN routing mechanism is based on the social interactions among

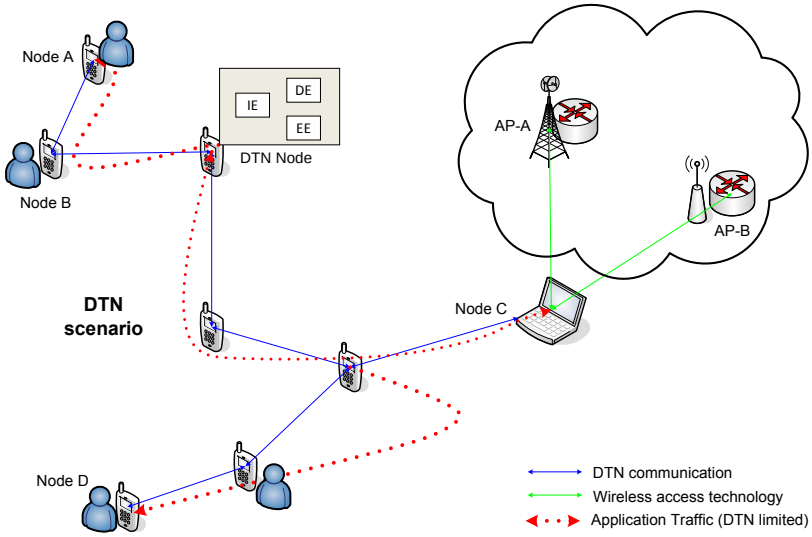


Figure 2.14: Generic Architecture for Components in a DTN Scenario

nodes. It is intended for human-carried devices (e.g. mobile phones interacting as DTN nodes), so that the dynamics and mobility of those DTN nodes can be seen as people's social behaviour that the routing protocol uses.

[11] describes the specification of the Bundle Protocol Query (BPQ) extension block, which allows applications to query the stores of nodes on the path along which a bundle containing a bundle query extension block is routed. The BPQ extension is intended to allow such queries that can be answered by intermediate nodes, where those nodes do not necessarily have to be addressed by the destination of a corresponding request message. The final purpose of integrating the HURRy protocol with the NetInf implementation of BPQ is to show the cooperation between previous knowledge of social encounters among nodes and the concept of intermediate caching of demanded content.

The combination of both prototypes will allow DTN nodes to act as NetInf caches and to process BPQ requests and responses for the content shared.

The integration plan is sketched into several phases, including the following:

- Individual prototype implementation: NetInf has developed a prototype with laptops showing how the BPQ extension is used to retrieve video content from a DTN node which has not access to the video server (using an intermediate cache). OConS is finishing its own implementation of the DTN routing (HURRy) for Android phones (hybrid deployment of phones and laptops);

- Adaptation tasks for integration: DTN routing is implemented in Java code, whereas BPQ extension is available in C++, so some adaptation will be needed to integrate both prototypes. One possibility is to have a hybrid DTN scenario with laptops acting as NetInf nodes (content caching is available) and Android phones acting as OConS nodes (routing based on social metrics): a NetInf node should act as the user who requests a certain video file (sending a BPQ request); OConS intermediate nodes would forward this request using social metrics to select the best route to destination; and another NetInf node would act as the cache that processes the BPQ response and serves the demanded video;

- Full integration: the final goal would be to port the BPQ extension into Java code so that all DTN nodes are running a complete (and fully integrated) implementation of both mechanisms. A hybrid scenario could be deployed with phones and laptops, both acting as NetInf/OConS nodes (i.e. all nodes are able to generate BPQ requests/responses and they use social metrics to route/forward packets to neighbours).

The realization details of the HURRy protocol implementation are as follows.

- Operating System: Android 2.2 or above

- HURRy protocol implementation: Social routing (PRoPHET based) implementation, developed in Java, taking the Bytewalla3 [12] project release as a basis

- BPQ Extension: BPQ functionality developed in Java and integrated into DTN2 implementation of the Bytewalla3 project

- DTVideo service: simplified video service

We are currently developing the implementation of the social routing in Android phones, the status of which is considerably advanced. A more detailed implementation and integration plan is presented in D.C.3 [4].

3 Dynamic Enterprise

In today's cloud environment, an enterprise or an application provider can make use of computing and storage resources that are dynamically allocated on a needed basis. However, network considerations such as connectivity between resources and specific requirements on the communication links, which are critical for some applications, are currently missing. CloNe enhances the cloud computing with networking aspects by allowing to establish connectivity between cloud resources allocated in different private or public data centres in the same dynamic and elastic fashion that today's cloud can allocate computing and storage resources. CloNe also address network resource optimisation and potential latency issues by distributing cloud computing and storage resources in the network, closer to the end users.

3.1 Objectives and Expected Benefits

The objective of the dynamic enterprise prototype is to illustrate the establishment of an elastic Flash Network Slice (FNS), connecting cloud computing and storage resources from multiple data centres over the WAN at the same time as those computing and storage resources are allocated in the data centres.

This prototype will demonstrate key concepts and their realization, allowing an enterprise to use heterogeneous and geographically distributed compute clouds to provide a web-based IT service to its customers. This includes:

- Ability for tenants to express complex systems and network infrastructures.
- Ability to create tiered architectures with the possibility to scale independently in the different tiers and apply firewall properties between tiers or among nodes (who can talk to whom).
- Topology of the infrastructure and capabilities of the network (speed, broadcast capabilities, security, isolation) can be specified and are acted upon.
- Create FNS that crosses administrative boundaries involving DC/LAN networks and WAN VPN networks with dynamic characteristics.
- Ensure isolation across administrative boundaries and mechanisms for data exchanges to allow the setup and management of the FNS.
- Ability to swiftly add and remove remote sites and entities within the enterprise boundaries.

Moreover, the prototyping activities will allow us to validate and refine the architectural concepts and the protocol specifications. For example, the prototype effort for this use case has allowed us to validate the appropriateness of the RESTful FNS APIs exposed to the users and to refine the mechanisms and protocols used in the link negotiations between different Data centre (DC) and Network Operator (NO) domains. Those issues and discoveries have been and will continue to be brought into the architecture discussions.

A second part of the demonstration aims at showing elastic networking (controlled by OConS) in cooperation with cloud management (controlled by CloNe). OConS monitors and controls flows using a distributed control plane (between the domains) and OpenFlow as protocol and forwarding

engine. The aim is a proof-of-concept demonstrator that allows to experiment with load-dependent resource allocation algorithms, validate the centralized control architecture, test the protocol variants between CloNe and OConS and evaluate the performance of different decision strategies in path selection across the domains.

3.2 Demonstration Setup

For the dynamic enterprise use case, the testbed has been divided in two independent parts. The first and larger part, supported by WP-D partners, integrates all components required to create and deploy a virtual infrastructure over multiple domains (data centres and network operators). The second part, supported by WP-C partners, is used to experiment with elastic networking inside the NO domain and to investigate the impact on management of connectivity over domain boundaries. The two parts of the testbed may eventually be integrated depending on the progress on the interfaces for managing network elasticity over the domain boundaries and the resources left in the project for realising such an integration. Experimentation results and findings will in any case be shared to clarify the interface between OConS and CloNe in the *Final Harmonised SAIL Architecture* deliverable (D.A.3 to be published [13]).

3.2.1 Cloud Networking – Virtual Infrastructure

An important requirement for SAIL has been to propose solutions amenable not only to future technologies but also applicable in currently deployed infrastructures. This is reflected in the testbed used for prototyping, which includes two MPLS networks and separate OpenFlow networks. An ambition, also reflected in the testbed, has been to cover heterogeneous cloud platforms as the market reality is that a variety of commercial and free alternatives have established themselves. They are not likely to disappear any time soon. Figure 3.1 shows an overview of the testbed.

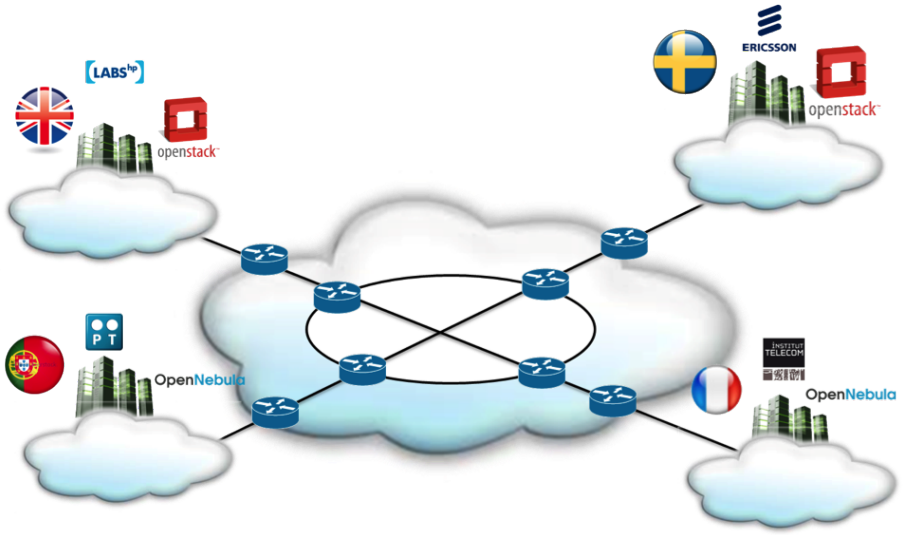


Figure 3.1: WP-D testbed with OpenStack and OpenNebula Clouds and EAB's MPLS Network

The testbed spans five different infrastructure domains; four geographically distributed cloud data centres and the aforementioned MPLS wide area networks. The cloud data centres located in Aveiro (Portugal, PTIN) and Paris (France, IT) both run the latest version of OpenNebula; the data centre in Stockholm (Sweden, EAB) uses OpenStack, as does the one in Bristol (United

Kingdom, HP). Apart from the cloud platform-specific APIs, all four data centres also implement a common service interface, the OGFs publicly available Open Cloud Computing Interface (OCCI).

One of the MPLS networks is software-based and runs as virtual machines on infrastructure located in Sweden (EAB). Both MPLS networks implement the Open Cloud Networking Interface (OCNI) as a service interface that is available through public IP addresses. This is an interface defined by Institute Telecom within SAIL. The network services currently integrated in the prototype are layer 3 Virtual Private Networks (VPNs). Since the testbed includes two MPLSs network, it allows experimentation with FNSes spanning multiple network providers. All cloud data centres all connected to both MPLS networks.

The PT lab infrastructure integrated in the testbed is shown in Figure3.2. It is composed of three domains, two data centre domains and one NO domain. Despite not appearing on the figure for simplicity, all three domains are connected with the EAB NO domain. The two data center domains are identical: they are both composed of two physical machines (one acts as frontend and both can host virtual machines) and run under the management of the OpenNebula Cloud Platform (one instance of the Cloud platform per data centre domain). As an active open source project, OpenNebula releases new versions within short time periods and the versions running within the data centres keep up with this evolution (currently upgrading to version 3.4.1).

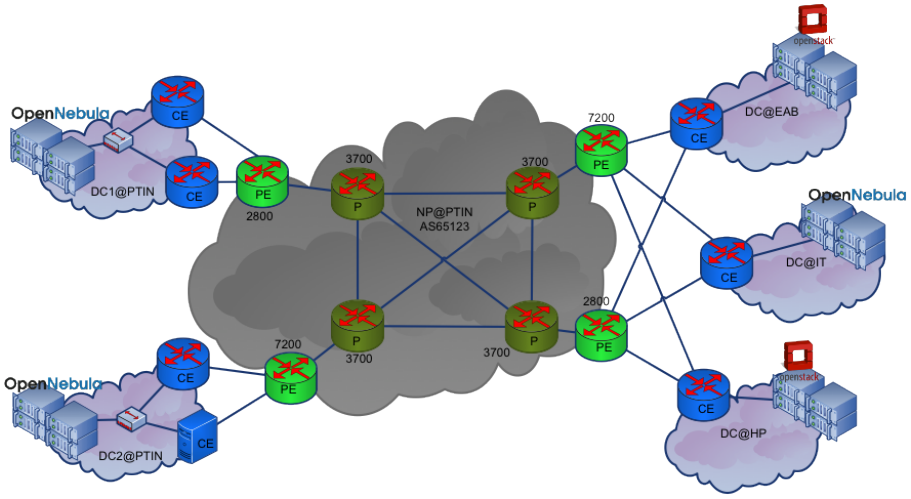


Figure 3.2: PT's Part of the Testbed

The PT data centres networks are flat, having two routing devices (each) that connect to the testbeds WAN network. In order to create isolated (network) environments, the 802.1Q protocol (VLAN tagging) is used, one of OpenNebulas supported techniques. Moreover, OCCI (natively embedded in OpenNebula according to the latest draft of the OGF OCCI API specification) is the external, public-facing interface available in both data centres. The NO domain is composed of physical Cisco routers: 4 core routers (i.e. cisco 3700) connected in a full mesh and 4 edge routers (Cisco 2800 and 7200) that are used to connect to the different DCs. The NO is running an MPLS network and is also running BGP (AS 65123) and OSPF networking protocols. The NO is able to dynamically provision dedicated links (i.e. FNS) between all DCs through the setup of VPN layer 3 services (or even VPN layer 2 services) on the physical network. This feature is available through the external public interface OCNI (pyOCNI).

Since the domains in the testbed are distributed across Europe and no dedicated network connectivity could be arranged for the project (due to cost and complexity reasons and for the simple fact that it does not add any valuable enhancement to the state of the art research in this area) the Internet is used to interconnect the infrastructure domains. To emulate point-to-point connec-

tions between domain edge equipment (like PE routers), IPSec and GRE tunnels are used. Since the experimentation focus is more on control plane aspects rather than data plane performance, reliance on Internet as underlying connectivity provider is not a serious constraint.

Finally, we needed an example application to deploy over the testbed. Enterprise applications, e.g. SAP R/3, can be very costly and are complex to setup and configure without the right competence. We have therefore chosen the open source application Opencart for the prototyping effort in the Dynamic Enterprise use case. Opencart is used to build e-commerce web sites with look and feel like Amazon, eBay etc. As such it provides a good compromise by being a real-world, production-grade enterprise application with a two-tier architecture (web front-end and database back-end) while still being quite simple to deploy. Figure 3.3 shows a screen capture of the OpenCart web shop used in the prototype.

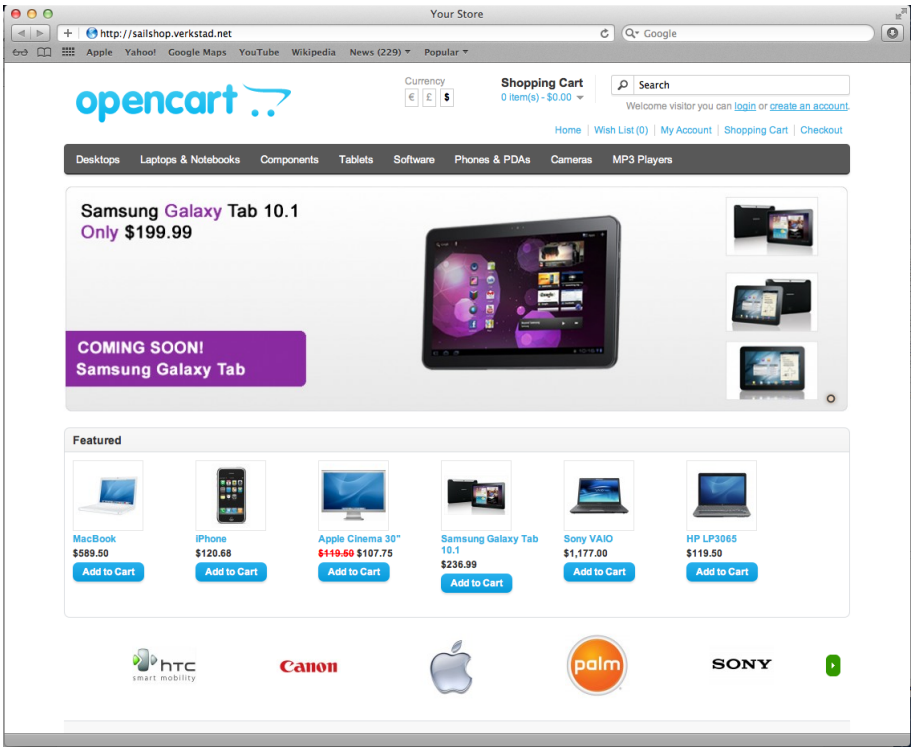


Figure 3.3: Opencart Web Shop Application Used in the *Dynamic Enterprise* Prototype

3.2.2 Elastic Networking with OConS

The set-up for the OConS Elastic Networking case includes application data centres as well as network operator data centres (e.g. for mobile access networks) in order to demonstrate their collaboration in case of elastic network resource allocation.

We assume that mobile networks consisting of a packet core network and the Radio Access Network (RAN) will be virtualized and "cloudified" to a large extent. In the RAN case this means that the dedicated processing capacity, e.g. for baseband processing, user processing and cell-related processing, will also be hosted in a cloud-oriented data centre, while at the cell site only the radio equipment is deployed. The challenges in such an architecture are the tight real-time conditions between the processing units in the cloud and the connecting network.

In order to experiment with such cloud characteristics and aspects, we had to set up a different demonstration testbed where we selected a distributed video application as an example for a networked application with corresponding demands and constraints.

Our demonstration set-up will model OConS concepts for connecting data centres (with specialized processing) and managing the load and connectivity between them. We then run a specific case with many users that simulates a flash crowd and, thus, challenges both the networking capacities as well as the available processing resources.

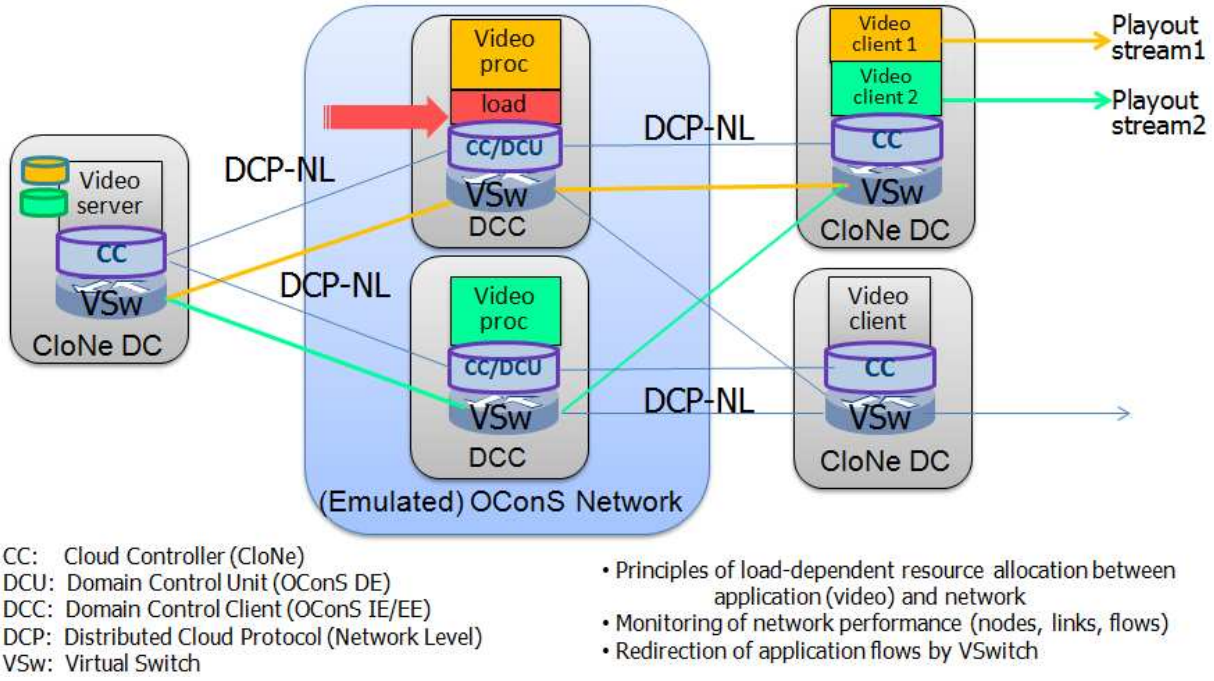


Figure 3.4: OConS-Clone Demo Setup: Elastic Networking with Video Application

Let us assume the following: a number of end users want to watch a video on their mobile phone, which in addition needs a personalized processing and adaptation in a video processing node in the (mobile access) cloud. Thus, an appropriate amount of processing units has to be reserved inside a mobile cloud data centre for this large amount of users and appropriate paths need to be set up to the video source(s) of the content. The utilisation of the networking and processing resources along these paths will be monitored by OConS. Once an overload situation (in the processing path) is detected, OConS either will initiate a redirection of the assigned paths (to inter-connect the data centres hosting the cloud application) or it will set up an additional path over less loaded processing nodes (data centres) that are served by this OConS domain. All this is set up and managed by the cooperating and distributed Domain Control Units (DCUs) servers, as OConS entities that control the resources assigned to the respective mobile cloud data centre resources. The CloNe Cloud Controller (CC) is the corresponding entity in the CloNe architecture that interacts with OConS DCU.

The OConS mechanisms used in this demo set-up and offering the necessary services are as follows:

- The overall reference model and control architecture for the OConS Data centre use case was introduced in D.C.1 [14] and D.C.1-Addendum [15] , where the OConS entity DCU was introduced as the responsible control entity of a single OConS domain.

- Decision (DE) and information collection (IE) entities are realized in the DCUs placed within the data centres and the connecting OConS network domains.

- All three OConS entities (DE, IE, and EE) are used inside the clients Domain Control Clients (DCCs) of the respective domains.
- The DCU interacts closely with the connected DCCs to collect intra OConS domain information.
- The DCU also interacts with adjacent OConS domains DCUs or CloNe CCs to exchange inter-domain information, e.g. on processing resources available remotely or link load between such processing resources.
- Several performance attributes such as resources currently offered or network quality of service (Quality of Service (QoS)), energy consumption or provisioning price can be used by the resource allocation algorithm.
- When the appropriate information is collected by the CCs and neighbour DCUs and a new processing task has to be carried out, the calculation and acquisition of appropriate processing resources and available network resources is performed by the resource allocation algorithm.
- When the network load changes, the DCU reacts accordingly thereby offloading processing in overload or flash scenarios and providing a better data-path efficiency, service, QoS or other target the operator has specified. This ensures an uninterrupted service for the users and also an optimal usage of the operator's network resources.

3.3 Demonstration Storyboard

The enterprise in this use case is a retailer of electronic appliances that wants to offer its products through a web shop. The web shop is to be integrated with the database used for its traditional shops. That database runs on the enterprise's own IT infrastructure in Sweden. The web front ends are to be provided from cloud data centres located in the targeted markets, in this case UK, Portugal and France.

The steps of the use case are as follows:

1. Using a cloud management tool, CloudWeaver (see further Section 3.5.1), the developers of the web shop service describe the system to be deployed. It involves specifying the relationship between the web front-ends and the database back-end and the type of connectivity they need. The developer provides constraints as to where (desired geographic regions) the components should be deployed and the properties of the connectivity (like bandwidth needs).
2. The decomposer component (in our prototype implemented inside CloudWeaver) determines where the different components should be deployed. It then generates appropriate OCCI and OCNI messages for the chosen domain.
3. The OCCI and OCNI messages are sent to the API service points exposed by the domains.
4. Upon reception of the request, each domain processes them. This will trigger interactions between the cloud data centres and the NO for the establishment of the required FNSes.
5. The Link Negotiation Protocol (LNP) message exchanges enable the data centre clouds and the NO to negotiate the necessary parameters needed to establish the FNS. This includes which physical link to use, the VLAN or other service identifier for the tenant (the enterprise), and the routing protocol to use (and associated parameters).

6. Once the LNP interactions have completed and all virtual resources have been instantiated (e.g., the needed VMs have been spun up), the web shop application has been dispatched and is accessible over the internet.

The web shop front-ends have an interface facing the public internet, which is used by the web shop customers to access the web shop. They also have an interface coupled to the FNSes, which they use to interact with the back-end database.

3.3.1 OConS Support for Cloud Networking: Elastic Networking

In the Elastic Networking part of the demonstration, the focus is mainly on the NO. A video processing application is used to demonstrate the dynamic and elastic networking as a distributed cloud feature. We show that tight time constraints on FNS reconfiguration can be met and that processing units can be distributed remotely in the cloud and be connected in a flexible way.

Video streamers running a remote cloud send video streams over an NO domain where video processing nodes are deployed using the CloNe mechanism (OCCI, OCNI). The video streams are transformed by those video processors (watermarking, ad insertion, ...) and forwarded to clients connected to different data centres (Figure 3.4).

Starting from a request to establish an FNS through the OCCI and OCNI interfaces, the domain controller sets up computing resources (video processors) and flows within its own network domain (intra-domain) and connect to the other domains using the LNP. This can be seen as part of the OConS service orchestration process. The resources used to establish the FNS are selected by the DCU in conjunction with the Traffic Engineering Database (TED) and Path Computation Entity (PCE). By changing the configuration of the controllers, different routing algorithms can be 'orchestrated' and activated on the fly such as 'shortest path', or 'constraint-based routing'.

The domain controller monitors the availability of the connecting links, the performance (QoS) of active flows and the load of the video processors and can rearrange flows in case of degradation of service or network congestion. This is illustrated by requesting new video from the clients and observing the reconfiguration of the resources in the network operator domain.

3.4 Prototype Architecture Overview

In this section, we give an overview of the architecture used in the demonstration prototype. It does not reflect the final architecture for neither OConS nor CloNe. Also, work is currently ongoing to harmonise the architecture and terminology between the two WPs and we already have some alignment on the concepts even if the terminology is not yet fully aligned. The alignment will be documented in D.A.3 [13].

3.4.1 CloNe Perspective

Figure 3.5 provides an overview illustrating the participating domains(three in this case, two Data centres and one Network Operator) and the various components within each domain involved in the creation of the virtual infrastructure involved in the dynamic enterprise prototype.

The user describes his virtual infrastructure, by means of a description language like VXDL. This may be done via a graphical tool or manually created. The description is sent to one of the domains as a infrastructure provisioning request. This request is parsed and analysed and, using the goal translation function, translated it in the appropriate set of commands to be sent to the infrastructure layer of the data centre or network provider(s) as appropriate. Note that this component (virtual infrastructure description parsing and decomposition) is a logical function and hence as such can be implemented by each of the domains itself or by an independent broker (as

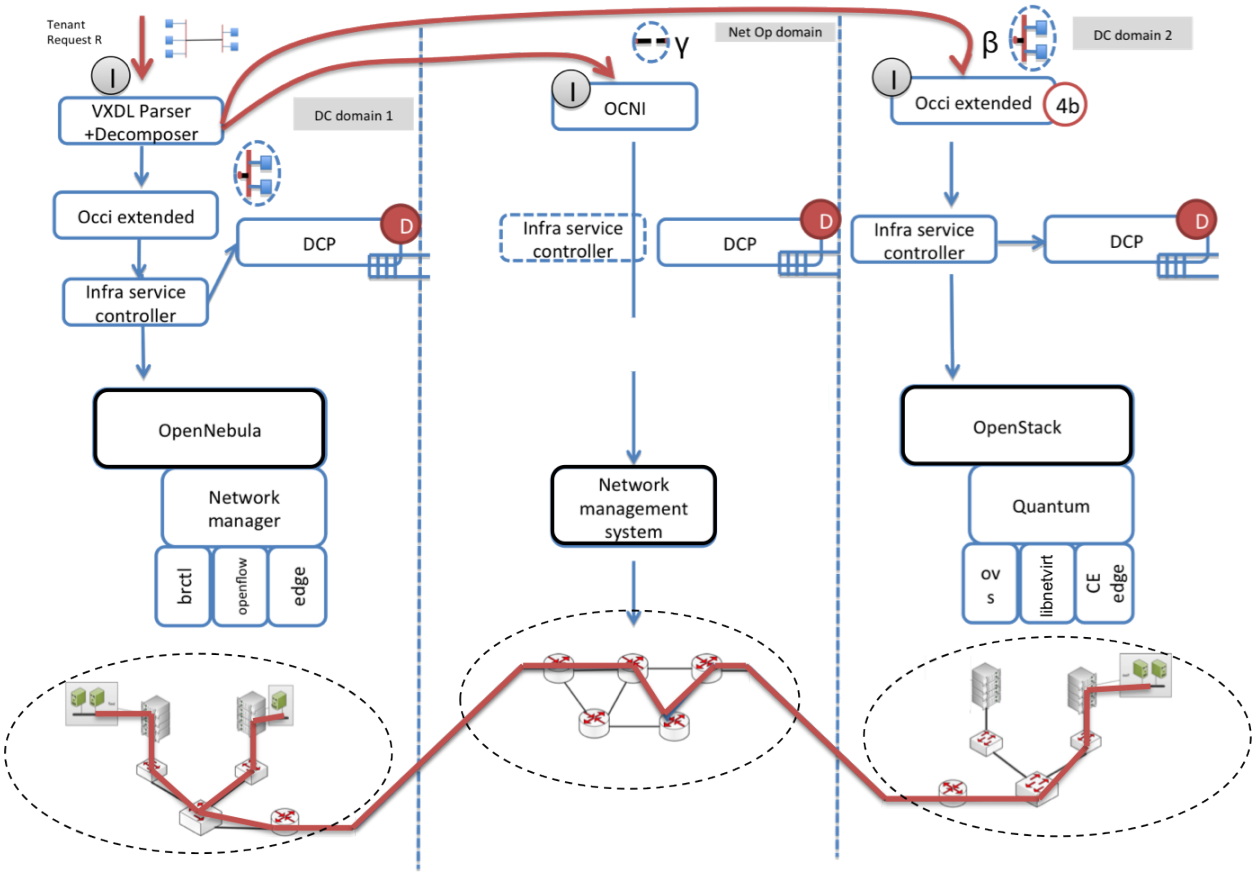


Figure 3.5: CloNe Prototype Architecture

done in the current prototype by means of CloudWeaver software) who does not own any physical infrastructure.

In the testbed, two different kinds of cloud operating systems is used: OpenNebula and OpenStack. In both cases, additional components need to be added to provide the functionality required. In each data centre and network domain a slightly different architecture is used.

Though only three domains are shown in the picture, multiple partners have implemented these domains and participate in the testbed.

Ericsson(EAB) is using OpenStack as cloud Operating System (OS). OCCI commands are translated into internal management commands that are sent to the VM and Network manager network elements for creating the the flash network slices. The FNS are implemented using MPLS in the network operator domain but with VLAN in the data centre domain. Ericsson also integrates the cloud resource management function developed by KTH in their OpenStack instantiation.

In the HP data centre, OpenStack is also used as cloud OS. Divertor/Cell-as-a-Service is used for the network virtualisation.

In the Institut Télécom data centre, OpenNebula is used as cloud OS. pyOCNI is used to translate the OCCI commands from CloudWeaver to OpenNebula API. The flash network slices are managed using OpenFlow.

Finally, PTIN is also using OpenNebula as cloud OS and MPLS for network virtualisation in their network domain.

3.4.2 OConS Perspective

First we give a sketch of the common OConS/CloNe architecture as depicted in Figure 3.6. The Distributed Cloud Manager (DCM) is part of the CloNe Distributed Infrastructure Layer and configures the cloud resources in the large. The DCM selects the involved domains and delegates tasks to the local, domain-specific services, like processing, storage and network services. Furthermore, the CloNe DCM informs the involved domains – using their CloNe CCs – about their connectivity in the cloud and their common attach points, namely Customer Edges (CEs) and Provider Edges (PEs). The DCM-CC communication with the domains uses a management protocol, which we call Distributed Cloud Management Protocol (DCMP), based on a further variant of Open Cloud Networking Interface (OCNI) to add OConS network resources and their management to OCCI. The DCM configures resources for the data centre domains as well for the connecting network domain.

The CC plays the central role in a data centre domain and is the equivalent of the OConS DCU. In the OConS case, it consists (cf. D.C.3 [4], chapter 2.1) of a database for permanent storage of network information and does the computation of the optimal resource placement and their connection.

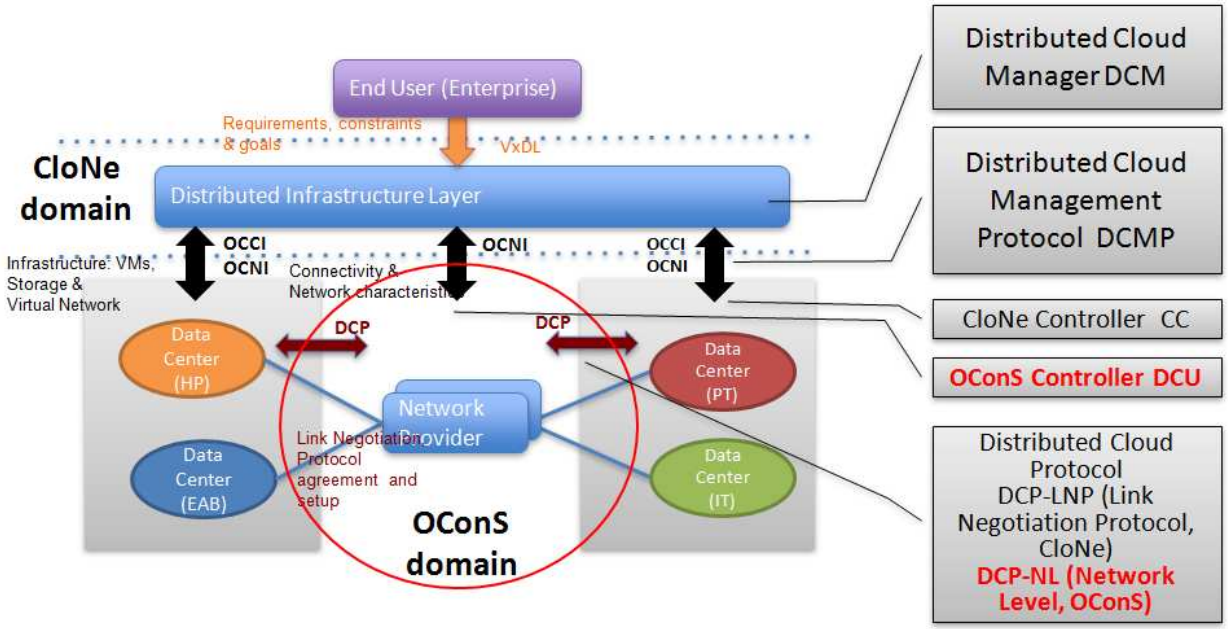


Figure 3.6: OConS/CloNe Architecture

Two domains of a cloud exchange their dynamic connectivity parameters using the Distributed Cloud Protocol (DCP). The DCP is a family of protocols that comes in two flavours for two different layers. The LNP organizes the adjacent CE-PE data plane connectivity at the attachment points via only CE-PE communication for the Link Level (as already defined by CloNe in D.D.1 [16]).

The DCP-NL is introduced for purposes of network aspects between the data centre (CloNe) domains and the networking (CloNe) domains. It dynamically builds the overall cloud topology and the routing/forwarding tables on the Network Level. Furthermore, the DCP-NL instantiates the application flows together with the needed processing resources and thus instantiates the end-to-end flows through the cloud. For setting up application flows according to network and processing load, it monitors performance indicators for processing load and link load. Thus the DCP-NL protocol is responsible for the dynamic resource allocation and monitoring over all involved domains via the

involved CCs, especially in OConS domains. While in CloNe the focus is on LNP to configure the connectivity, the actual focus in OConS is on the DCP-Network Layer protocol and its mechanisms, where LNP only is needed for configuring the data centre - network interface (CE-PE) at link level.

3.4.2.1 Roles and Functionality Split Between OConS and CloNe

Next we depict the roles and the functional split of OConS and CloNe within the foreseen prototype. Within the Cloud Data centre, the video processing for the flash crowd will be dynamically organized in a distributed cloud network between CloNe and OConS (cloud application). Thereby, CloNe manages the overall selection and reservation of involved DC and Network (NW) domains constituting the FNS DCMP part. Also, CloNe establishes the dynamic connectivity between DC and NW via CE-PE interface in its LNP role. OConS dynamically establishes the flows between the involved application nodes for multiple instances of distributed video processing resources in its DCP-NL role. Also, OConS monitors the network load – both link and processing – and redirects the application flows between the application end points as needed (i.e., seamlessly) in the DCP-NL.

3.4.2.2 Functionality of DCP-NL Between Involved CCs

Within the OConS control interfaces we differentiate three flavours of DCP-NL. First, on the CloNe-OConS interface between DC-CC and Network-CC the request for compute resources must be announced. This includes requirements about the kind and quantity of these compute resources but also the specification of the link requirements between or towards these compute resource.

Second, the OConS-OConS interface between Network-CCs (intra-domain) monitors the load status of nodes and links, computes intra-domain paths and sets up and releases them. Moreover, the intra-domain Network-CC interface allows to request inter-domain information to compute inter-domain paths and also facilitates to set up inter-domain paths and resource monitoring requests.

Third, the OConS-OConS interface between Network-CCs (inter-domain) is meant to perform the inquiry of a domain membership for a node. Moreover, this interface allows the retrieval of sub-path information within the neighbour domain including its metrics. This interface is ruled by the common overall agreements of the cloud provider and its involved network providers for the sake of a inter-cloud connectivity in a (future) CloNe-OConS ecosystem. Therefore we assume that known issues of information hiding between the providers may not apply in this case.

3.5 Prototype Components

We present here an overview of the different components involved in the dynamic enterprise prototype demonstration. The prototype components are described in more details in [4, 5].

3.5.1 CloudWeaver

The CloudWeaver tool is composed of two modules: a web-based graphical user interface and a core module responsible for processing and decomposing the request. This latter module is able to communicate with the different domains using the OCCI and OCNI interfaces for which they provide access points. Using these interfaces, the request for virtual infrastructure resources such as VMs, storage and FNS are made.

The graphical tool used by the end-user automatically generates the corresponding VXDL description.

The Decomposer module converts abstract or high level requirements into concrete actions to be performed on physical resources. Using a VXDL file as an input (either generated from the Graphical User Interface (GUI) or provided as a file), this module makes the translation from the VXDL request to a set of requirements for physical resources.

The different providers (data centres or network operators) are registered with CloudWeaver and specify some of their physical specificities (for example, their geographic position and resource types available). The VXDL request is split according to the requirements on the physical resources and configuration messages are sent to the providers matching the requirements.

As a single provider might not have enough resources to implement the user request or the user request might specify the use of different providers, it is necessary to be able to delegate the allocation to multiple providers. Cloudweaver contains goal translator and decomposer functionality to enable this.

To implement this feature, the CloudWeaver software can automatically generate the appropriate (intermediate) requests to network providers to instantiate one or several Flash Network Slice(s). Once they are requested and later created via the Infrastructure Service/IaaS interface, communication will be possible between the different data centre providers involved in the user-request.

3.5.2 pyOCNI

pyOCNI [17] is the actual implementation of OCNI. pyOCNI is developed in Python; it uses Eventlet as a concurrent networking library and ZODB3 as a native object database. pyOCNI is composed of six main blocks:

- Server: the component that can receive a request
- Client: the component that can send a request
- Serialization: the component that can do the information serialization
- Registry: the component that store the resource information
- Specification: the component that implement the OCCI and OCNI specification
- Backend: the component that implement the CRUD instruction related to the controllers.

To use pyOCNI with a specific controller, two steps should be done. The first step is the definition of the attributes that is needed by the controller. The second step is the implementation of the Create, Read, Update, Delete (CRUD) function related to the controller.

The OCNI interface is introduced in D.D.1 [16] and will be further refined in D.D.2 [5].

3.5.3 OCNI Back-end

The pyOCNI relies on plug-ins to adapt to a specific technology used by the network operator. In SAIL, two plug-ins are developed, one for MPLS (EAB) and one for OpenFlow (IT).

The OCNI back-end translate the OCNI into specific network configuration commands that are transmitted to the network elements.

3.5.4 Cloud Message Brokering Service

Cloud Message Brokering Service (CMBS) is one of the major components of the CloNe DCP. CMBS is a solution for exchanging messages between infrastructure providers. In the CloNe setup, it is used to exchange the LNP messages for connectivity configuration.The current implementation relies on ZeroMQ. More details on CMBS can be found be in [5].

Another incarnation of such a CloNe provider is the specialized management component handling NetInf provider requests, see Chapter 4.2.

3.5.5 LibNetVirt: Abstracting the Network

LibNetVirt is a library that uses the single router abstraction to describe a network. It is composed of a set of drivers. Each driver implements the required configuration for a specific underlying technology. LibNetVirt aims to reduce the time for the creation and termination of virtual networks. A common set of calls is defined and used to instantiate different Virtual Networks (VNs), in a programmatic and on demand fashion. Only the definition of the endpoint is necessary to create the network. Currently we are using OpenFlow but we have plans for other underlying technologies including MPLS.

In the Lisbon meeting, some of the library calls were demonstrated. First it was shown how the description of the network with XML and with the OCNI interface mentioned in Section 3.5.2. Secondly, two Flash Network Slices were created. Then, an additional endpoint was added and removed. Finally, the whole VN was removed. In each step the connectivity in and isolation of the VN was tested.

3.5.6 OConS Flow-Based Domain Connectivity Control

This prototype realizes experimental concepts for flow-based connectivity control per (network or technology) domain to be applied in multi-technology networks across layer 3 routing, layer 2 switching, and in future, possibly optical switched networks. Emphasis in on control, management and algorithmic flexibility.

This is achieved by introducing a unifying control element in each supported network domain called the DCU that separates the control functions from the data forwarding functions by concentrating most of the control plane mechanisms and intelligence in a single entity. The DCU combines concepts of an OpenFlow-based controller, PCE and TED for topology/resource advertisement and discovery as well as path computation and path/flow realization across multiple domains.

For experimentation, we followed an OpenFlow-like approach to realize a flexible and extensible test installation of multiple virtualised networking domains based on the MiniNet examples. This enables us to demonstrate intra- and inter-domain path computation and flow establishment in a mixed environment of cooperating (layer 2) switches and (layer 3) routers in virtual and physical networks. Such a scenario is common in interconnectivity of large distributed data centres.

The current demo can show control and management of a “data paths” between endpoints, e.g. located in distributed remote (cloud) data centres with inter-domain connectivity via web-based service and management interfaces from any browser (hence providing a REST-ful DCU interface for service invocation and orchestration).

For further details, see D.C.3 [4].

3.5.7 Using OpenFlow to interconnect Data Centres

The prototype demonstrates the interconnection between data centres using OConS connectivity services. Current approaches in order to offer data centre interconnection are based on complex networking using L2 and L3-VPN hardware solutions which are a difficult to manage and undermine its scalability.

The proposed use-case shows how an OConS end-point based on OpenFlow as technical solution can implement the data centre interconnection use case, using an infrastructure that implements multi-path transport services in a distributed/collaborative architecture. Openflow provides a potential solution for some of the limitations identified in data centres and helps lowering the entry

barrier to "programming-the-network" solutions. The basic idea is to introduce an OpenFlow-based element that provides new paths towards other data centres and remove some expensive network elements and simplify both control and management planes. It also provides a tag-less scenario for current IaaS services that is cheaper and more scalable than state-of-the-art hardware based solutions.

The prototype implementation also serves as a proof-of-concept for WP-C/WP-D cross work package integration. We are exploring possible solutions to integrate the control interfaces developed in WP-D – namely the Distributed Control Plane (DCP) and the OCNI/OCCI interfaces – in the prototype and how they can be used to integrate this prototype with the one presented in Section 3.5.6. One of the components developed in the scope of WP-C that will be retrofitted into WP-D is the OpenFlow specification for Open Connectivity Services.

For further details, see D.C.3 [4].

3.5.8 Dynamic Resource Allocation with Management Objectives

This component provides a resource management system for cloud environment that is integrated into the OpenStack cloud management software. It focuses on computing resource allocation to users (i.e., tenants) through virtual machines (VMs). Our resource management system has two unique features.

First, it allows specifying management objectives that specify the desired property of resource allocation. Examples of management objectives include balanced load across physical resources, minimal energy consumption by the physical infrastructure and minimal network bandwidth usage. The demo showcases the "balanced load" objective and we are currently working on the "minimal energy consumption" management objective.

Second, the management system supports dynamic resource allocation. Specifically, it implements a mechanism that changes the mapping of VMs to physical machines, in response to events such as change in resource demand, in order to ensure that the management objective is achieved at all times. As such a feature is lacking in OpenStack (and all other open source cloud management software we know of), we implemented and integrated our own gossip-based resource optimization protocol that was reported earlier in D.D.1 [16].

3.6 Evaluation and Assessment of Prototyping

The purpose of the WP-D specific prototyping is essentially to evaluate the feasibility of the selected use cases, specifically dynamic enterprise and elastic video distribution. This entails in large part to determine suitability and completeness of the proposed control plane protocols and APIs. Furthermore, a second goal is to validate some of those management functions that need an integrated testbed for evaluation.

In this regard, our conclusion so far is that the implementation of dynamic enterprise and creation of on demand elastic networking between multiple remote data centres is feasible as it has been show-cased. This is the validation of the interfaces, both northbound (from end-user to infrastructure providers) as well as the eastbound (communication and dynamic negotiation between providers). Also the main concepts developed in the CloNe architecture appear to be valid and meet our requirements. On the vertical angle, the validation of specific features and functionalities is going to be achieved. The proved to be feasible and valid functions are resource management within the data centre, network abstract model and elasticity are considered to be successful.

4 Elastic NetInf Deployment

This chapter describes an elastic growing and shrinking virtualized deployment of a NetInf implementation on CloNe's testbed. Starting with a discussion of the benefits, the demonstration setup and scenarios are described and the envisioned architecture specified.

4.1 Objectives and Expected Benefits

The NetInf network architecture differs significantly from today Internet architecture. That is, for NetInf (and any other information-centric networking approach) today's network routing protocols need to be changed, name resolution evolves from and end-to-end protocol into a intrinsic network function, and in-network storage is required.

Thus, when implementing or deploying a NetInf network there are two options. One option is to update/replace current networking hardware and software (protocols). Such a "long-term" option requires a significant amount of development effort and investment, which is difficult to commit when the benefits are unclear. A more short-term option is to build an overlay of NetInf-enabled nodes/routers which rely on convergence layers that translate NetInf into existing network protocols. However the problem remains that NetInf is not designed as an end-to-end solution, and ideally requires *in-network* resources or at least resources close to the core of the Internet (for caching, routing and name resolution).

CloNe provides a framework for managing and interconnecting virtual resources providing Infrastructure as a Service (IaaS). CloNe's architecture features the possibility to allocate resources across multiple locations (i. e., data centres) and cloud providers. This especially includes the provisioning of *network* resources inside and between data centres. Moreover, CloNe aims to enable easy adoption to changing conditions, such as service load, load in the data centre, etc.

The desire of NetInf to use resources close to the network and its ability to utilize existing network protocols (convergence layer) make it a perfect candidate for a dynamic CloNe deployment across different DCs. This not only sketches a migration path towards an Internet-wide NetInf deployment without upfront hurdles and investments of network operators, but also allows for coexistence of different network architectures.

A typical NetInf overlay would run on hosts consuming content (e. g., at home) and on host offering content (e. g., in enterprises). Given that cloud data centres are typically closer to the network core than enterprises and homes, using CloNe the NetInf overlay can be extended with such nodes. In case (some) routers are CloNe-enabled, those could as well be turned into a NetInf node and could serve as NetInf router and/or rendezvous server.

Once cloud computing and storage solutions are spreading wider across and further into the network, CloNe allows NetInf an almost ideal deployment in terms of location of resources. The challenge, however, is to provide appropriate interfaces between the actually used hardware (CloNe) and the information-centric networking mechanisms (NetInf). This is where the *Elastic NetInf Deployment* prototype will close existing gaps:

- To enable an optimal placement across multiple locations and IaaS providers, we will extend the existing CloNe testbed with a *resource location aware management component* that divides service deployment requests into small chunks and send these to appropriate data centres.

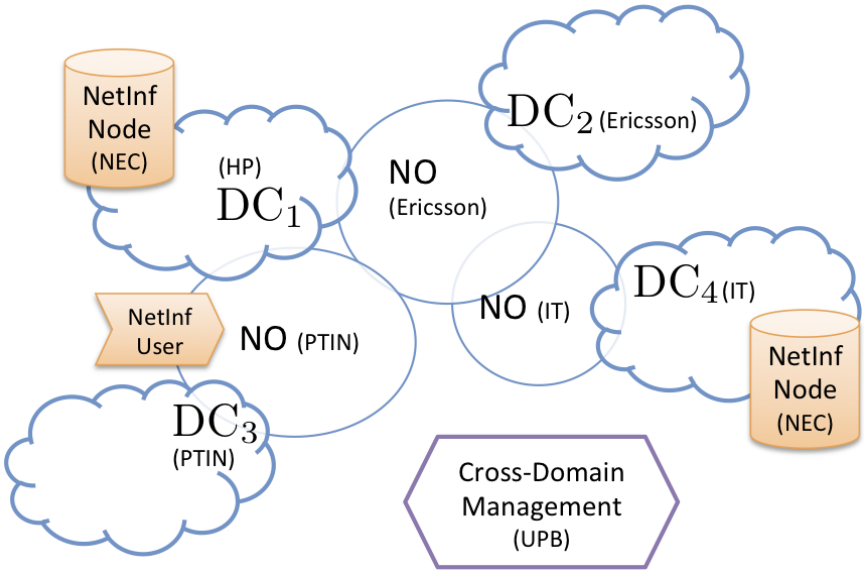


Figure 4.1: Demonstration Setup for the Elastic NetInf Deployment Scenarios

- In order to detect changing conditions we will extend both CloNe and NetInf to monitor Key Performance Indicators (KPIs), such as bandwidth utilization, CPU usage in the virtual machines as well as cache hit rates or object request load. Then the management component will use these measured KPIs to adapt the deployment of NetInf nodes, e. g., move NetInf nodes to where they are required, create additional NetInf nodes in times of overload, or shutdown nodes when they are under utilized.
- The last contribution is to extend an existing NetInf implementation (NNRP in this case) to support automatic configuration and integration of nodes joining and leaving the NetInf overlay.

4.2 Demonstration Setup

Before detailing the scenarios to be show in this section we summarize the setup that all scenarios rely on. Figure 4.1 shows the three main components of this prototype: (i) the CloNe testbed (clouds and circles), (ii) the NetInf deployment (cylinders and arrows) and (iii) the cross-domain, load-adaptive management component (hexagon). For details on the CloNe testbed and the NetInf implementation we refer to corresponding sections of this deliverable.

CloNe Testbed We use the general *Dynamic Enterprise* testbed, which is described in more detail in Section 3.2.1 (see also D.D.2 [5]). The high-level layout of the testbed is depicted in Figure 4.1 and is composed of four DCs at different locations and three NOs connecting the DCs. For this *Elastic NetInf Deployment* prototype we will in particular use the following features of the testbed:

- Use integrated IaaS capabilities to deploy Virtual Machine (VM) images (of NetInf nodes) and provide network connectivity between them.
- Both the DCs and the NOs externally offer OCCI and OCNI interfaces, which will be used by the cross-domain management.
- DCs can monitor VM's CPU utilization as NOs can monitor connection's bandwidth consumption, e.g., provided by OConS (compare Section 3.1). We will use such monitoring of at least one location to inform the management module.

Cross-domain & Load-Adaptive Management As an extension to this foundation, a new management component offers higher level services by aggregating the resources of all infrastructures, thus acting like a broker. This is a specialized version of the similar CloNe provider concept (e. g., CloudWeaver module, Section 3.5.1) but adapted with a specialized interface to handle NetInf providers request as well.

This component manages the deployment of NetInf nodes across the data centres. An internal logic decides on where to place NetInf nodes. Therefore NetInf sends load data, the KPI, to the management component steering the placement: New nodes are deployed where needed and old nodes are shut down if no longer needed. Thus the location and the amount of nodes are adapted to the need and load of NetInf.

In contract to the other domains, this provider act only on cross-domain level as a special infrastructure provider without own infrastructure.

NetInf Router and Users As NetInf implementation we use the NNRP (see Section 2.5.3 for more details) as well as client software to query objects from the NNRP instantiation of NetInf. The relevant parts for this prototype are:

- The NNRP implementation allows Linux-based deployment inside virtual machines. It is build modular and thus allows easy configuration of NetInf nodes as pure routers or as NetInf caches.
- We use a NetInf enabled `wget` to generate requests for objects from a configurable NetInf node. This allows us to steer requests to certain "entry" points in the NetInf overlay.
- NNRP nodes will provide request load and cache hit rates to the management module.

4.3 Demonstration Storyboard

In this cross-work package cooperation, we show multiple benefits of running NetInf on CloNe. We gradually build up the complexity of the demonstration in order to highlight different achievements in each scenario.

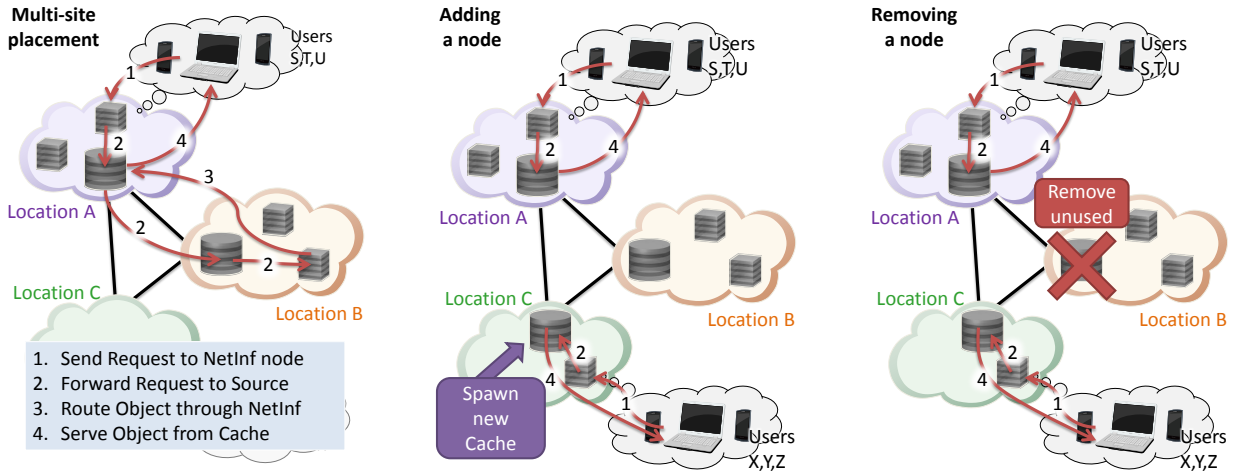
4.3.1 Scenario A: Multi-location NetInf Deployment

In a first step, a NetInf service provider will request the deployment of a set of NetInf nodes (i. e., VMs) including where to deploy each node. For this demo, we envision location specifications such as "at location/data centre X". CloNe will then determine the best (the one that fits most requirements) allocation of available resources and initiate the deployment. Then, we will publish content at one node and request the object at different nodes (including nodes at different locations). Figure 4.2a shows such a multi-site deployment and how NetInf operates inside (normal NetInf nodes are cubes and caches are cylinders).

Goal: This Scenario shows an on-demand deployment of NetInf nodes across multiple domains and sites, which can communicate with each other and serve objects to clients.

4.3.2 Scenario B: On-demand Changes to NetInf Deployment

Starting from an initial deployment of NetInf nodes as described in Scenario A, the NetInf service provider wants to extend its deployment. A reason could be that it expects an increase in utilization due to a planned event in the future. In this scenario, the NetInf provider requests an additional cache (NetInf node with activated caching) at a certain location (Figure 4.2b). After the new cache is deployed we publish new content there and retrieve it from another NetInf node. Likewise we



(a) Scenario A: Multi-site placement

(b) Scenario B: Add node on-demand

(c) Scenario B: Remove node on-demand

Figure 4.2: Scenarios A & B for the Elastic NetInf Use-case

will request a remote object at that cache to show complete integration with the rest of the NetInf overlay.

After a while, the event is over and the NetInf provider wants to remove the additional cache to save deployment costs. Thus, it tells CloNe to shut down a certain node. Of course, the NetInf provider has to ensure first that the node requested for shut-down does not store any object that is available exclusively at that node. During this process we keep generating requests for objects (Figure 4.2c).

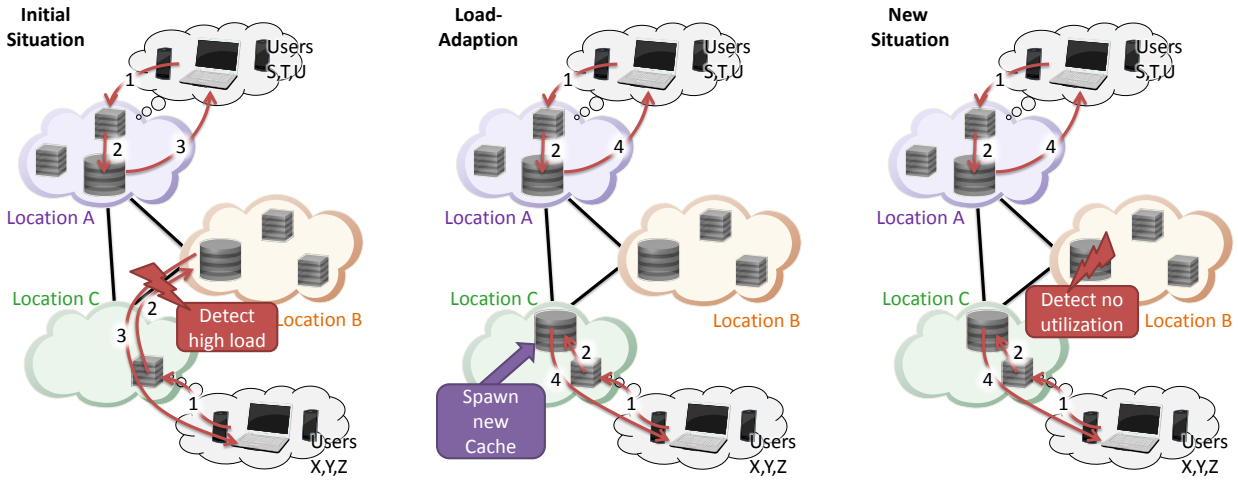
Goal: In this scenario, CloNe can grow and shrink an elastic NetInf deployment on demand while NetInf is able to continuously serve objects. Moreover, we show that our NetInf implementation can easily deal with nodes arriving to and departing from the overlay (elasticity).

4.3.3 Scenario C: Load-adaptive NetInf Deployment

Again we base this show case on Scenario A. Consider Figure 4.3a: we let users S, T, and U repeatedly request objects at Location A (on top). These are typically served from the cache in Location A. We also have users X, Y, and Z requesting objects at Location C. However, there is no cache in Location C and all the requests are served from Location B. In this situation, CloNe can detect the high load on the inter-location link and react with a pre-defined, NetInf-specific action. This action could be part of the deployment request from the NetInf provider. In our example this action is to add a new cache in Location C (Figure 4.3b).

From then on, the requested objects will also be cached in Location C, and after a while no further inter-location traffic is required (except for updates, of course). Because CloNe not only monitors the load on the inter-location links but also the CPU load of the VMs, it now detects that the cache in Location B is no longer required (Figure 4.3c). Again using a predefined policy, this unused node is shut down to save cost (causing a similar situation as in Figure 4.2c).

Goal: This scenario shows how monitored KPIs can help to automatically adapt the deployment to different load conditions.



(a) Detetion of high load (b) React by spawning a new cache (c) Detect low utilization

Figure 4.3: Scenario C: Load-adaptive NetInf Deployment

4.3.4 Envisioned Future Extension

Although we do not plan to implement the following extensions, we do envision them showing further potential of CloNe e. g., when running NetInf or other types of "applications" on top of it.

Topology sharing An option for closer integration of NetInf and CloNe would be to share topology information. For example, CloNe could provided the (physical or abstracted) topology including location information to NetInf. NetInf can in turn use this information to form an overlay that resembles the real network. To some degree this case can be realized by letting the management component export the deployment topology, but NetInf does not yet provide means to use such information.

Another option would be to provide the NetInf overlay structure as an additional input to CloNe when processing change requests (both on-demand and triggered from monitoring KPIs). The overlay structure allows to make a better informed allocation of resources, e. g., through a NetInf-provided algorithm.

Migration of nodes Comparable to our load-adaptive scenario, we also envision to migrate a node from one location to another instead of starting a new one and removing an old. NetInf caches are candidates for this, however nodes with more complex state (e.g. in-network computing) would even benefit more. This has the advantage that state (e. g., cached objects) are moved along together with the node.

A use case could be that nodes can travel around the globe following the peak usage hours or following their roaming mobile users, reducing the network distance to a minimum. Migrating NetInf caches trades off the need to re-populate a new cache (delay) for higher migration cost (copying the cached objects over the network). Depending on the economic situation it might be preferable to instead let NetInf move along the objects from cache to cache internally. This requires more caches but allows smarter per-object transfers.

Considering such a peak time zone migration, OConS could provide special connectivity for the high bandwidth virtual machine transfers, keeping the cost for the migration overhead low.

Adaption to NetInf-provided KPIs Scenario C adapts the NetInf deployment towards general KPIs measured by CloNe. In a similar fashion, it would be possible to adapt the deployment to service specific KPIs, such as Cache-Hit-Rate in the case of NetInf.

4.4 Architecture

The envisioned architecture (shown in Figure 4.4) has a NetInf and a CloNe part. The NetInf prototype is running in virtual machines on CloNe resources and thus can be spread in a real world, wide area network across multiple network operators.

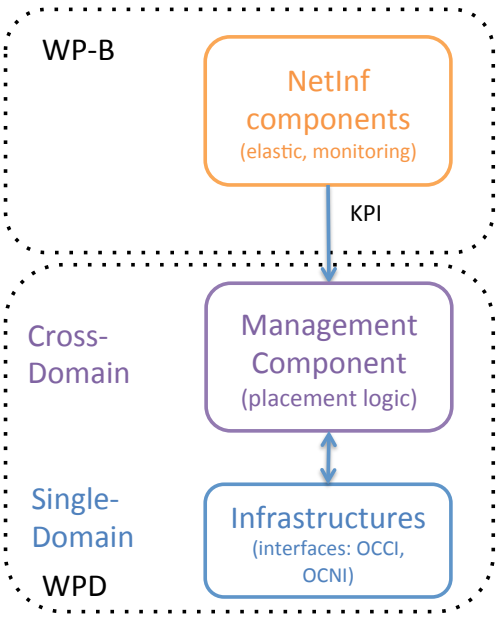


Figure 4.4: System Architecture and Responsible Cross Work

To realize that, the prototype architecture is composed of three components or parts working together on different levels: the NetInf nodes, a Management component, and the CloNe-enabled infrastructure. This architecture distinguishes between two operators or providers: the NetInf provider and the CloNe provider. Both can be the same legal entity, but we focused on a more flexible model in which an arbitrary NetInf company can deploy its software on arbitrary CloNe providers. Hence, a NetInf provider uses the service offered by a CloNe provider.

For deploying NetInf on cloud resources, NetInf software will run in virtual machines. These have to be setup and managed by the NetInf provider. The provider also knows how the different NetInf nodes (e. g., caches, name resolution) fit together. This description and the virtual machines are handed over to a management component operated by the CloNe provider.

This management component deploys the pre installed NetInf software onto CloNe-enabled infrastructure. As a mediator between one NetInf provider and different CloNe infrastructures, this component takes care of infrastructure discovery, pre-uploading necessary images, and establishing flash network slice connectivity between NetInf nodes.

CloNe infrastructures are data or compute centres at different locations, connected by networks. On this level, the provisioning of resources for virtual machines and the coordination between data centres and network operators takes place. A detailed description is available in D.D.1 [16].

4.4.1 NetInf Components

We refer the reader to Chapter 2 and existing SAIL Deliverables D.B.1 and D.B.2 [6,10] for details on the architecture and features of NetInf. However, for this prototype it is important to detail the way how NetInf nodes can be instantiated.

Different nodes of a NetInf system have different roles and hence might run different software stacks. These software stacks are provided in form of virtual machine images. For the scenarios it does not matter if each role has its own image or one image can cover different roles.

The NetInf prototype needs to be elastic in a way that configuration or contextualization is done during or directly after VM start-up. This allows to launch a VM using the same master image at different data centres. Only this way, large-scale deployment is possible. During the start-up process, a NetInf node is informed how to connect to the rest of the system or which role it plays.

Any NetInf implementation which covers these requirements for the virtual machine image creation and allows to query and retrieve named objects can be used in the described scenarios. We base this prototype on NNRP, the NetInf implementation from NEC, which supports all these requirements.

4.4.2 Management Component

A CloNe provider offers a single point of access to deploy NetInf nodes for a NetInf provider. This CloNe provider can decide how much of the request can be served by itself and which parts of the request are delegated to further CloNe infrastructure providers. Thus, this management component takes place at the cross-domain layer as an example for a specialized distributed infrastructure service. Details of the CloNe architecture can be found in D.D.1 [16] Section 2.

The important functionality of this management component is:

- As a single point of access for tenants and NetInf providers, it discovers and knows CloNe infrastructure providers. For our scenarios this is less important so we assume this functionality.
- The NetInf provider describes virtual machine images, resource requirements, and policies within the requests. These requests contain information like NetInf destination location (Scenario A, B) or policies how to react on load changes (Scenario C).
- The logic to decompose an all-in-all request across different data centre locations (VM provisioning). Before the first deployment, NetInf prototype images have to be uploaded to the data centres using the OCCI interface. The same interface is later used to request the deployment of NetInf prototype VMs.
- The most advanced functionality is used for Scenario C: Upon regularly collected and received load data (KPIs), a placement logic decides automatically to deploy new or shut down old NetInf nodes. The decisions are configured by policies and for the scenarios, upper and lower thresholds for acceptable load are defined.

The management component rely on the functionality provided by the lower single domain layer (Section 4 in [16]) through the infrastructure service interface, both from data centres (via OCCI) and NO (via OCNI).

4.5 Evaluation and Assessment of Prototyping

The purpose of the described demonstration is to evaluate the feasibility of a virtualized NetInf prototype deployment across multiple domains. We are not aware of any system or prototype that provides ICN functionality on top of virtualized ressources. Therefore it is not possible to compare

our solution in terms of performance or other metrics. For that reason we consider the prototype successful if Scenario C shows the intented reduction in inter-location link utilization.

As a side effect the prototype will examplify service-specific interfaces and placement logic (in this case NetInf-specific) as well as KPI monitoring inside NetInf and CloNe. Moreover, the prototype validates CloNe's architecture and north- and eastbound interfaces to support higher level services.

5 Other prototyping activities

This chapter presents additional prototyping activities that occurred during the SAIL project. many of those activities have been used to validate or experiment specific SAIL concepts and have been reused to some extent in the more integrated prototype demonstration described in the previous sections. For each contribution, we will present the main objectives and give a short description of the prototype. In many cases, more detailed description of the prototypes are found in the WP-specific deliverables (D.B.4, D.C.3, D.D.2 [3–5]).

5.1 WP-B Network of Information

5.1.1 Localized-CDN with OpenNetInf

The Localized-CDN prototype demonstrates a rendezvous-based ICN control plane that operates on a legacy (non-ICN) data plane. The prototype showcases migration path(s) from the current Internet to the ICN, for an ICN system which is not tied to any specific transport or network layer. Rendezvous requires powerful search/name resolution functions and for this we have used OpenNetInf prototype components from University of Paderborn (UPB).

This prototype deals with content distribution where ICN technology is used to manage distributed content storage/caching system. This system retrieves content from the network to the local caching system based on the request and serves them from there. The system itself is hidden from the network and communication endpoints, thus allowing legacy applications to work with it. In other words, the prototype creates an ICN domain where (not necessarily on-path) local caches/storages are created in the operator/ISP networks and managed and operated through the rendezvous-based control plane.

The prototype is implemented using Java and contains the following separately scalable modules: monitor, rendezvous node, NRS (OpenNetInf) and cache node. In addition, the prototype uses web browsers or VLC players[1] as subscribers, Wbox server[2] as a content distributor, and DNS server. As such, our prototype is self-contained and demonstrates all the key parts of a rendezvous-based ICN architecture. Further details of the prototype and scenario are available in D.B.1 [10] and D.B.2 [6].

The Localized-CDN (L-CDN) prototype was demonstrated in the Lisbon SAIL meeting; the main message was to show how ICN concepts and mechanisms work inside the network without having to reveal the ICN functionalities outside the L-CDN network, i.e., even legacy clients and servers can interact with the L-CDN network.

The separate prototype and scenario work on the L-CDN has ended; nevertheless, selected functionalities of the L-CDN will be merged and incorporated into the EwLC scenario and prototyping activities. The L-CDN functionalities will add infrastructure support in the EwLC scenario, thus enlarge the footprint of EwLC usage.

[1] http://www.videolan.org/vlc/
[2] http://www.hping.org/wbox/

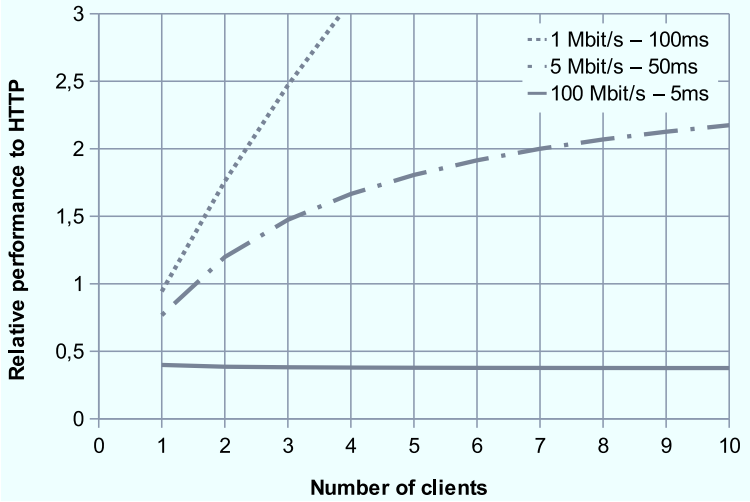


Figure 5.1: Performance of Subversion/OpenNetInf Relative to Subversion/HTTP (values above 1 are better)

5.1.2 Subversion over OpenNetInf

It is easy to argue that the ICN approach is ideal for large-scale information distribution. It is certainly true that the largest efficiency gain is expected for this type of application. As a complement, we have also investigated the benefit for another type of application which can take advantage of local ICN caching rather than always interacting with a remote server, making the application less dependent on good connectivity to that server. As the example application scenario we are using our own project meetings. A group of people in the same room or building is collaborating and sharing documents using a Subversion[3] version control repository. The repository is typically located in a different country and the available access network capacity is less than the demand.

We modified a Subversion server and client to use the communication service of the two ICN research prototypes OpenNetInf[4], from the 4WARD EU project, and CCNx[5], from the CCN project at PARC. The modifications were straightforward – one important experience in itself from the exercise.

By using the ICN communication services, Subversion clients automatically fetch updated files in the repository from other, nearby clients when possible. The experiments show (Figure 5.1) that already with two clients, there is a performance benefit in common scenarios despite the fact that these research prototypes are far from optimised.

More details of this work can be found in a master thesis report [18] and in a workshop paper [19].

5.1.3 Video distribution over an Information-centric Disruption Tolerant Network

In previous projects, we have built a Disruption/Delay Tolerant Network (DTN) system for the local population of the Padjelanta national park in Swedish Lapland. This area is very remote and does not have access to traditional communication infrastructure. Within SAIL, we build on previous research to combine the NetInf ideas with our DTN experience.

Since much Internet traffic today consists of video content, it is likely that users in a remote community will also be interested in this content. The problem is that this type of content is large, which leads to high system resource usage when retrieving it. This becomes worse when multiple

[3] `http://subversion.apache.org/`
[4] `http://www.netinf.org/`
[5] `http://www.ccnx.org/`

users in the same region download the same popular content and waste resources by transmitting the same bits multiple times. As it is common that many users wants to access the same content (e.g., the latest news broadcast), we saw the potential to leverage the benefits of NetInf in conjunction with DTNs by making intermediate nodes aware of the name of the requested content and by allowing them to cache the content to serve subsequent requests from their local cache. Thus, we have developed and implemented DTVideo, a video distribution service over DTNs that make use of the BPQ extension [20] (developed by Trinity College Dublin (TCD), SICS, and NEC within SAIL) to the Bundle Protocol [21] to leverage caching in the network. This prototype was developed as a proof of concept and enables us to show how the use of BPQ dramatically reduces the resources usage of the system. In addition, the prototype was deployed in a real network with real end users in the Swedish mountains.

Reference for further details: [22]

5.1.4 NetInf Device Prototype

The aim here is to develop a mobile tablet device that uses DTN (via the Bundle Protocol [21]) and NetInf protocols as its primary means of communication with other nodes and that could also be used in the EwLC scenario. A tablet was chosen as the physical device as its interface limits the configuration options available to the end user, and we are also interested in how such limitations might affect the usability of the device.

Given the time and resources available, the scope of the prototype has been limited to the following functional areas: search for and access NDOs via a filesystem, discovery and access to various system aspects of the tablet via the /proc and /dev filing systems.

The tablet hardware offers WiFi (802.11b/d/g) connectivity as is conventional on tablets.

The functionality currently being integrated includes:

- DTN2 Reference Implementation Bundle Protocol Agent providing an NDO cache via the bundle store provided by DTN2,
- `ni:` naming of the bundles using the BPQ extension [20] to the Bundle Protocol,
- publication and retrieval of NDOs over a NetInf DTN Bundle Protocol Convergence Layer
- access to the NDO cache as a filing system via the ni: names,
- textual search in the cached NDOs using the Apache Lucene[6] search engine,
- local and remote searching via a Firefox web browser plug-in, and
- access to operating system information using `ni:` names to refer to items in /proc and /dev on the tablet.

TCD are also intending to provide the ability to act as a gateway between the Hypertext Transport Protocol (HTTP)- and DTN-based convergence layers using the NetInf nilib code that has been developed as part of the SAIL project. TCD will endeavour to have the EwLC directory application running on this device in due course.

[6] http://lucene.apache.org/

5.1.5 Global Information Network (GIN) Prototype

This proof-of-concept prototype is mainly aimed to assess the feasibility of an information-centric network based on the GIN approach [23]. GIN takes an integrated approach for name-based resolution and routing. The name resolution system is designed according to the MDHT/REX architecture. Multi-level Distributed Hash Table (MDHT) is a distributed hierarchical DHT with embedded topology which provides intra-domain, local-aware resolution services in a service provider network or in a customer network. Name resolution in the global Internet is provided by extending MDHT with the independent Resolution Exchange System (REX) which provides support for the inter-domain resolution.

MDHT works in conjunction with a stateless packet delivery protocol, namely the GIN Protocol (GINP), that manages the name-based forwarding of data units over a heterogeneous underlay of interconnected L2/L3 networks. Data requests are routed over a resolution path according to the MDHT hierarchy, but following data packets are fast forwarded over the shortest direct path. Performance and scalability of GIN/MDHT have been analyzed in [23].

GIN, as a general-purpose, name-based networking system, is not tied to a specific application scenario. The main idea is to provide a network infrastructure able to interconnect not only devices but also data objects and application objects. For the time being, there are no plans about demonstrating any specific application scenario or use case. In principle, however, GIN may address scenarios for serverless web, with in-network cooperative caching and storage, native support for object mobility and multicast traffic, efficient distribution of data as well as support of real-time communication. The proof-of concept prototype shall allow the assessment of the GIN/MDHT architecture and the demonstration of basic ICN features, such as publishing and retrieval of data objects by name with anycast routing and locality exploitation (i.e. content locality, resolution locality, routing locality) over heterogeneous L2/L3 underlays.

The GIN proof-of-concept prototype comprises two main components:

1. a network node with an MDHT Dictionary and a Next Hop Table for fast lookup and forwarding of packets with named destinations;

2. a proxy client which receives HTTP requests from legacy clients and performs registration, upload and download of data in the GIN network.

As to the practical implementation of the prototype, GIN nodes will be implemented on FreeBSD machines, in C. The prototype will also be provided with Ethernet, IPv4 and IPv6 interfaces. An Apache Server will be run over proxy clients to handle HTTP requests, with PHP procedures. A testbed of a few virtual nodes will be deployed over a VMware platform. The GIN prototype will provide a packet interface at the GIN layer. That is, it will be possible for the applications to send/receive data packets forged according to the GIN Protocol (GINP). A high-level ICN interface will also be implemented, offering functionalities for registration (PUT) and retrieval (GET) of data objects.

Current GIN node implementation provides the ability to forward GIN packets by ID over heterogeneous networks comprising Ethernet, IPv4 and IPv6 segments. Next Hop Table (NHT) and forwarding mechanisms have been implemented according to the GINP stateless delivery protocol. At the moment, routes for named entities can be only statically configured in the NHT with a command interface. The Dictionary (i.e. NRS) function is currently under development.

In addition to the already implemented GIN forwarding system, we expect the following items to be ready by the end of the summer 2012: HTTP legacy proxy client, Dictionary function for Name Resolution according to the MDHT scheme, primitives for publishing/retrieval of data objects by name. An end-to-end multipath transport protocol and dynamic routing mechanisms will be implemented only in a future release.

5.2 WP-C Open Connectivity Services

5.2.1 Mobility with OConS

Dynamic Distributed Mobility with OConS: This prototype has two main goals. First, it aims at assessing the feasibility of the OConS architecture, by implementing and orchestrating the different functional entities required, e.g., to offer an enhanced mobility service. Second, to offer this mobility service, the prototype implements a configurable access selection mechanism in conjunction with a mobility execution mechanism, thus showing the dynamic activation of mobility anchors in Access Nodes/Routers.

Accordingly, on one hand, the prototype component architecture complies as much as possible with the current OConS architecture, and it includes the supporting functional entities as specified within the OConS framework (see next paragraphs). On the other hand, since we have exemplified with a Mobility-related service, the corresponding OConS mobility management entities were also part of this prototype; thus, we enable the use of direct IP routing of traffic flows/sessions initiated on mobile's current Access Router (AR), while supporting handovers and traffic forwarding/tunnelling between the Access Routers (current and previously used anchor-ARs); moreover, the combination of the two types of mechanisms (selection and execution) enables the possibility to include different mobility-related procedures, which are triggered by events coming from various sources (e.g., high load situation at an access element, low link quality, etc).

In the OConS approach all entities are controlled by an supporting component that fulfills the role of the Service Orchestration Process (Service Orchestration Process (SOP)). In this sense, the rest of the OConS entities need to register to an Orchestration Register (OR), i.e., during the bootstrapping process. Afterwards, any entity is able to know the location of any other one relying on the SOP and the OR functionalities, as indicated in the configuration phase; thus, once the registration has been completed, the inter-entity and inter-node communication can be carried out whenever appropriate.

As mentioned before, one of the goals of this prototype was to assess the OConS architecture feasibility; hence, information gathering, decision-making, and execution phases have been carried out according to the OConS architecture; in this sense, they have followed the specified procedures, such as bootstrapping, discovering, and registration. Likewise, the OConS entities for mobility management (Decision Entity, *DE*; Information Entity, *IE*; Execution Entity, *EE*) have been implemented within both the user terminal and the access elements and their appropriate operation has been validated.

The current running prototype will be enhanced so as to be better aligned with the OConS specification, which will be reported in coming deliverables. In particular, special attention will be paid to the refinement of the orchestration process and to the signalling protocols between the corresponding entities. The implementation will be updated so as to generate a library which could be used by other developers as a wrapper to support OConS functionalities. However, the available functionalities have served their initial purpose, since the demonstration shows the possibility to integrate a particular mechanism (which is relevant for the flash crowd use case) within the OConS framework and, furthermore, how it can be enhanced by means of such integration. For further details, please see D.C.3 [4].

The use of this demonstration is relevant for the flash crowd scenario, since the optimal data path is used each time a new flow is launched, and the forwarding/tunnelling functions are activated only when needed so as to support flows' delivery in mobility situations; this brings about a more efficient connectivity since it uses direct routing whenever possible.

5.2.2 Optimising User-Perceived Quality

This is a bi-partite prototype with focus on two different aspects to provide end-users with acceptable content quality

- A decision system matching the quality profiles of currently running applications and their parameters to the available connectivity options in order to deliver the best possible quality to the user. Quality profiles also include power consumption and monetary cost, which are generic parameters relevant to the users' perception of quality. This mechanism can integrate, amongst others, into the DMM mechanism of Section 5.2.1.
- A content-aware proxy to support legacy HTML5-based applications, leveraging the `Content-Type` header of HTTP transactions to identify the type of object being transferred and select the most appropriate way to do so in terms of path, reliability or congestion-control use to transport it.

The decision system is a pure DE which interacts with both other types of OConS entities. It relies on two types of IEs to provide information on the available networks and their characteristics, and to be informed of the currently running applications and their quality profiles. It provides instructions for EEs such as described in Section 5.2.1 or, more generally, any EE which can manage network-layer connectivity and paths.

The HTML5 proxy presents itself as an EE establishing the HTTP connections in the selected way. However, in early development stages, it is also expected that it would act as a DE, selecting paths on its own. Ultimately, the DE described above is expected to provide decisions for the proxy as well. Information about document types is expected to come to a remote IE, either co-located with the web server, or somewhere along the path (e. g., OConS-aware caching proxy), but is currently obtained from a `HEAD` HTTP request which, though it provides the same functionnality for development purposes, is likely to introduce some additional delay (about one RTT) until integration with the OConS library is done.

Prototyping activities are done along three axes. A simulation model using OPNET[7] has been developed in order to evaluate the quality-aware DE, implemented as a linear programming problem using CPLEX,[8] in complete network scenarios [24]. Building blocks for these scenarios include LTE and Wi-Fi connectivity, as well as one or more mobile devices using an early OConS protocol model to exchange information on which to base the decision and enforce it. Current results with this model have shown improvements on the quality perceived by the user, at the cost of slightly increased price and power consumption. This increase is, however, in no way commensurate with other approaches that enable all interfaces at all times [25].

An issue surfaced for web traffic as the decision for this type of flows is based on ITU's objective models for IP traffic [26], which rely on knowledge of the user's expectation in terms of completion time. Various formulations (e. g., moving averages or estimates based on recent behaviours or quality) have been explored in order to estimate this expectation, but none has been fully satisfactory so far. The OPNET model is flexible enough to allow us to directly explore possible solutions to this problem

Apart from simulations, real-system prototyping has started on two aspects. An Android-based MIPv6 [27] testbed is currently being built. It includes multiple care-of address support [28], and lower-layer tunnelling techniques (such as L2TP [29] or Teredo [30]) in order to provide NAT-resistant connectivity in visited networks where native IPv6 is not available.

The HTML5 IE/EE is implemented as an extension to the Polipo web proxy.[9] Its generic request-handling system has been split in order to allow a three-step approach as follows

[7] `http://www.opnet.com`
[8] `http://www-01.ibm.com/software/integration/optimization/cplex-optimizer/`
[9] `http://pps.jussieu.fr/~jch/software/polipo/`

1. Query the IE to obtain object properties for a given URL;
2. Allow a DE to provide decisions about which transport protocol and path (i. e., interface) to use to transfer the object;
3. Establish a connection with the relevant parameters (internal EE).

In order to allow seamless transport protocol agility, this prototype also includes a reverse proxy, which can either be co-located with a specific content-providing server, or at the edge of an NO's domain. The OConS IE would easily be implemented alongside this reverse proxy.

5.3 WP-D Cloud Networking

5.3.1 In-NetDC

Data centre federation across wide area networks has traditionally been tackled in a static manner by leasing costly extra bandwidth or dedicated lines. Overlay approaches such as Akamai and store-and-forward mechanisms (appropriately schedule transfers depending on Internet Service Provider's rate limiting policies) have been proposed as more cost-effective alternatives. However, they rely on fixed resources located on the periphery of ISP networks. This has three major inconveniences: 1) users pay for the resources even when they are not using them; 2) the transfers need to be scheduled depending on the fixed capacity of the resources available; and 3) overlay nodes are usually located at the edge of a network (which implies inefficient hops between the core and the edge of a network).

The In-Network Data Centre (In-NetDC) approach aims to eliminate the aforementioned pitfalls and explores unknown possibilities that may increase the performance of services and applications run by service providers. The concept makes use of the highly over-provisioned links in the core of networks participating in the path. To be more specific, it is used as an alternative location for (portable/non-portable) data centres. This implies that resources can be dynamically initialised within the network in a similar fashion to those done at the edges. This In-NetDC demo uses Common Open Research Emulator (CORE) (developed by Boeing Research) to emulate a simplified version of a realistic topology. A virtual service infrastructure has been implemented together with an OCCI interface so that the environment imitates the actual requests accurately.

Since background traffic accounts for approximately 40% of the inter-data centre traffic, a bulk data transfer is used as an example application. We compared this setup with other files transfer approaches – namely, End-to-End (E2E), random store-and-forward (SnF) and Bit-Torrent (using μTorrent) – under realistic environment assumptions – i.e., rate-limiting policies applied based on the diurnal pattern of each geographical location. We found that In-NetDC performs significantly better than all other methods in both transfer time and transferred volume within 24 hours.

The emulated scenario of bulk data transfer using In-NetDC showed that services and applications can significantly benefit from highly on-demand instantiation of resources from one end to another (or multiple recipients). Nevertheless, the current architecture of the Internet does not satisfy this requirement. A sophisticated mechanism/framework is still needed in order to enable a smooth response across multiple administrative domains (e.g. DC1 $\leftrightarrow$ ISP1 $\leftrightarrow$ ISP2 $\leftrightarrow$ DC2).

Employing Flash Network Slice (FNS) is potentially a practical solution due to its dynamic characteristic and smooth collaboration between providers (e.g. network providers, infrastructure providers) In addition, it also enables a scalable architecture for resource allocation as well as interesting features such as end-to-end isolation (e.g. through VNET and IP filters).

Under the deployment of In-NetDC, we learned that core network locations are better alternatives for file transfers (currently, it has been tested only on bulk data). Although these locations are not typical for network operators, new business model can potentially be an incentive for them to implement the scheme. While placing In-NetDC nodes is beneficial, its economic model and

the effect of exact locations that may yield different performance remain to be discussed. We are currently taking these issues into account.

5.3.2 TPM-based auditing function

The security goal translation function forms the backbone of the CloNe security architecture. It receives security constraints from the different entities involved in the CloNe infrastructure and translates them to concrete resource specifications. For example, a cloud tenant specifies the geographic location where its resources need to be hosted. The location constraint is translated by the security goal translation function and forwarded to the auditing and assurance module. The auditing and assurance module invokes the auditing function, which attests the geographic location of the physical resources provisioned to the cloud tenant. The auditing function facilitates the cloud tenant's policy compliance.

An auditing function prototype using Trusted Platform Module (TPM) [31] is being developed to determine the geographic location of the resources and will be shown separately to the overall CloNe prototype. As testbed we use laptops equipped with TPMs. We install a modified XEN which can access the TPM. In addition, we implement an extension in the VM for accessing the TPM via XEN. To secure the XEN from malicious modifications we attest the components of XEN that are relevant for the access to the TPM.

5.3.3 Hybrid Flash Slice – Customizable Resource allocation and optimization

Current cloud infrastructure management platforms are typically designed to deal with computing and storage resources. When deploying highly distributed applications with strict network requirements, such as low delay or bandwidth guarantees, the support for specification and configuration of such requirements still lacks. Moreover, resource allocation strategies and algorithms are usually hard coded into the cloud platform's core, making it very difficult to improve or adapt these strategies to better fit individual application and environment needs.

In line with the general goals of CloNe, in NEC's *HyFS* prototype we introduce a new approach to cloud platform design emphasizing three main aspects:

1. Robust networking for coupling cloud computing with modern network paradigms (in this case OpenFlow software defined networking),
2. Specification of complex virtual infrastructures, including network topology and application requirements (based on VXDL), and
3. Programmability via an API to ensure customization at the core of the platform's resource allocation and optimization strategies.

Figure 5.2 depicts the conceptual building blocks of our architecture, instantiated as an example over NEC's internal testbed. In the top part of the architecture we placed the humans involved in the processes of request, establishment, and maintenance of virtual infrastructures over clouds. The cloud provider in the architecture is represented by its human *Administrator*. We divided resource management into resource allocation (*Allocation Planner*) and resource adaptation and optimization (*Optimization Planner*). The former is intended to deploy an infrastructure for a given service based on the *Initial Specification*. While the latter adapts this infrastructure to various elasticity parameters (*e.g.* demand fluctuations) also defined in the *Initial Specification*. In short, HyFS allows to manage and control cloud infrastructure like SDN allows to management and control a network.

Our initial prototype [32] was presented at NOMS 2012 in the Student Demo session. In Figure 5.3 you can find a selection of screenshots of the Web GUI of the prototype. The prototype will be extended towards a NEC in-house version of the *Elastic NetInf Deployment* scenario(see

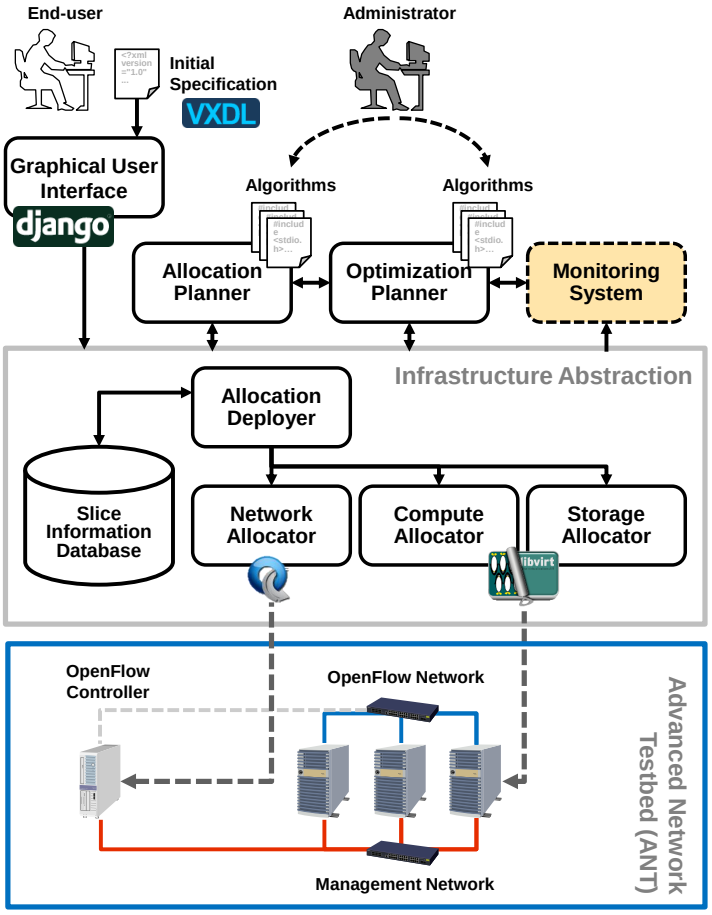


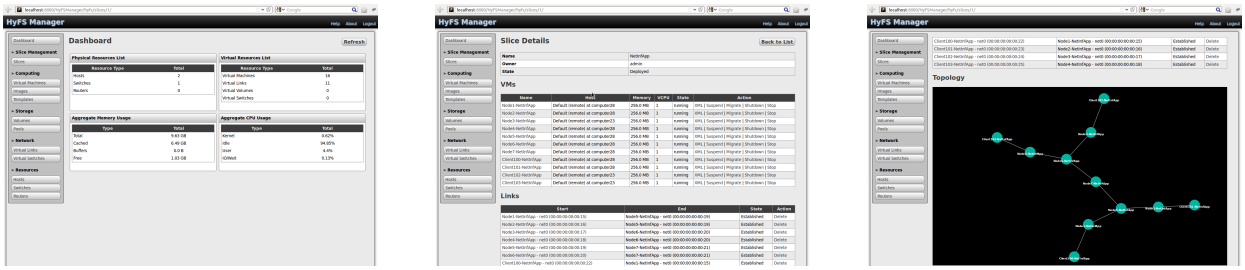
Figure 5.2: HyFS: Prototype Conceptual Architecture Deployed over ANT

Chapter 4). This version will include migration of VMs, but not include inter-provider deployments. We are currently also in the process of publishing a full paper version of this architecture and prototype implementation.

5.3.4 Show the VXDL elasticity to sustain a highly volatile workload

Many applications deployed within clouds have dynamic needs in term of resources. Buzz effects, flash crowds and other types of gossip events may explain the volatility of their workload. Cloud operators may face tough challenges in conveying the proper level of QoS to their hosted applications when these applications exhibit dynamic workload profiles. They may then have crucial needs for efficient schemes to dynamically and rapidly allocate/release resources when an application needs it. Capturing these needs and translating them into action are real challenges for a Cloud Network controller.

In this demo, we demonstrate the VXDL language and CloudWeaver extensions to adequately model and enforce high elasticity within a cloud network so as to cope with the need of a hosted, very dynamic application. The purpose of this demo is twofold. First, we will use our theoretical model (see D.D.1 [16]) to reproduce the workload dynamics of an-demand service that may be subject to buzz effects and flash-crowds types of events. This model exhibits buzz-free period (normal behavior) and buzz period (abnormal behavior) where the instantaneous workload of the e-service surges very sharply. Second, using this model as the workload generator, we will use the VXDL language and CloudWeaver (Cloud Network Controller) extensions to dynamically adjust



(a) Dashboard (b) Slice Management (virtual resources) (c) Slice Management (topology view)

Figure 5.3: HyFS: Screenshots of the Graphical User Interface

the level of provisioned resource that permits to adequately host the application. A new feature in this process stems from the use of large deviation principle that will assist the decision making process for allocating or releasing resources.

The Use Case will be centred on a Video-on-Demand (VoD) service. A server broadcasts video to a huge number of potential watchers. As the number of watchers varies, so does the workload on the VoD service (we assume no multicast transfer here).

This demo will be deployed on the Grid'5000 platform. Agents acting as the VoD server or as potential watchers will be deployed on geographically distant nodes. The popularity of a video will be directly linked to the number of agents who have "seen" it in the past. On top of this, buzz (flash crowd) caused by an external event may occur and cause a sudden surge in the popularity of a video. In the demo, we will be able to directly trigger buzz. Overall, the subsequent workload exhibit large and sometimes steep variations in its behavior.

At the VoD server, we will collect measurements data and we will use this information to better adjust the level of resource dedicated to this application. Of course, we will consider a QoS objective but we will also try to minimize to number of reconfiguration/reallocation, viz. CAPEX vs OPEX. Our allocation scheme will be mainly based on the large deviation properties. We will rely on the virtual description language to dynamically provision and release resources to match the current need.

The implementation is currently ongoing (in C) and should be deployed on the Grid'5000 during the Summer. We are targeting to obtain demonstrable results during Fall.

5.3.5 Authorisation logic to support delegation across providers

Security is widely acknowledged to be a major barrier to the adoption of cloud computing by enterprises. A critical aspect of security is authorization to control access to services. Cloud computing is characterized by autonomous, loosely coupled services. Authorizations need to be generated and consumed dynamically and autonomously. Therefore solutions assuming a single central authority, in which attestation and authorizations are controlled by human administrators, will not work.

CloNe is such a cloud computing service (IaaS), using multi-tenancy and the delegation principle to manage virtual infrastructures spanning multiple providers. A tenant (or user associated with a tenant) requests virtual infrastructure from a provider, implicitly authorizing the provider to manage the infrastructure on his behalf. The provider can repeat the delegation, creating a delegation chain and so an authorization chain. At the same time, the tenant retains the authority to determine who (which users and other tenants) are allowed to access his virtual infrastructure as he perceives it. One particular feature of the CloNe architecture is the need for providers to interact to coordinate management actions, for example, to connect a flash network slice across

administrative boundaries.

HP Labs has developed an authorization logic that supports delegation and does not rely on a single root of authority. This prototype demonstrates the use of automatic authorization policy generation and enforcement in CloNe. For simplicity the prototype uses a single authorization policy service comprising a policy database and logic checking engine implemented in Java. This can be accessed using either a REST interface or a JMS messaging interface. A client-side API has been implemented to give the providers a set of methods for registering relationships among their information models that are encoded as policy.

Using this arrangement the infrastructure providers are able to perform entirely local actions, such as identifying that the role of one object has been delegated to another, or that two objects are expected to be able to link to each other. They can also ask questions such as can a given user inspect or link to a given object. The authorization logic engine in the server deals with interpreting these actions in a global context.

The authorization server has been implemented and is being integrated with a mock provider in the HP infrastructure. We are building out the client-side abstractions to provide fully automatic distributed authorization. We will investigate the possibility of integration with the dynamic enterprise prototype.

In principle there is no reason to centralise the logic processing as we have done in this case. Other options are to distribute sufficient logic to providers to allow them to use a local version of the service, or to distribute the processing, so that the logic processing runs across multiple provider's authorization services.

6 Integration and cooperation plan

For each of the scenarios described in the previous chapters, we present a high-level integration plan and ways of working. Without going in full detail in the current document, we highlight the key milestones that should give the readers an overview of how the final demonstration and prototyping goals are achieved by the end of the project.

Since the project-internal prototyping and demonstration workshop in Lisbon (Jan.2012), intensive cross work package discussions (also in face-to-face meetings of WP-B/WP-C and WP-D/WP-C) have taken place to achieve this aligned and commonly agreed view. In the current deliverable, we document the state of project-wide cooperation plans as of May 2012.

When project-wide activities need further alignment across the work packages, this will be done in the ongoing teleconference and physical meetings of the *Theme Prototyping and Experimentation* established in the project organization.

6.1 Event with Large Crowd

The following steps (milestones) are planned towards the end of the project, in order to realize the prototyping activities as sketched in Chapter 2.

- June/July 2012: presentation and discussion with WP-B on detailed mechanisms adopted by OConS to manipulate the NetInf forwarding mechanisms and the convergence layer to perform the multi-path content retrieval/delivery.
- NetInf–OConS face-to-face meeting during the SAIL General Project Meeting, Bristol, week of 17 Sept. 2012: finalized descriptions of inter-NetInf–OConS architecture, interfaces and data models. Align use case realizations across WPs.
- OConS with NetInf in a public demonstration at the *MONAMI Conference*, Hamburg, September 2012:
- Jan. 2013: document CloNe–OConS demonstration/prototyping realization in deliverable D.C.5 *Demonstrator for Connectivity Service* and the overall project-wide architecture, interfaces etc. in D.A.3 *Final Harmonised SAIL Architecture*.
- Final NetInf–OConS demonstration at *Future Internet Assembly (FIA)* in Dublin, February 2013, preferably for project review purposes.

6.2 Dynamic Enterprise

The following steps (milestones) are planned by end of the project, in order to realize the prototyping activities as sketched in Chapter 3:

- *Dynamic Enterprise* public demonstration at the *Future Networks and Mobile Summit*, Berlin, 4-6 July 2012: concept of inter-CloNe–OConS architecture, interfaces and data models (extended OCNI).

- July 2012: alignment of WP-C deliverable D.C.3 *OConS - Demonstrator Specification and Integration Plan* and WP-D deliverable D.D.2 *CloNe - Description of Implemented Prototype*
- CloNe–OConS face-to-face meeting and code sprint session during the SAIL General Project Meeting, Bristol, week of 17 September 2012: finalized descriptions of inter-CloNe–OConS architecture, interfaces and data models (extended OCNI). Align use case realizations across WPs.
- Final CloNe–OConS demonstration at e.g. final SAIL General Project Meeting, January 2013, preferably for project review purposes.
- Jan. 2013: Document CloNe–OConS demonstration/prototyping realization in deliverable D.C.5 *Demonstrator for Connectivity Service* and the overall project-wide architecture, interfaces etc. in D.A.3 *Final Harmonised SAIL Architecture*.

6.3 Elastic NetInf deployment

The following steps (milestones) are planned towards the end of the project, in order to realize the prototyping activities as sketched in Chapter 4:

- Ongoing and to be continued until end of project: Bi-weekly telephone conference calls to coordinate progress and track status.
- June/July 2012: Organize a hands-on code-sprint event
- September 2012: First version of demo at the Bristol General Project Meeting. Organize Face-to-Face meeting there, discuss next step and further modifications. Maybe identify additional demo scenarios to implement.
- October–December 2012: Additional code sprint events will be scheduled if necessary.
- End of project: Final demo at review.
- Contributions to Deliverables D.A.3 *Final Harmonised SAIL Architecture* and D.D.3 *CloNe Refined Architecture*.

7 Conclusion

We have presented three prototype demonstration scenarios. They were derived from the overall technical use case and scenarios that were established at the beginning of the project. The demonstrated scenarios have been selected because they illustrated well typical problems that are addressed by SAIL. They also require the collaboration from more than one WP, showing the benefit of integrating the proposed solutions. For each of the selected scenarios, we have establish a testbed setup and proposed a storyboard to demonstrate and validate the features and advantages of the solutions put in place during the project. We described the prototype architecture and presented the main components required to realised the demonstration. We have established criterias for the evaluation of the achievements.

The *Event with Large Crowd* scenario which focuses around NetInf illustrates how mobile users during planned or impromptu large gathering of people benefits from the NetInf solutions with the support OConS mechanisms. It aims at demonstrating the general feasibility of the NetInf approach, the interoperability of the components developed by the different partners during the SAIL project and the specific benefits of the NetInf system for content distribution. The experimentation will be two-fold. First, it will use real devices to interact with the NetInf components developed by the SAIL partners to show the feasibility and interoperability of the solutions. Second, it will use an emulator to study scalability of the solution and establish comparison with the current internet protocols used for content delivery. Multi-path routing for ICN is expected to be presented at the MONAMI conference in Hamburg in September 2012 while the full *Event with Large Crowd* scenario is expected to be showcased at the Dublin FIA in February 2013.

The *Dynamic Enterprise* scenario shows how Virtual Infrastructure (VI) for running applications can be dynamically deployed over multiple infrastructure service provider (data centre or network) domains taking advantage of the CloNe features. In collaboration with OConS, it also showcases elasticity of the flash network slice. The main testbed is deployed over multiple partner premises and uses different instantiations of cloud OS and network technologies. This allows to validate the applicability of the concepts and constructs used for the establishment of VI in different cloud provider environments. The *Dynamic Enterprise* scenario will be demonstrated at the Berlin Future Network & Mobile Summit (FuNeMS) in July 2012.

The *Elastic NetInf Deployment* illustrates how NetInf can be deployed as an overlay on top of the Internet taking advantage of CloNe. This scenario also provides the required context to validate the CloNe interface from a NetInf management perspective. It illustrates how NetInf nodes can be deployed or removed on-demand from the network depending on the current load for content distribution. This scenario provides a viable migration path for NetInf and allows NetInf to coexist with other Internet overlay approaches. We intend to present the *Elastic NetInf Deployment* in a project event (final project general meeting or end-of project review) early 2013.

We also described other prototyping activities that occurred during the project in complement of the three demonstration scenarios. These activities were targeting at specific architectural problems or were used as preliminary work for the integrated prototypes.

Not all details were given for each prototype scenario and components as more details are already (or will be) included in other project deliverables. The components of the NetInf *Event with Large Crowd* are described in D.B.2 [6], OConS components and integration points are described in D.C.3 [4] and CloNe implemented components and integration will be reported in D.D.2 [5].

List of Acronyms

AIMD	Additive Increase/Multiplicative Decrease
AR	Access Router
BPQ	Bundle Protocol Query
CMBS	Cloud Message Brokering Service
CE	Customer Edge
CL	Convergence Layer
CC	Cloud Controller
CloNe	Cloud Networking
CORE	Common Open Research Emulator
CRUD	Create, Read, Update, Delete
DCC	Domain Control Client
DCMP	Distributed Cloud Management Protocol
DCM	Distributed Cloud Manager
DCP	Distributed Cloud Protocol
DCU	Domain Control Unit
DC	Data centre
DE	Decision Making Entity
DTN	Disruption/Delay Tolerant Network
EE	Execution and Enforcement Entity
EwLC	Event with Large Crowd
FIA	Future Internet Assembly
FNS	Flash Network Slice
FuNeMS	Future Network & Mobile Summit
GIN	Global Information Network
GINP	GIN Protocol
GUI	Graphical User Interface

HTML Hypertext Markup Language

HTTP Hypertext Transport Protocol

HURRy HUman Routines used for Routing

IaaS Infrastructure as a Service

ICN Information Centric Networking

IE Information Management Entity

KPI Key Performance Indicator

L-CDN Localized-CDN

L2TP Layer-2 Tunneling Protocol

LXC Linux Container

LNP Link Negotiation Protocol

MDHT Multi-level Distributed Hash Table

MN Mobile Node

NAT Network Address Translation

NDO Named Data Object

NetInf Network of Information

NHT Next Hop Table

NNRP NEC NetInf Router Platform

NO Network Operator

NRS Name Resolution System

NW Network

OCCI Open Cloud Computing Interface

OCNI Open Cloud Networking Interface

OConS Open Connectivity Services

OConS Open Connectivity Services

OR Orchestration Register

OS Operating System

PCE Path Computation Entity

PE Provider Edge

PRoPHET Probabilistic Routing Protocol based on Historical EncounTers

QoS	Quality of Service
RAN	Radio Access Network
REX	Resolution Exchange System
RVS	RendezVous Server
RTT	Round-trip Time
SAIL	Scalable Adaptive Internet Solutions
SOP	Service Orchestration Process
TCD	Trinity College Dublin
TED	Traffic Engineering Database
TPM	Trusted Platform Module
UPB	University of Paderborn
URL	Universal Resource Locator
VI	Virtual Infrastructure
VM	Virtual Machine
VN	Virtual Network
VxDL	Virtual infrastructure Description Language
WAN	wide area network
WP	work package

List of Figures

Bibliography

[1] The SAIL Consortium. Description of Project-wide Scenarios and Use Cases. Deliverable FP7-ICT-2009-5-257448-SAIL/D.A.1, SAIL project, February 2011. Available online from `http://www.sail-project.eu`.

[2] The SAIL Consortium. The sail project web site. `http://www.sail-project.eu/`.

[3] The SAIL Consortium. The Network of Information - Prototyping and Evaluation. Deliverable FP7-ICT-2009-5-257448-SAIL/D.B.4, SAIL project, January 2013. Will be available online from `http://www.sail-project.eu`.

[4] The SAIL Consortium. OConS - Demonstrator Specification and Integration Plan . Deliverable FP7-ICT-2009-5-257448-SAIL/D.C.3, SAIL project, April 2012. Available online from `http://www.sail-project.eu`.

[5] The SAIL Consortium. CloNe - Description of Implemented Prototype. Deliverable FP7-ICT-2009-5-257448-SAIL/D.D.2, SAIL project, July 2012. Will be available online from `http://www.sail-project.eu`.

[6] The SAIL Consortium. The Network of Information - Content delivery and operations. Deliverable FP7-ICT-2009-5-257448-SAIL/D.B.2, SAIL project, May 2012. Will be available online from `http://www.sail-project.eu`.

[7] E. Davies A. Lindgren, A. Doria and S. Grasic. Probabilistic routing protocol for intermittently connected networks. Active Internet-Draft, May 2012.

[8] J. M. Cabero, V. Molina, I. Urteaga, F. Liberal, and J. L. Martin. Acquisition of human traces with bluetooth technology: Challenges and proposals. *Ad Hoc Networks*, Accepted. To appear.

[9] S. Perez-Sanchez, N. Errondosoro, J. M. Cabero, I. Urteaga, and I. Olabarrieta. Human routines optimise routing in disrupted networks: the hurry protocol. *ACM MobiCom Workshop on Challenged Networks CHANTS 2012*, Submitted.

[10] The SAIL Consortium. The Network of Information - Architecture and Applications. Deliverable FP7-ICT-2009-5-257448-SAIL/D.B.1, SAIL project, July 2011. Available online from `http://www.sail-project.eu`.

[11] D. Kutcher S. Farrell, A. Lynch and A. Lindgren. Bundle protocol query extension block. Active Internet-Draft, 2012.

[12] Communication System Design (KTH, Stockholm). Android application version 3 of the project bytewalla, 2010.

[13] The SAIL Consortium. Final harmonised SAIL architecture. Deliverable FP7-ICT-2009-5-257448-SAIL/D.A.3, SAIL project, January 2013. Will be available online from `http://www.sail-project.eu`.

[14] The SAIL Consortium. Architectural Concepts of Connectivity Services. Deliverable FP7-ICT-2009-5-257448-SAIL/D.C.1, SAIL project, July 2011. Available online from `http://www.sail-project.eu`.

[15] The SAIL Consortium. Architectural Concepts of Connectivity Services - Addendum. Deliverable FP7-ICT-2009-5-257448-SAIL/D.C.1 Addendum, SAIL project, January 2012. Available online from `http://www.sail-project.eu`.

[16] The SAIL Consortium. Cloud Networking Architecture Description. Deliverable FP7-ICT-2009-5-257448-SAIL/D.D.1 Rev. 2.0, SAIL project, January 2012. Available online from `http://www.sail-project.eu`.

[17] pyOCNI - a Python implementation of an extended OCCI with a JSON serialization and a cloud networking extension. Online URL: http://occi-wg.org/2012/02/20/occi-pyocni/.

[18] Lars Brown and Erik Axelsson. Use of information-centric networks in revision control systems. Master's thesis, Royal Institute of Technology (KTH), January 2011.

[19] Bengt Ahlgren, Börje Ohlman, Erik Axelsson, and Lars Brown. Subversion over OpenNetInf and CCNx. In *4th International Workshop on Architectures, Services and Applications for the Next Generation Internet (WASA-NGI-IV)*, Bonn, Germany, October 4, 2011. In conjunction with IEEE LCN.

[20] S. Farrell, A. Lynch, D. Kutscher, and A. Lindgren. Bundle Protocol Query Extension Block. Internet-Draft draft-irtf-dtnrg-bpq-00, Internet Engineering Task Force, May 2012. Work in progress.

[21] K. Scott and S. Burleigh. Bundle Protocol Specification. RFC 5050 (Experimental), November 2007.

[22] Anders Lindgren and Mahesh Bogadi Shankar Prasad. Dtvideo: Information-centric dtn video distribution. In *HotMobile 2012*, February 28-29 2012.

[23] Matteo D'Ambrosio, Paolo Fasano, Mario Ullio, and Vinicio Vercellone. The global information network architecture. Technical Report TTGTDDNI1200009, Telecom Italia, 2012.

[24] Xi Li, Olivier Mehani, Ramón Agüero, Roksana Boreli, Yasir Zaki, and Umar Toseef. Evaluating user-centric multihomed flow management for mobile devices in simulated heterogeneous networks. Under review, May 2012.

[25] Olivier Mehani, Roksana Boreli, Michael Maher, and Thierry Ernst. User- and application-centric multihomed flow management. In Tom Pfeifer and Anura Jayasumana, editors, *LCN 2011, 36th IEEE Conference on Local Computer Networks*, pages 26–34, Los Alamitos, CA, USA, October 2011. IEEE Computer Society, IEEE Computer Society.

[26] Estimating end-to-end performance in IP networks for data applications. Recommendation G.1030, ITU-T SG12, Geneva, Switzerland, May 2006.

[27] David Johnson, Charles E. Perkins, and Jari Arkko. Mobility support in IPv6. RFC 3775, RFC Editor, Fremont, CA, USA, June 2004.

[28] Ryuji Wakikawa, Vijay Devarapalli, George Tsirtsis, Thierry Ernst, and Kenichi Nagami. Multiple care-of addresses registration. RFC 5648, RFC Editor, Fremont, CA, USA, October 2009.

[29] W. Mark Townsley, Andrew J. Valencia, Allan Rubens, Gurdeep S. Pall, Glen Zorn, and Bill Palter. Layer two tunneling protocol. RFC 2661, RFC Editor, Fremont, CA, USA, August 1999.

[30] Christian Huitema. Teredo: Tunneling IPv6 over UDP through network address translations (NATs). RFC 4380, RFC Editor, Fremont, CA, USA, February 2006.

[31] Dan Wang and Dengguo Feng. A hypervisor-based secure storage scheme. In *Networks Security Wireless Communications and Trusted Computing (NSWCTC), 2010 Second International Conference on*, volume 1, pages 81 –86, april 2010.

[32] Juliano Araujo Wickboldt, Lisandro Zambenedetti Granville, Dominique Dudkowski, and Marcus Brunner. Hyfs manager: A hybrid flash slice manager. In *Presented in Student Demo Contest of the 13th IEEE/IFIP Network Operations and Management Symposium (NOMS 2012)*, April 2012.