Objective FP7-ICT-2009-5-257448/D-4.3

Future Networks

Project 257448

"SAIL – Scalable and Adaptable Internet Solutions"

D-4.3
(D.C.3) Demonstrator Specification and Integration Plan

Date of preparation: **2012-08-15**
Start date of Project: **2010-08-01**
Project Coordinator: **Thomas Edwall**
Ericsson AB

Revision: **1.1**
Duration: **2013-01-31**

Document Properties

Document Number:	D-4.3
Document Title:	**(D.C.3) Demonstrator Specification and Integration Plan**
Document Responsible:	Michael Soellner (ALUD)
Document Editor:	Michael Soellner (ALUD)
Authors:	Ramón Agüero (UC), Pedro A. Aranda Gutiérrez (Telefónica I+D), Luis Fco. Díez (UC) Marta Garcia (UC), Olivier Mehani (NICTA), Susana Perez (Tecnalia), Peter Schefczik (ALUD), Michael Soellner (ALUD), Lucian Suciu (FT-Orange), Asanga Udugama (UHB)
Target Dissemination Level:	PU
Status of the Document:	Final
Version:	1.1

Production Properties:

Reviewers:	Marcus Brunner (NEC) Stephen Farrell (TCD)	Luis M. Correia (IST) Benoit Tremblay (EAB)

Document History:

Revision	Date	Issued by	Description
1.0	2012-05-23	Michael Soellner	Final Version
1.1	2012-08-15	Michael Soellner	Change confidentiality status, fixed a few typos

Disclaimer:

Document:	FP7-ICT-2009-5-257448-SAIL/D-4.3		
Date:	August 15, 2012	Security:	Public
Status:	Final	Version:	1.1

Abstract:

This document is a public deliverable of the EU-FP7 project SAIL (Scalable Adaptive Internet Solutions, [1]) and describes the demonstrator activities and prototype integration plan for the SAIL workpackage 'Open Connectivity Service (OConS)' with the target of final realization at project end. Moreover, details of cross-WP cooperation are given from an OConS perspective in addition to the project-wide description in the forthcoming deliverable D.A.9 "Description of overall prototyping use cases, scenarios and integration points".

Keywords:

open connectivity services, prototyping, demonstration, cloud networking, network of information, open-flow, dtn, icn, multi-path, distributed connectivity control, distributed mobility management

Executive Summary

This document is a public deliverable of the EU-FP7 project SAIL (Scalable Adaptive Internet Solutions, cf. [1]) and describes the demonstrator activities and prototype integration plan for the SAIL workpackage '*Open Connectivity Service (OConS)*' with the target of final realization at project end. Moreover, details of cross-WP cooperation (with workpackage (WP) '*Cloud Networking (CloNe)*' and with WP '*Network of Information (NetInf)*') are given from a OConS perspective in addition to the project-wide description in the forthcoming deliverable D.A.9 "Description of overall prototyping use cases, scenarios and integration points".

The practical evaluation work in WP OConS started with the design and realization of a first experimental project phase (Phase1) which resulted in intermediate demonstration show cases. These results were presented and discussed at a project-internal workshop in January 2012.

Based on these components and the gained experience, this document now defines the further step of OConS prototype and demonstration activities which focuses in a Phase2 on the updated use case scenarios (from D.C.1-Addendum [2]), the workpackage internal cooperation based on the progress of the OConS architectural framework, and the cross-WP cooperation with CloNe and NetInf.

For the final OConS-CloNe interaction, we focus on developing an "elastic video networking" use case scenario, whereas the OConS-NetInf interaction deals with multi-path connectivity services for Information-Centric Networks (ICN)-like content delivery, as well as improving Delay Tolerant Network (DTN) connectivity and routing.

The plans given here are aiming at the final realizations at the project end.

Contents

1 Introduction

This document is a public deliverable of the EU-FP7 project SAIL (Scalable Adaptive Internet Solutions, cf. [1]) and describes the demonstrator activities and prototype integration plan for the SAIL workpackage '*Open Connectivity Service (OConS)*' with the target of final realization at project end. Moreover, details of cross-WP cooperation (with workpackage (WP) '*Cloud Networking (CloNe)*' and with WP '*Network of Information (NetInf)*') are given from a OConS perspective in addition to the project-wide description in the forthcoming deliverable D.A.9 "Description of overall prototyping use cases, scenarios and integration points".

The SAIL project aims at compelling, prototype-based evidence that its research results and innovations will inspire the Network of the Future. To achieve this by the end of the project, we use a more agile approach than usually done in a research project. In a first phase (until project month 18), the prototyping activities have early started from a bottom-up perspective, both at partner and WP levels, with minimal overall planning and cross-task specification work. This allows an immediate transfer of experience gained in early phases to the second project phase (month 19-30).

The practical evaluation work in WP OConS started with the design and realization of a first experimental project phase (Phase1), driven by early demonstration and experimentation activities of the partners accompanying the OConS technical framework definition, which resulted in intermediate demonstration show cases. These results were presented at the project-internal workshop in January 2012 (project month 18).

Based on these components and the gained experience, this document now defines the further step of OConS prototype and demonstration activities which focuses in a Phase2 (project month 19-30) on the updated use case scenarios (from D.C.1-Addendum [2]), the workpackage internal cooperation based on the progress of the OConS architectural framework, and the cross-WP cooperation with CloNe and NetInf.

The plans given here are aiming at the final realizations at the project end.

The document is structured as follows. We continue the introduction with a recapitulation of the initial clustering of prototyping and experimentation activities as defined at project start. This view was based on the project-wide use case scenarios of D.A.1 [3] and their WP-specific versions in D.C.1 [4]. Then we describe how the use cases and technical focus of OConS as a starting point of Phase2 demonstration activities have been revised due to the more mature discussions and updates contained in D.C.1-Addendum [2].

Chapter 2 gives a technical description of the OConS prototype building blocks in terms of motivation, functionality, component architectures, realization details, interfaces and services, and demonstration show-cases. This is mainly based on the realization experience for the Lisbon workshop activities in January 2012, however also paves the way for the extention of the components by the project end.

A deeper analysis of the updated OConS scenarios and use cases of the D.C.1-Addendum is given in chapter 3. For the purpose of deriving the essential OConS services to practically support and demonstrate OConS aspects, we focus on the "mobile access and datacentre interconnection use case" for supporting CloNe and the "mobile and multi-P for information centric networks use case" for supporting NetInf. The cross-WP considerations are based on the documents D.B.1 [5], D.B.2 [6] for NetInf, and the architecture document D.D.1 [7] for CloNe.

The results of the OConS analysis and the discussions with the workpackages CloNe and NetInf with respect to a final OConS demonstrator design are described in chapter 4. OConS approaches requiring support by prototyping activities are the generic connectivity mechanisms, architectural decomposition into Information Management Entitys (IEs), Decision Making Entitys (DEs) and Execution and Enforcement Entitys (EEs), and the OConS orchestration mechanism.

Finally, chapter 5 will give the final plans for integration and cross-WP cooperation together with the cross-WP integration points, as agreed in the cross-WP meetings.

1.1 Phase 1: OConS Initial Prototyping and Demonstration Activities until Lisbon Workshop

At project start, OConS identified a few basic use cases that gave the orientation for the first phase of (bottom-up) prototyping activities.

1. **UseCase#C.1: Wireless challenged networks**

   A heterogeneous set of wireless nodes is willing to build a network to provide to end-users connectivity between them and towards a fixed Internet infrastructure in an adverse environment. Support extra-low-bandwidth and delay-tolerant services (e.g. SMS, Facebook updates, E-mails) which shall satisfy the communication needs of the majority of the people.

2. **UseCase#C.2: Use of multi-path multi-protocol transport to optimize services with heterogeneous content**

   The heterogeneous services could include: e.g., uploading videos of the event or accident, downloading the same video or news about the same event (both people on site and others may be interested in the content which is uploaded); chat, VoIP or other applications may also used simultaneously. To reflect the technical challenges which arise from the use of MultiP techniques: congestion control, multi-cast trees, etc

3. **UseCase#C.3: Optimize QoE for end user, cloud and operator services**

   There are a number of available networks and a number of services provided to end users. E.g. 3G/anyG, mobile WiMax, Wi-Fi infrastructure mode. A self organised community mesh network in a pedestrian / public space location could complement the available communication means. Finally, ad-hoc networks e.g. using Wi-Fi Direct may also be available. Optional technology or power failure may disable one or more technologies.

4. **UseCase#C.4: (Autonomous) Interoperation and connectivity of Cloud and Netinf datacentres**

   Shows how end-user needs/actions create a demand for connectivity between distributed Cloud and Netinf data/service centres (acting as aggregators for a group of end users on the core infrastructure side), which is not necessarily triggered directly by end user actions. OConS functional entities themselves (e.g., mobility mgmt., resource mgmt., and so on) can be placed and moved depending on various topological or resource constraints.

These use cases were supported by the following clusters of prototyping activities:

1. Cluster: Challenged wireless environment (UseCase#C.1)
   - Distributed routing and forwarding protocol for Delay/Disruptive Tolerant Networks:

  - based on self-learning and self-management techniques from each node's dynamic environment (for mobile resource-constraint devices)

2. Cluster: End-to-end multipath transport (UseCase#C.2)
   - Multi-path support for content delivery in information centric networks:
     - transport mechanisms in wireless and wired network entities using multiple paths to carry content to required locations
     - extending current ICN-like implementation with multipath capabilities
   - End-to-end multi-path management for mobile transport (UseCase#C.2 and C.3):
     - multi-path transport protocols (incl. multi-congestion control) adapting to media content type for advanced apps e.g. web using HTML5
     - mobile device decision making for congestion control, reliability, interface and wireless service provider (to maximise use of available resources and QoE)
3. Cluster: Mobile access networks (UseCase#C.3)
   - Smart resource management for mobile OConS access:
     - implementation of a subset of the OConS architecture, interfaces and signaling (with more emphasis at the access part)
     - smart selection of mobile access - higher layers
   - Dynamic distributed mobility management:
     - distributed mobility management scheme with dynamically allocated mobility anchoring in access routers
     - optimized mobile per-flow connectivity and forwarding functions related to the flash crowd scenario
4. Cluster: datacentre interconnect between edge/core networks (UseCase#C.4)
   - Interconnectivity of service datacentres:
     - multi-layer control and protocols for path and flow optimization between distributed datacentres (CloNe or NetInf nodes)
     - client-to-network and cross-layer interaction of IP/MPLS routing and optical switching
     - metro/core functionality for management and control of virtualized networks between DCs
   - Inter-provider connectivity of Service datacentres:
     - multi-layer control and protocols for path and flow optimization between distributed datacentres based on OpenFlow concepts
     - focus on inter-provider interfaces and protocols in OconS

1.2 Phase 2: Towards the Project-wide Flash Crowd Scenario

The scope of OConS connectivity mechanisms range from access to core networks. All OConS mechanisms described in D.C.1 [4] can be orchestrated with each other to provide OConS connectivity. Some of them can be used alternatively or optionally, but due to their common architecture and interfaces they can be orchestrated as required.

As examples of connectivity services to support innovative concepts such as CloNe and NetInf, we focus on two use cases related with these, within the overall SAIL flash crowd scenario as presented in D.A.1 [3]. Based on the requirements from CloNe/NetInf and available network resources, a combination of OConS mechanisms is proposed to improve connectivity, showing the deployment and the instantiation of OConS entities, mechanisms and control functions.

In the flash crowd scenario, depicted in Figure 1.1, a large number of users concurrently and suddenly (e.g. due to a particular event) ask for connectivity and, in general, for network services, in the very same geographical region. Multiple network operators might be available in that area, each one deploying radio access networks with different technologies, e.g. 3G, LTE, WLAN, including mesh and relays. Moreover, users are equipped with different terminals, and of course they are mobile, i.e. they may frequently change their point of attachment. Finally, users access different services, e.g. real-time services which download content or upload on-spot generated data. Several OConS services can be therefore deployed on appropriate actors from this scenario to cope with the extreme dynamicity, to provide enhanced end-user experience, and to optimize network resource usage.

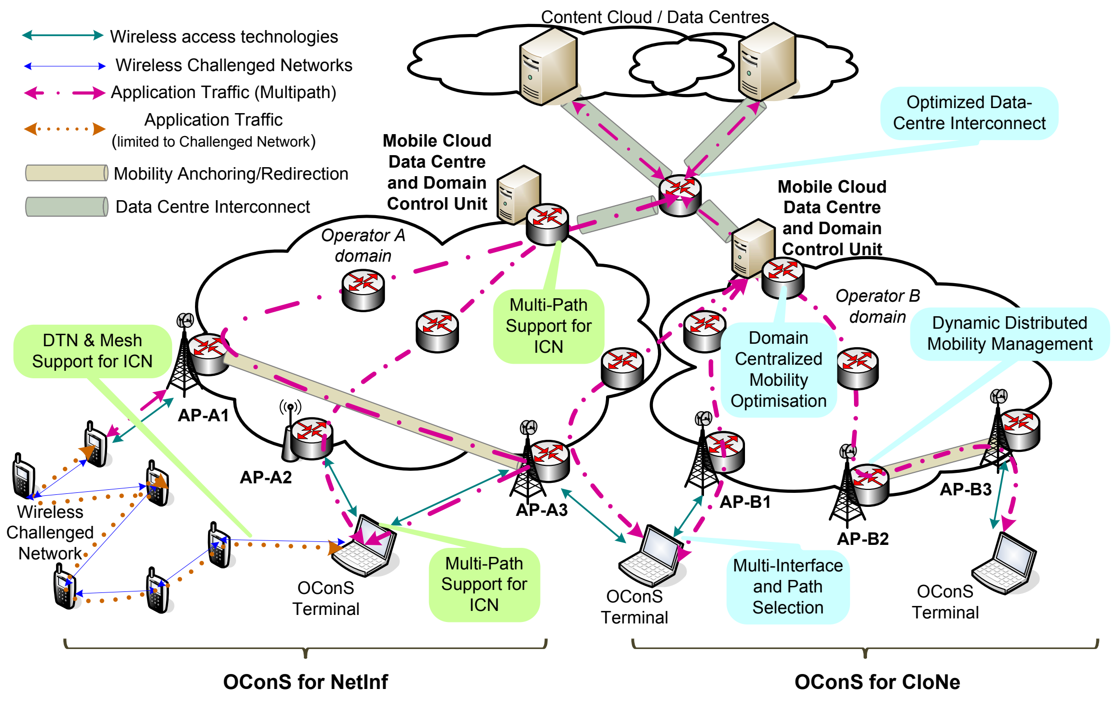

Wireless access technologies
Wireless Challenged Networks
Application Traffic (Multipath)
Application Traffic (limited to Challenged Network)
Mobility Anchoring/Redirection
Data Centre Interconnect
Content Cloud / Data Centres
Optimized Data-Centre Interconnect
Mobile Cloud Data Centre and Domain Control Unit
Mobile Cloud Data Centre and Domain Control Unit
Operator A domain
Operator B domain
Multi-Path Support for ICN
DTN & Mesh Support for ICN
Domain Centralized Mobility Optimisation
Dynamic Distributed Mobility Management
AP-A1
AP-A2
AP-A3
AP-B1
AP-B2
AP-B3
Wireless Challenged Network
OConS Terminal
Multi-Path Support for ICN
OConS Terminal
Multi-Interface and Path Selection
OConS Terminal
OConS for NetInf
OConS for CloNe

Figure 1.1: Flash Crowd scenario, illustrated with selected OConS services for two use cases

1.3 OConS for CloNe: Mobile Access and Datacentre Interconnection Use Case

This use case specifically shows how the OConS provides enhanced connectivity services for Cloud Networking (CloNe), both for enhanced wireless accesses (within a heterogeneous environment), as well as for the core datacentre interconnection.

The mobile flash crowd scenario is characterised with spatial and temporal distributions that are largely not predictable. The management solutions of this scenario necessitate continuous monitoring, flexible interacting decisions and enforced realization of resource allocations, which exceed the capabilities of currently locally-acting autonomous mechanisms.

Furthermore, in this mobile flash crowd use case, we assume that there are resource short-comings at the wireless interfaces (due to overcrowded spectrum and mobility management entities) as well as at the access to popular community and content distribution services (represented by their cloud gateways and datacentres), which require a strict management of all available resources between both ends.

The OConS framework is applied to solve that challenge between different domains (wireless providers, content/social community providers) at different connectivity levels (wireless access vs. core connectivity) and in cooperation with CloNe management of cloud resources. In each of the involved domains, OConS information elements are used to monitor and collect information on available paths between clients and servers of their domain, as well as information on congestion of the network and path availability.

As soon as the flash crowd happens, the OConS framework has to react on the sudden increase of users requesting the very same premium content. Thus, the OConS framework can orchestrate appropriate OConS services. This is decided by the DEs (possibly hosted on a domain-specific OConS Control Unit) and enforced by the EEs implemented on servers, routers, switches, end-terminals. As depicted in Figure 1.2, the orchestration of the OConS services considers the following actions:

- Enable multi-path transport services on the servers in the datacentre, to leverage all the paths between clients and servers, in particular exploiting the least congested ones (see D.C.1 Section 5.2.1);
- Create new paths towards other instances of the same server, located in other datacentres, communicating with DEs possibly in other domains, via OConS signalling (see D.C.1 Section 5.2.3);
- Facilitate the handover of some users from congested to non-congested access networks by interfacing with the DEs of the mobility management (see D.C.1 Sections 6.1.1 to 6.1.4), where different users will be handled depending on their capabilities and technology;
- Set policies in user-terminals with multiple interfaces, enabling the simultaneous use of two or more interfaces for load balancing purposes (see D.C.1 Section 5.2.7).

This example shows that coordination and execution of multiple OConS services can reduce the congestion in different access networks and provide better Quality of Experience (QoE) for the flash crowd users. Local replicas of the same server with improved connectivity can be made available to users, which can be seen as part of an overall "cloud management" that interfaces with CloNe.

1.4 OConS for NetInf: Mobile and Multi-P for Information Centric Networks Use Case

In this use case we illustrate how OConS provides connectivity services to Information-Centric Networks (ICN) (in particular to NetInf), creating and sustaining the connectivity in challenged wireless networks while using multi-path enhancements. We start with the assumption that there is a street performer and someone likes his/her performance very much, so he/she spreads the word through social networking, and a flash crowd is spontaneously gathered. Some people record and upload the event to a social network (e.g. Facebook), spread the word about the video around and

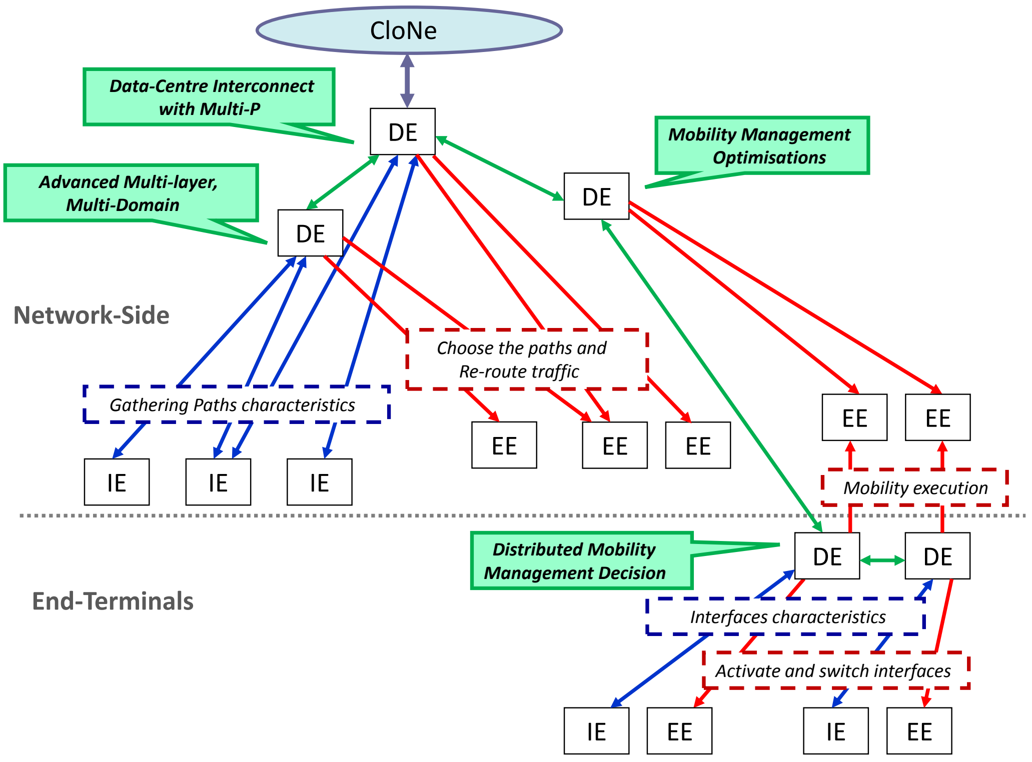

CloNe
Data-Centre Interconnect with Multi-P
DE
Mobility Management Optimisations
DE
Advanced Multi-layer, Multi-Domain
DE
Network-Side
Choose the paths and Re-route traffic
Gathering Paths characteristics
IE
IE
IE
EE
EE
EE
EE
EE
Mobility execution
Distributed Mobility Management Decision
DE
DE
End-Terminals
Interfaces characteristics
Activate and switch interfaces
IE
EE
IE
EE

Figure 1.2: Mobile Access and Datacentre Interconnection Use Case

the flash crowd and other followers of the social network start to download the video, as well as background information about the artist.

The users employ NetInf nodes caching, and forward the produced content based on their name and locator. In this flash crowd some users have good connectivity, while others experience poor or intermittent connections. OConS services can improve the connectivity of the flash crowd users, which require a minimum reliability, a minimum QoE, and therefore, necessitate optimized resource utilization. More specifically, the message forwarding functionality in each NetInf node needs to be enhanced by OConS. Thus by collecting network information via the OConS IEs, the DE in the OConS enhanced NetInf message forwarding function can decide to activate and configure the OConS mechanisms using the appropriate EEs.

In this use case the following OConS functions are considered (Figure 1.3):

- OConS multipath connectivity services (see D.C.1 Section 5.2.4) can select the best multipath strategy (distribution, splitting, and replication) to retrieve content to the mobile devices. The selection and enforcement of these strategies are performed at the participant devices as well as in the network.
- For poorly or temporarily disconnected users, OConS Delay Tolerant Network (DTN) routing (see D.C.1 Section 7.2.2), OConS Mesh Networking (see D.C.1 Section 7.2.1), optimized CQI channel allocation (see D.C.1 Section 7.1.2), and OConS Network Coding (see D.C.1 Section 5.1.2) can further improve the NetInf nodes connectivity.

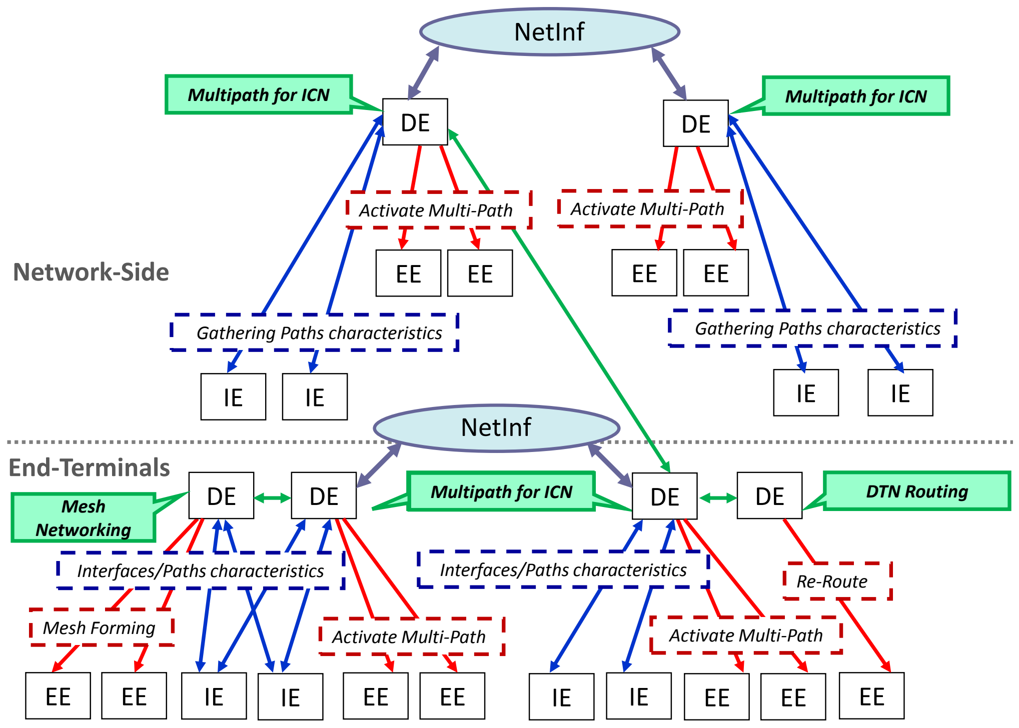


Figure 1.3: Mobile and Multi-P for Information Centric Networks use case

2 Description of Prototype Building Blocks

This chapter describes the prototype and demonstration activities based on the specific partners' contributions and building blocks as they were planned and realized in Phase 1 clusters (section 1.1); moreover, some of them have been shown and demonstrated during the Lisbon demo event in last January. Likewise, we have tried for each specific descriptions to include the following:

- functionality, purpose of the prototype, and how it contributes or relates to OConS
- component architecture (i.e., the description of sub-components/modules)
- realization details (platform, OS versions, programming language, etc.)
- internal and external interfaces ('open' protocols or services to be reused)
- basic use case(s) with the main steps followed (functional at an OConS level)
- demonstration and show-case aspects (e.g., including the environment needed to visualize the OConS functionality)

Our prototyping activities mainly focus on the two use cases, 'OConS for CloNe' (as detailed in 2.1 and 2.2), and 'OConS for NetInf' (as presented in 2.3 and 2.4), as well as on some common components that could be reused for both use cases (see 2.5).

In addition to these building blocks that are already available, we provide in chapters 4 and 5 what modifications are planned and on what we intend to work internally in WP-C to support the two main use cases, CloNe and NetInf, respectively.

2.1 Building Block: OConS Flow-Based Domain Connectivity Control

2.1.1 Functionality and purpose of the prototype

The prototype realizes experimental concepts for flow-based connectivity control per (network or technology) domain to be applied in multi-technology networks across layer3 routing, layer2 switching and in future even optical switching, especially with regard to control, management and algorithmic flexibility. It aims at providing a testbed allowing experiments with new variants of path and resource allocation algorithms and their interfaces in a realistic network environment, demonstrating technical feasibility and gaining first performance measurements on processing delays, complexity, and capacities.

The architecture under consideration introduces a unifying control element in each supported network domain called the Domain Control Unit 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 Domain Control Unit (DCU) combines concepts of OpenFlow-based controller, Path Computation Entity and Traffic Engineering Database for topology/resource advertisement and discovery as well as path computation and path/flow realization across multiple domains.

For experimentation purposes we follow an OpenFlow-like approach to realize a flexible and extensible test installation of multiple virtualized 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 (layer2) switches and (layer3) routers in virtual and physical networks, as it is common in the context of interconnectivity of large distributed datacentres and cloud centres.

The current demo can show control and management of a data paths between endpoints in distributed remote (cloud) datacentres with inter-domain connectivity via web-based service and management interfaces from any browser (hence providing a Representational State Transfer (REST)-ful DCU interface for service invocation and orchestration).

2.1.2 Component architecture with relevance to OConS architecture

How instances of DCU control servers and their clients communicate is shown in Figure 2.1, as well as their role in the OConS architecture as IE, EE, and EE.

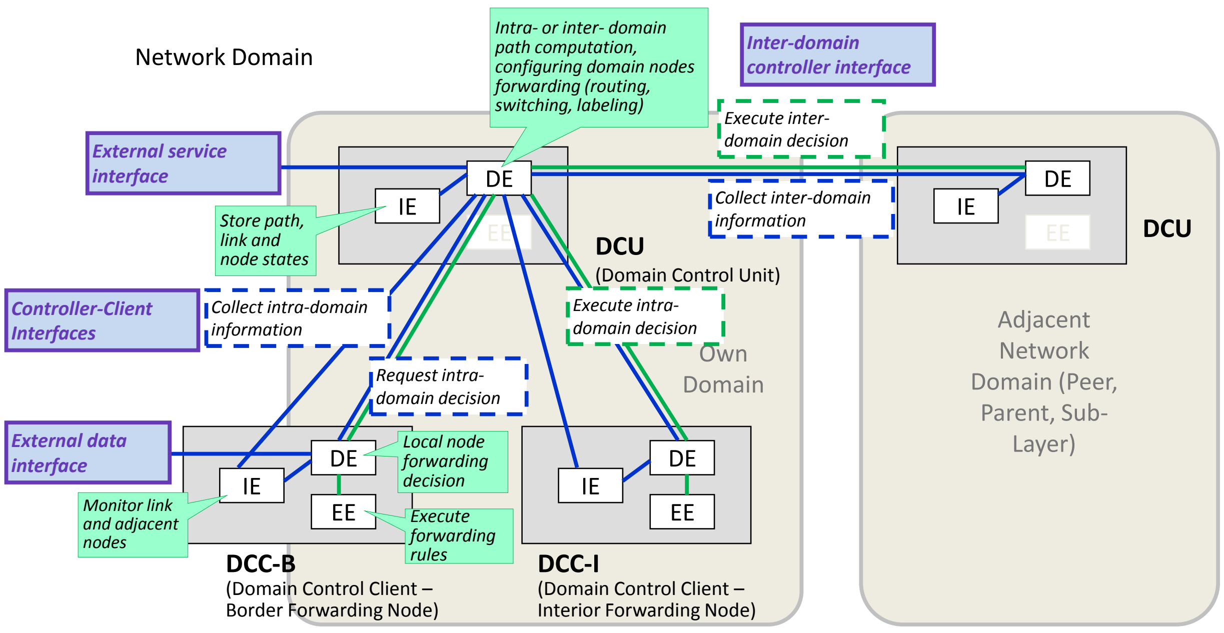

Network Domain
Intra- or inter- domain path computation, configuring domain nodes forwarding (routing, switching, labeling)
Inter-domain controller interface
Execute inter-domain decision
External service interface
DE
IE
EE
Store path, link and node states
Collect inter-domain information
DE
IE
EE
DCU
DCU
(Domain Control Unit)
Controller-Client Interfaces
Collect intra-domain information
Execute intra-domain decision
Own Domain
Adjacent Network Domain (Peer, Parent, Sub-Layer)
Request intra-domain decision
External data interface
DE
Local node forwarding decision
IE
EE
Execute forwarding rules
Monitor link and adjacent nodes
IE
DE
EE
DCC-B
(Domain Control Client – Border Forwarding Node)
DCC-I
(Domain Control Client – Interior Forwarding Node)

Figure 2.1: Domain control architecture - components and interfaces

The Domain Control Client (DCC) comes in two flavours. The border node DCC (DCC-B) for inter domain connectivity and the domain internal DCC (DCC-I). The internal functional architecture of the DCU and the interaction between its components is illustrated in Figure 2.2.

2.1.3 Description of components/modules of the prototype

The DCU is the control server consisting of several internal components, namely Dynamic Flow Control (DFC), Advertising Supporting Function (ASF), Traffic Engineering Database (TED), Path Computation Entity (PCE), Request Processing Entity (RPE) and the external interface communication via Domain Control Protocol (DCUP) [8].

The RPE catches, interprets and coordinates incoming requests and converts them into the PCE protocol language if needed. It transforms the incoming requests into a thread which keeps the DCU and the TED stateless and allows for concurrent requests. Thus the RPE takes up the role of

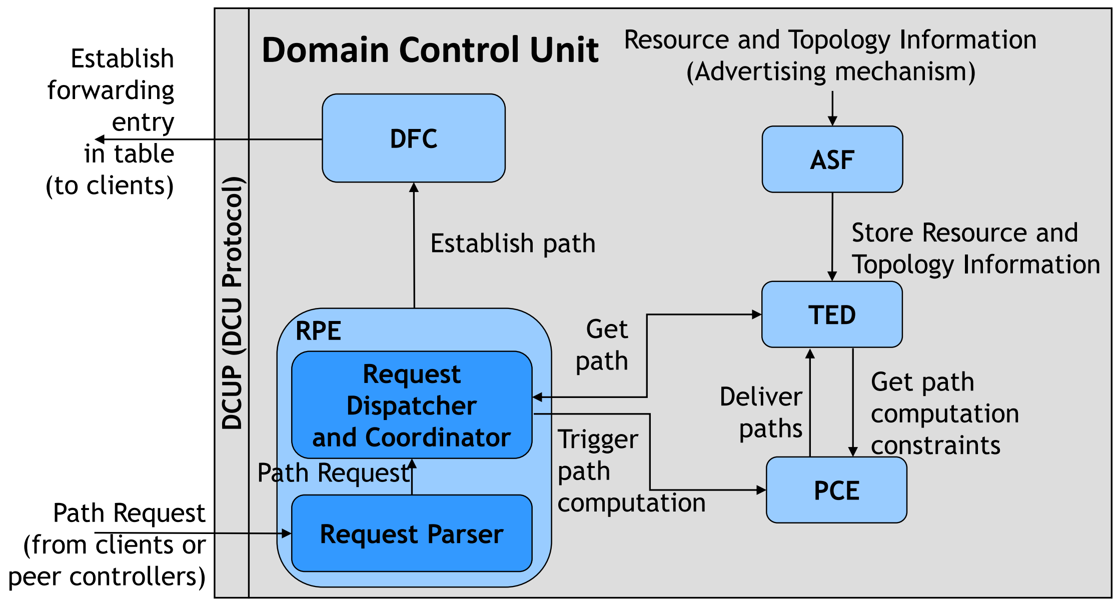

Domain Control Unit
Resource and Topology Information (Advertising mechanism)
DCUP (DCU Protocol)
Establish forwarding entry in table (to clients)
DFC
ASF
Establish path
Store Resource and Topology Information
RPE
Request Dispatcher and Coordinator
Get path
TED
Deliver paths
Get path computation constraints
Trigger path computation
Path Request
Request Parser
PCE
Path Request (from clients or peer controllers)

Figure 2.2: Domain control architecture - internal DCU components

the Inter-Node Communication (INC) function. It identifies incoming messages and directs them to the appropriate DCU internal entities, e.g. to the PCE DE.

The functionality is able to process unknown flow requests as well as explicit path set-ups. In case of an explicit path set-up the DFC is directly triggered from the RPE to realize the path in the NEs. If an unknown flow request arrives and an explicit path calculation is needed to serve this request, the RPE triggers the PCE for path calculation.

The TED is the IE that persistently holds the current topology information, the current state of the network resources, traffic parameter, user/provider policies of the local network domain and pre-calculated (and continually updated) paths. In order to speed up the end-to-end path construction, the TED is configured to contain a set of pre-calculated inter-domain paths. It is also recommended that the TED stores a set of calculated paths, which guarantees quick availability of paths in case of failure.

The ASF retrieves topology and resource information from the Network Elements (NEs), which are supported from appropriate measurements in the NEs. Thereby the ASF fulfils the function of the Orchestration Entity (OE), e.g. during the start-up phase of the network it registers all DCCs of its domain. The ASF fills or modifies the appropriate entries in the TED. Beyond this, the topology and resource advertising mechanism can be triggered by event, periodically or on demand. In this way, the ASF assures that the TED is always kept up-to-date. However, for scalability reasons it is important to balance the actuality and the number of advertisings with an appropriate dynamic configuration of the advertising mechanism.

The PCE part is responsible for the computation of intra-domain and inter-domain network paths. It is provided with current topology and resource information, with traffic parameters and management policies, and feeds the forwarding tables in the remote NEs of the domain via the DFC.

2.1.4 Realization details

This prototype consists of the DCU controller and the network emulation.

The DCU controller is realized in a Java/Open Service Gateway Initiative (OSGI) Environment which is embedded in an Eclipse (Helios) Integrated Development Environment (IDE) [9]. This IDE runs on Linux as well as on Windows XP. In its functionality, it is based on and extending the OpenFlow Controller 'Beacon' [10].

The test network realization including the OpenFlow switches is realized by Mininet Virtual Machines (VMs) [11]and virtual switches at distributed LINUX platforms (currently based on Ubuntu 10.04 LTS).

For further details, including external service/data, controller-client, and inter-domain controller interfaces, please see [8].

2.1.5 Internal and external interfaces, used protocols or services

As depicted in figure 2.2, the interfaces are the following:

- External service interface: REST-ful control interface based on AJAX web technology, provides the Graphical User Interface (GUI) for the controller in any web browser (HTML, http, JSON coded).
- Inter-domain controller interface: uses the same data coding and functions as the service interface, just without GUI presentation. Can be invoked by functions like 'wget' and 'curl' with respective parameter sets.
- Controller-Client Interfaces: uses the OpenFlow controller to OpenFlow switch protocol V1.0 or V1.1.
- External data interface: data plane interface with IP-Ethernet stack.

2.1.6 Demonstration and show-cases

The current prototype version realizes the following basic functionalities:

We show a virtual network testbed with three network domains and their controllers, physically connected to each other via dedicated attachment points for control and data plane.

Triggered by data traffic (via the external data interface), or managed by a web interface to the domain controller, flows can be set up, monitored and released within the own network domain (intra-domain), realized by a virtual network testbed (including the switching/routing clients).

The domain controller monitors availability of any switching client and connecting link in the network domain, also performance (QoS) of active flows, and can rearrange flows in case of degradation.

In the inter-domain flow set-up, we use a separate 'umbrella' controller that only has knowledge about the inter-domain connectivity via the attachment points form the external view, and provides the necessary inter-domain controller communication (on domain membership of certain nodes, and possible sub-flows between the domain edges).

The controller web interface shows network states in terms of active flow tables, available network resource tables and visualizes actual flows on the network topology, for its own domain. The 'umbrella' controller shows the same features on an abstract inter-domain topology, without details of the involved sub-domains.

By orchestrating the plug-ins of the controllers, different routing algorithms can be activated on the fly such as 'shortest path', or 'constraint-based routing'.

For further demonstration and show cases (in phase 2), we develop an demo application to drive the basic functionalities above.

Video processing is used to demonstrate the internal functionality. We implement two end users each watching a video. For the video processing an appropriate processing amount on certain processing units has to be reserved inside the Mobile Cloud Datacentre. Moreover the links to and inside the Mobile Cloud Datacentre service have to be established and monitored. All this is orchestrated and managed by a DCU. This DCU is the control entity that collects (IE) the needed information about current load of processing resources and link resources. Using a resource allocation algorithm the DCU DE decides on which processing resources the users are processed inside the Mobile Cloud Datacentre. Thereby the video flows can be switched over to other resources using other paths if needed. Connectivity changes are enforced (EE) by the DCU in the DCU clients.

In the demonstrator we cannot implement a full cloud data processing chain. Thus we choose the video processing analogue above to show that tight time constraints can be met and the needed processing units can be distributed remotely in the cloud and also connected in a flexible way.

2.2 Building Block: Interconnecting Datacentres using OpenFlow

OpenFlow is taking up as a way of providing a unified control plane for future networks. The use case of OpenFlow in the datacentre has been presented by Google in the 2012 Open Networking Summit. The objective of this contribution is to provide a proof-of-concept for OpenFlow in the Core Network interconnecting datacentres and spreading computing and storage power from the Datacentre to the Network and describing this new network functionality as an Open Connectivity Service (OConS). This work takes advantage of some of the functionalities provided by Cloud Networking (CloNe) and therefore is one of the components in the cross workpackage work between WP-C and WP-D.

2.2.1 Functionality

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

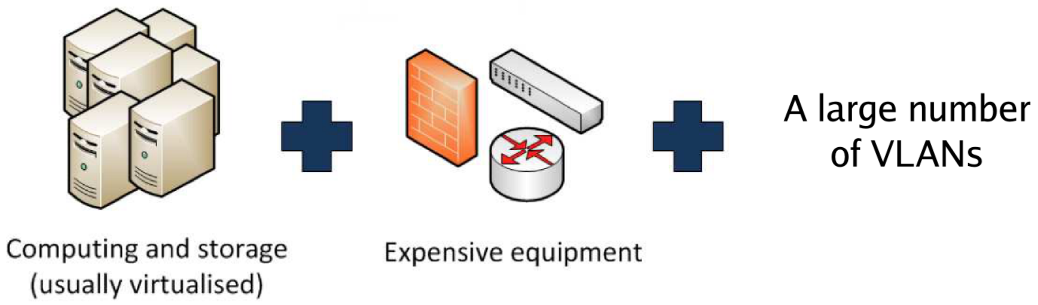


Figure 2.3: Current Infrastructure As A Service solution - Datacentre technical solution

The use case proposed and implemented here shows how an OConS based on OpenFlow as technical solution can implement the datacentre interconnection use case, using an infrastructure that implements multi-path transport services in a distributed/collaborative architecture. The OpenFlow-based OConS allows to implement a network interconnection datacentres that is a distributed datacentre itself. OpenFlow provides a potential solution for some of the limitations identified in datacentres and helps lowering the entry barrier to “programming the network” solutions. The use of OpenFlow removes some expensive network elements (i.e. firewalls are substituted by the flow specification embedded in the OConS) and simplifies both control and management planes. In the datacentre, the OpenFlow based OConS also provides a means for a tag-less implementation for current Infrastructure as a Service (IaaS) services that is cheaper and more scalable than state-of-the-art hardware based solutions.

The current state of the demonstrator shows how it is possible to define and manage an IaaS solution that controls the connection of different datacentre resources such as hosts, switches, etc. over inter-domain environments using NetFlow and FlowVisor as a Network Slicing Solution. On a first stage we implemented an environment based on Mininet on a virtualized environment that can be run on a PC for easy demonstration purposes. In parallel, we also have implemented a demonstrator based on general-purpose hardware. Both environments are controlled via a GUI that enables us to demonstrate the different technologies in a user friendly environment.

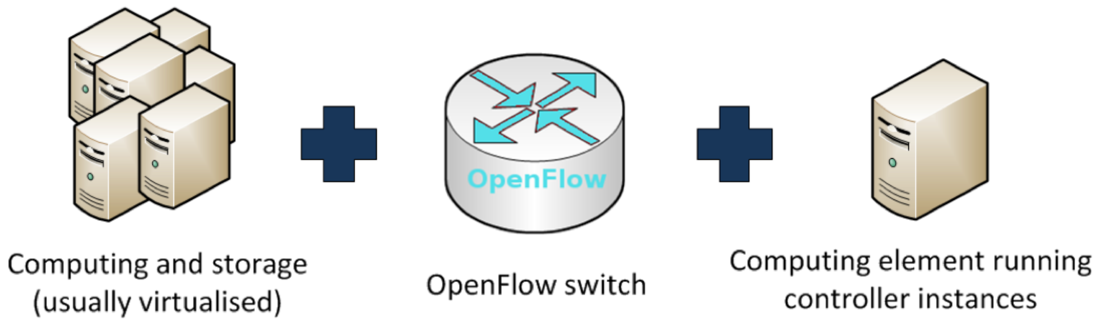

OpenFlow
Computing and storage (usually virtualised)
OpenFlow switch
Computing element running controller instances

Figure 2.4: Proposed Infrastructure As A Service solution - Datacentre technical solution

2.2.2 Component Architecture

This demo integrates different elements of the OConS infrastructure into the OpenFlow-based Datacentre implementation. Figure 2.5 shows how the integration works.

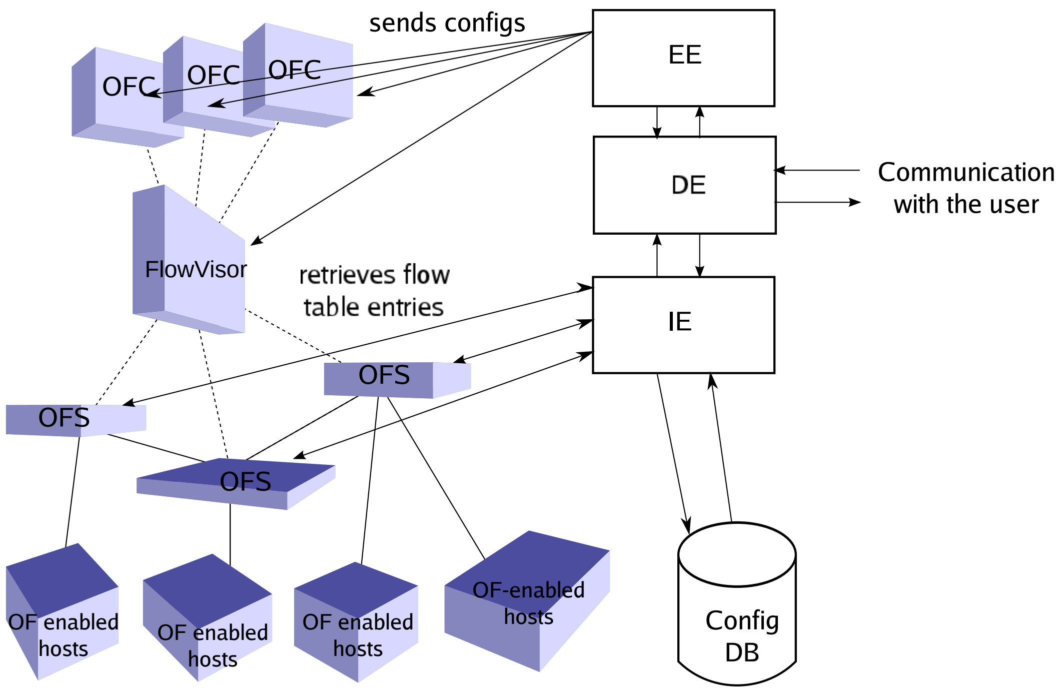


Figure 2.5: Integration of OConS elements in the OpenFlow Data Centre infrastructure

The Information Management Entity (IE) contains all the information of the service. In the integration with the OpenFlow enabled datacentre, the IE interfaces with the database that hold the description of the infrastructure.

The Decision Making Entity (DE) receives requests, checks whether the service is available in the datacentre and whether resources are available to complete the service request. In case resources to successfully complete the request are available, it progresses the request as a sequence of request fragments to the EE.

The Execution and Enforcement Entity (EE) interfaces with the FlowVisor and the different OpenFlow Controllers. It sends the request fragments related to client isolation to the FlowVisor and request fragments related to the implementation of the service for a specific client to the OpenFlow controller associated with it.

2.2.3 Description of Components

2.2.3.1 Information Management Entity

The IE interfaces with the OpenFlow Controllers and the FlowVisor. It collects the active flows from the switches.

2.2.3.2 Decision Making Entity

The DE interfaces with the OpenFlow infrastructure and the external interfaces defined in WP-D. It receives queries for new OpenFlow paths, computes the best path in the infrastructure, hands this information to the EE and sends the result received from it to the external interfaces.

2.2.3.3 Execution and Enforcement Entity

The EE interfaces the DE with the OpenFlow infrastructure. It sends the OpenFlow path establishment commands to the OpenFlow controllers and the FlowVisor and sends their responses to the DE.

2.2.4 Implementation details

The current implementation of the different OConS and CloNe components is OS and system neutral. We use Python 2.6 as the basic programming language to interface to the OpenFlow elements in the infrastructure.

The current version of the prototype interfaces both with a Mininet [11] 'OpenFlow in a box' demonstrator running in a VirtualBox [12] environment and a hardware infrastructure with nodes that provide networking and datacentre capabilities based on the XEN hypervisor [13]. This virtualization platform integrates an OpenFlow switch implementation known as OpenVSwitch.

2.2.5 Demonstration use cases

The current prototype integrates into the Mobile Datacentre demonstration. Currently, it demonstrates how a Datacentre can be implemented using OpenFlow. At the end of the project, it is expected to demonstrate how OpenFlow can be used to implement the networking component of Open Connectivity Services and how OConS can provide an interface that fully integrates the Datacentre into a providers network infrastructure.

2.3 Building Block: Multi-path Content Delivery for Information Centric Networks with OConS

2.3.1 Functionality

Multi-path connectivity to networks in modern computing devices is becoming the norm in today's computing. These multiple paths can be used in many ways that benefit the users of these devices as well as the different service providers involved. The ICN architectures that are currently being defined have considered the use of multiple paths natively. Though these architectures provide the use of multiple paths simultaneously, no formal mechanisms have been defined to utilize them in the best possible manner. The architecture and prototype developed here demonstrates the use of a number of multi-path strategies for ICN through the functionality of the OConS framework to retrieve and deliver content in the best possible manner.

2.3.2 Component Architecture with Relevance to OConS

The architecture of the prototype consist of a number of components that interact with each other to effect the selection and the implementation of multi-path strategies. These components, that follow the functionality of the OConS framework extend the operations of the overlaid ICN architecture to perform the multi-path operations.

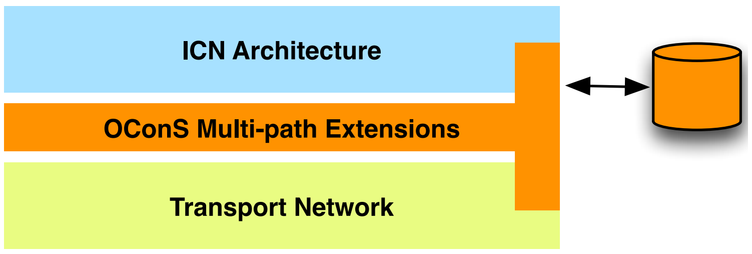


Figure 2.6: Protocol Stack (Generic Architecture)

Figure 2.6 shows how the extensions are positioned in terms of the layers of the protocol stacks of current ICN architectures. These extensions, operating as a shim layer between the ICN layer and the lower layers that provide transport connectivity implements the OConS components to perform multi-path.

The OConS DEs, located at selected locations of the ICN network identify the appropriate strategy to adopt based on the information provided by the IEs. There are 3 categories of strategies that the DEs can make.

- Distribution Strategy
- Splitting Strategy
- Replication Strategy

Each of these high level strategies have sub strategies. The details of these strategies are held in a repository in terms of rules that consist of conditions and actions. Regular receipt of information will result in re-evaluating the conditions and implementing new decisions. The EEs are located in the user terminals and in the network.

2.3.3 Description of Components of the Prototype

The first prototype is based on Content-Centric Networking (CCN) architecture as an exemplification of an ICN approach; it follows a modular architecture residing at different parts of the network.

Information collection modules - The information collection modules provide changes to the information held by the ICN node in which the collection occurs. These include expiration information to requests for content and current path information. This information is passed on to decision elements to be used in rule evaluations. This module has a number of sub modules that focus on collecting information of a specific type. This modular architecture is extensible in a way that every time a newer type of information needs to be collected, a new specialized module is developed and plugged in.

Decision modules - The decision modules hold a repository of the defined rules and re-evaluate them every time the required information is received. Once a new course of action (i.e. strategy) is identified, it is informed to the enforcement modules to implemented them.

Enforcement modules - The enforcement modules interact with the decision modules to retrieve the decisions and then implement it at the location where the module resides. The implementation is specific to each of the ICN architectures that these multi-path extensions support. In the initial version of the prototype, a CCN base enforcement module is used as the overlaid ICN architecture is based on CCN.

2.3.4 Realization Details

The prototype is Linux based and a simplified ICN architecture is developed and used to extend for the OConS functionality. A video application is used to show the benefits of using multiple paths. Figure 2.7 shows how the prototype is operated in a test-bed where a terminal oriented scenario of making multi-path decisions is used.

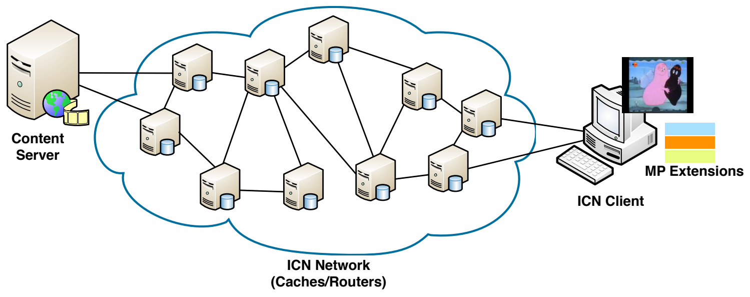


Figure 2.7: Realization of Extensions

The realization details are as follows.

- Operating System: Linux 2.6 or above based
- ICN Architecture: Simplified CCN implementation (uccn version 0.1), developed in C
- Video Streaming: CCN based video streaming wrapper for VLC
- Multi-path Extensions: IE, DE and EE functionality user terminal based and developed in C

2.3.5 Internal and External Interfaces

A number of interfaces are used by the prototype to obtain information and enforce the decisions. These interfaces are implemented through the overlaid ICN architecture that carries the messages back and forth.

The external interfaces mainly relates to the interfaces to the ICN architecture. They include retrieval of information about the status of the ICN operations. These interfaces are handled by the information collection modules. The other type of external interfaces are handled by the enforcement modules to control the ICN to implement the different multi-path strategies decided upon by the decision modules.

There are a number of internal interfaces that are related to the OConS framework. These interfaces are used to communicate information and decisions between the information collection modules, decision modules and the enforcement modules.

2.3.6 Demonstration Use-case

The primary use case that is used to demonstrate the benefits of these extensions is the Flash Crowd scenario.

2.4 Building Block: DTN routing based on adaptive learning from historical encounters with OConS

Exploring self-* properties of nodes belonging to a Delay Tolerant Network (DTN) and learning from neighbour encounters (context awareness), becomes of a great value in order to design an optimized transport strategy to improve service performance in this specific type of networks. Connectivity in DTN scenarios implies that nodes do not have permanent physical paths to certain destinations, but only to some of their closest neighbours instead. This work aims at the development and implementation of a mechanism that helps the node take a decision regarding packet routing and forwarding. There is a wide range of combinations that could be validated for several specific situations where delay tolerant transmissions would be optimized so as to be characterised by a certain expected Quality of Service (QoS). Our aim is to design and implement a prototype that makes use of a valuable subset of these properties and is able to exploit them for a smart management of the connectivity in DTNs formed by human-carried devices.

We propose a novel protocol, based on people's social routines, which introduces some enhancements to solve the problems detected with the use of Probabilistic Routing Protocol based on Historical EncounTers (PRoPHET) in certain environments studied. Our mechanism also infers the social behaviour of nodes from the history of contacts but unlike [14], incorporates the contact duration to the information retrieved from historical encounters among neighbours. It also modifies the assignation of direct probabilities in [14], introducing a mathematical equation that evaluates the quality of the contact. The specification of this solution is based on the outcomes of a thorough experiment performed in a working environment, which collected contact traces of 56 human carried devices in the same office building during several weeks. The analysis carried out over the voluminous contact database obtained from [15], highlighted the importance of distinguishing between short and long contacts and deriving mathematical relations in order to optimally prioritize the available routes to a destination. Some simulation results, as compared with the performance of PRoPHET for the same scenarios, are summarised in [16] (paper submitted to CHANTS workshop).

2.4.1 Functionality

When a DTN node receives a packet addressed to a certain destination, a set of steps are triggered in order to find the most suitable way of reaching this destination node. The easiest situation (apart from being the destination node) would be that the destination is one of the node's direct neighbours: in that case, the packet is just at one hop distance from its final destination, and there is a physical connection available. Otherwise, the node will need to analyse its available routing information and take an action according to one or several of the following aspects: accept or discard the packet (buffer constraints), store the packet and wait for a suitable forwarding instant (based on the probability of reaching the destination node within a certain time period), forward the packet immediately to an intermediate node with higher probability of finding the destination (based on connectivity or mobility pattern estimation, self-learning parameters...), combine the packet with other previously stored ones headed to the same destination node (related to Network Coding techniques), etc. The prototype presented in this section is aimed to show how this routing decision can be performed using inter-contact and contact duration times of historical encounters among neighbour nodes in a DTN. This mechanism is based on probabilistic routing techniques, although it incorporates the contact duration of encounters (not considered in previous approaches), and it proposes a novel reasoning algorithm for a DTN node to estimate the rating probabilities of possible paths to a certain destination.

2.4.2 Component Architecture with Relevance to OConS

OConS architecture is based on the entities described as Information Management Entity (IE), Decision Making Entity (DE) and Execution and Enforcement Entity (EE). As per the inherent nature of the connectivity established in a DTN environment, each node becomes an autonomous entity and all mechanisms are performed in a hop-by-hop basis, since the end-to-end concept is no longer true and/or available. Figure 2.8 represents a possible OConS scenario, where several nodes are part of a mobile DTN and eventually one OConS terminal (Node C) might have access to the outside world (i.e. the Internet) through a wireless technology interface (apart from its available DTN physical interface, which might be based on the same or different wireless technology).

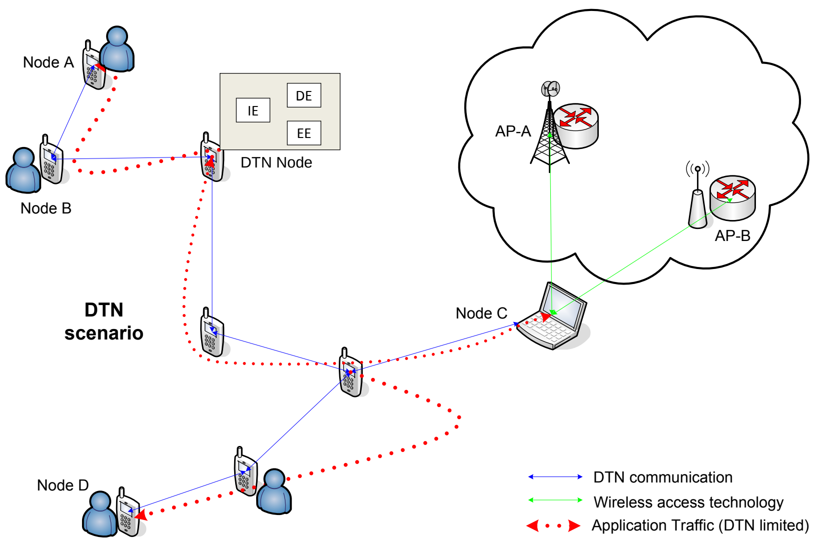


Figure 2.8: Generic architecture for components in a DTN scenario

In this case, the three OConS components of the architecture are present in every single node belonging to the DTN: IEs are the components responsible for interchanging information regarding previous encounters and estimated path-ratings with neighbouring nodes; DEs are those in charge of performing the proposed decision algorithm and come up with rating values for all possible learned paths; and EEs are the components that execute the forwarding of pending packets to a certain destination, via the best next hop, previously decided (node with the highest rating value) by the DE. In Figure 2.8 Node A must find a physical path for reaching Node C and gain access to the outside world. Let us assume that every DTN node in the scenario is aware of the capability of Node C for accessing the Internet. If Node A demands a specific content (e.g. a video from Youtube) from the Internet, it will need to decide how to reach Node C (i.e. decide which intermediate neighbour would probably contact Node C faster, or in a more reliable way). DTN nodes are mobile, and they might be storing information about who contacts whom, with which frequency and for how long. Once Node A contacts Node B, they exchange information about the probability with which they expect to see Node C 'in person' (this probability is based on their own previous contacts with Node C, or on learned probabilities from intermediate nodes). If the probability of Node B for reaching Node C is higher than the one stored by Node A, the latter will forward its packet (headed to Node C) to the former. Also, Node A would store Node B as its best next hop, for packets headed to Node C, in its routing table. If a new direct neighbour contacted Node A and provided a better probability value for Node C, this would be updated accordingly (adaptive learning process).

2.4.3 Description of Components of the Prototype

The overall process implemented in a DTN node can be represented by the steps sketched in Figure 2.9.

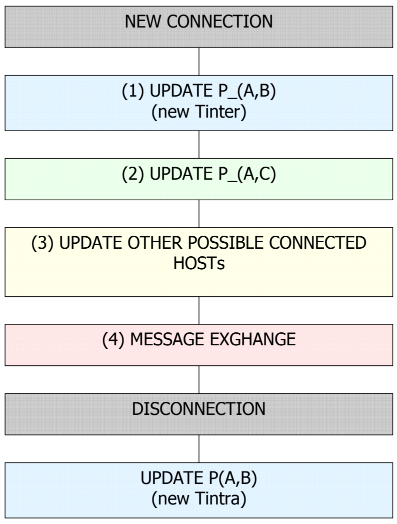
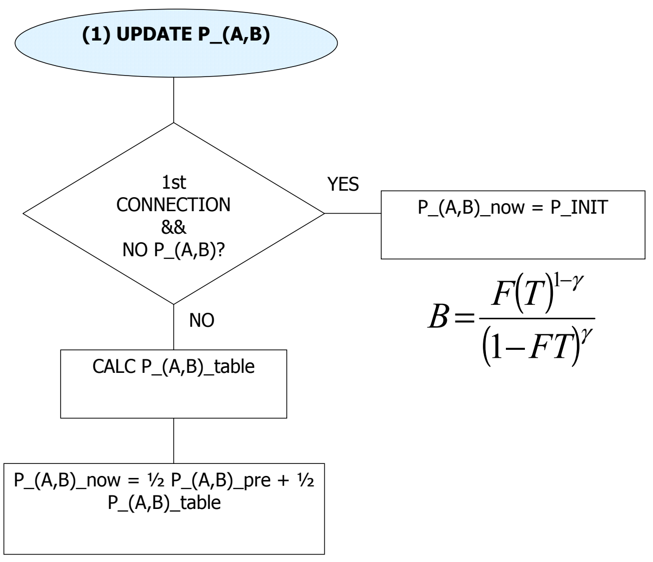

NEW CONNECTION
(1) UPDATE P_(A,B) (new Tinter)
(2) UPDATE P_(A,C)
(3) UPDATE OTHER POSSIBLE CONNECTED HOSTs
(4) MESSAGE EXGHANGE
DISCONNECTION
UPDATE P(A,B) (new Tintra)

Figure 2.9: Flow Chart of the mechanism implemented in a DTN node

Figure 2.9 shows the flow chart of the whole mechanism from a node's perspective, node A, when it detects a new physical connection to node B. P_(A,B) is the direct probability of node A contacting its neighbour node B, and updated with the inter-contact time since their last encounter (new Tinter). After that, node A would update the values of non-direct probabilities, P_(A,C), learned through transitivity (node B informs about its probability of reaching the rest of nodes, represented here by letter C). Node A checks if other possible connections are detected simultaneously (other direct neighbours) in order to update the values of its own direct probabilities, before it exchanges this information with node B. If physical connection with node B is lost (disconnection), the value of P_(A,B) is also updated with the last contact duration (new Tintra).

Each of these steps is implemented by a specific sequence of methods and algorithms. As an example, Figure 2.10 shows the detail of how a Decision Element (DE) would estimate a direct probability P_(A,B).


(1) UPDATE P_(A,B)
1st CONNECTION && NO P_(A,B)?
YES
P_(A,B)_now = P_INIT
NO
CALC P_(A,B)_table
P_(A,B)_now = ½ P_(A,B)_pre + ½ P_(A,B)_table

$$B=\frac{F(T)^{1-\gamma}}{(1-FT)^{\gamma}}$$

Figure 2.10: Detail of the assignment of direct probabilities

In this flow chart, the formula of B (Goodness) represents the way of measuring a neighbour's quality (direct rating algorithm). In this formula two parameters have been considered: the frequency of contacts, or the inverse value of inter-contact time; and the contact duration, between two neighbours. Considering both parameters normalized to the same period, F and T would represent the weighted mean of historical values stored by the nodes. The process of estimating a representative mean value (both for inter-contact and contact times) is based on the statistical features that characterise both mathematical distributions.

2.4.4 Realization Details

Figure 2.8 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 realization details are as follows.

- Operating System: Android 2.2 or above based
- Routing protocol implementation: social routing (PRoPHET based) implementation, developed in Java, taking the Bytewalla3 project release as a basis [17]
- Optional Bundle Protocol Query (BPQ) Extension [18]: BPQ functionality to be developed in Java and integrated into the Bundle Protocol implementation of the Bytewalla3 project
- DTVideo service: simplified video service implemented over Android phones

2.4.5 Internal and External Interfaces

As stated in section 2.4.1, all components of the OConS architecture are implemented on every DTN node of this prototype. This way, Information, Decision and Enforcement Elements are internal components of a node, and could be seen as sets of functions interacting both with their peer components of different nodes, and with other component within the same node. Moreover, the interfaces playing a role in this prototype are the following:

- IE-IE: this is an interface between peer components of different nodes. Messages exchanged include information about node identification and stored probability values of reaching third parties (DTN nodes not directly connected at the time, not direct neighbours)
- IE-DE: this is an internal interface within a single entity. The IE component informs to the DE about probability values learned through transitivity, and also about parameters collected from direct encounters. The IE stores both the inter-contact and the contact duration times, which are used by the DE for the estimation of direct probabilities
- DE-EE: this is an internal interface through which the DE delivers a forwarding petition to the EE. Whenever a node has pending packets to be sent, the DE makes an order to the EE with the physical connection to be established (according to the rating process executed and the highest probability value for reaching the destination node)

The external interfaces mainly relates to the Bundle Protocol implementation for the realization of the prototype. They include the interface to the application service used for demonstration and, eventually, interface to specific module implementations (i.e. [18]).

2.4.6 Demonstration Use-case

The primary use case that is used to demonstrate the benefits of these extensions is the Flash Crowd scenario.

2.5 Building Block: Distributed Access Selection and Mobility Decision with OConS

2.5.1 Functionality and purpose of the prototype

This demonstrator aims at assessing the feasibility of the OConS architecture, which is challenged to bring about a dynamic and distributed mobility management. The demo embraces different mobility-related procedures, which are triggered by various events, coming from various sources (e.g. high load situation at an access element, low link quality, etc). Completely based on the OConS architecture, mechanisms and (signalling) protocols, the demo shows how the decisions are taken on a distributed manner by the network nodes and the end-user terminals.

Since the main goal is to assess its feasibility, the information gathering, decision-making, and execution procedures have been carried out according to the OConS architecture; in this sense, they have followed the involved mechanisms, such as bootstrapping, discovering, etc. Some OConS entities (*Decision Making Entity* (DE); *Information Management Entity* (IE); *Execution and Enforcement Entity* (EE)) have been implemented within both the user terminal and the access elements, and their appropriate operation has been validated.

2.5.2 Component architecture

From a networking perspective, as depicted on 2.11, the demonstration is composed of one user terminal and a number of network access elements; all of them incorporate the OConS architecture so as to enrich the decision-making mechanisms, considering the information gathered by the OConS entities.

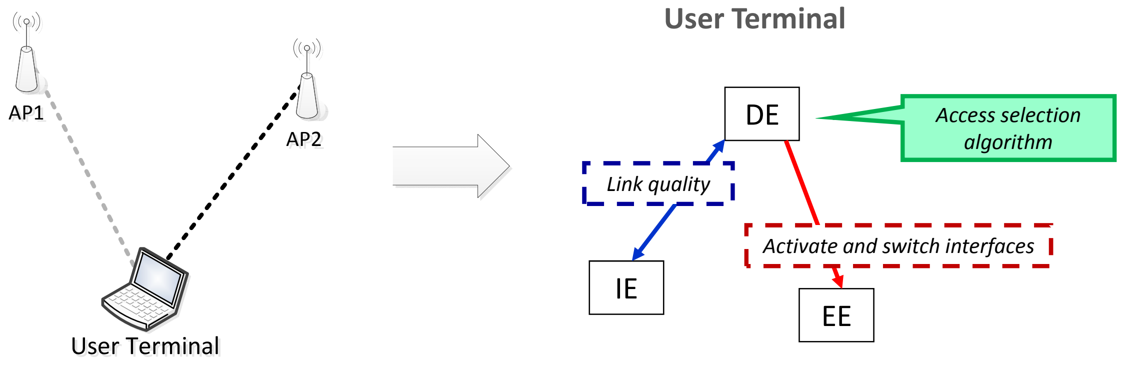


Figure 2.11: Component architecture

As general method, all entities are controlled by an additional component, which resembles the role of the Inter-Node Communication (INC) process in general. In this sense, the rest of entities (during the bootstrapping process) need to register to the INC. Afterwards, any entity might be able to know the location of any other one either through the INC or by means of a dedicated message sent by the INC, if it was configured to do so (see an illustrative example in 2.12). After the registration has been completed, the inter-entity communication can be carried out.

The already available mechanisms include the bootstrapping, registration and discovering processes, which are needed so as to later on leverage an enhanced access selection[1].

[1]At the time of writing, this is taken at the user-terminal side, but this will be extended so as to also consider the access elements within the decisions process, including e.g. load information.

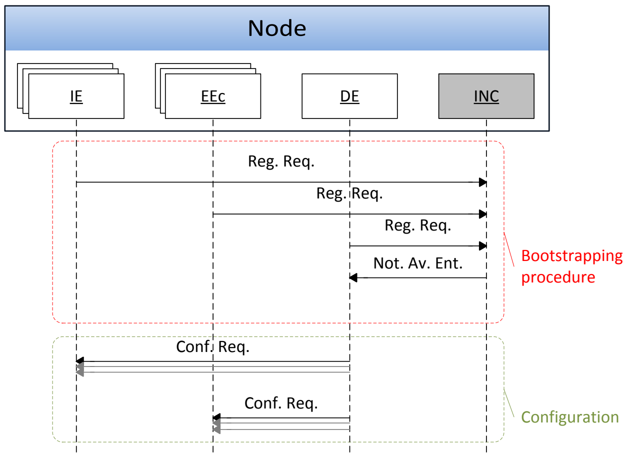


Figure 2.12: General registration process

The OConS implemented architecture consists on some entities which are able to establish the optimum point of attachment to the network according to the information gathered by the corresponding IEs. To do this, the end user terminal incorporates an IE, which is responsible for collecting information about the available access elements and report it to the decision entity (DE). Whenever the DE takes a decision, this is sent to the EE, which will instance the corresponding mechanisms to enforce it, and change the connection to the selected access element. Finally, once the connection is established the discovering mechanism is accomplished through the INCs belonging each node.

2.5.3 Description of components/modules

At the time being, the demo consists on the basic access network shown in Figure 2.11. In addition, we have developed a GUI that eases the process of monitoring the overall demonstration performance and operation. The GUI allows the monitoring of both the OConS mechanisms and message exchanges, according to the OConS interfaces specification. In addition it can be used so as to emulate certain situations, for the sake of a better tracing of the corresponding methods and procedures, which would eventually lead to a handover event[2].

2.5.4 Realization details

Regarding the details of the development, the whole demo has been implemented over regular laptops[3]. *Linux/Ubuntu* (stable version 10.04) has been used as the operating system, with the $2.6.32-28-generic$ kernel. The implementation, which can be divided into two main streams, uses two different programming languages; on the one hand the OConS architecture has been coded in *C++*, while the GUI was implemented in *JAVA*. Furthermore, all the communications between the different entities (between OConS modules and with the GUI) are socket-based.

[2]At the time of writing it specifically can modify the quality of the links with all the available access elements, so as to force a handover situation whenever it is required.

[3]In this sense, the access elements use the *hostap* functionality so as to fully emulate the role of a regular Access Point

2.5.5 Internal and external interfaces

The interfacing between OConS entities is done by means of a proprietary protocol, which uses Type Length Value (TLV) coding. It is somehow similar to the IEEE 802.21 (Media Independent Handover) specification, although it has been tailored to better serve our purposes. Since the currently available demonstration does not need any interaction with external entities nor services, no specific external interface has been yet defined.

2.5.6 Available basic show-cases, demonstration use cases

At the moment the demo shows a step-by-step OConS bootstrapping process, which can be monitored by the GUI. The various entities register to the Orchestration Entity, which facilitates the discovery and registration processes. Afterwards, the demonstration show how the OConS framework might bring about enhanced access selection procedures, by considering the information gathered by the corresponding information entities and enforcing the appropriate decisions.

2.6 Building Block: Dynamic Distributed Mobility Execution with OConS

2.6.1 Functionality and purpose of the prototype

The idea is to have a dynamic activation of mobility anchors in Access Routers (ARs), so that we use direct IP routing of traffic flows/sessions initiated on mobile's current Access Router, while supporting handovers and traffic forwarding/tunnelling between the Access Routers (current and previously used anchors ARs).

2.6.2 Component architecture with relevance to OConS

Focus here was put on the Mobility Execution Entity, see Figure 2.13. Thus, for the flash crowd scenario, the optimal data path is used each time a new flow is launched, the forwarding/tunnelling functions are activated only when needed to support flows' delivery in mobility situations, and the efficient connectivity (direct routing whenever possible).

2.6.3 Description of components and modules of the prototype

Several functions are needed on the Access Routers and on the Mobile Node (MN).

- Detection of the new applicative sessions: On MN we use Linux Conntrack to track the connections and then in user space we have a function to wait for Netlink events from Conntrack; we also check who initialized to connection, the mobile node or the correspondent. On the AR, the detection of the new applicative session is done with pcap library.
- Detection and handling the handover (L2 and L3): The MN detects the radio handover at L2 (i.e., the new access point) by listening to control messages from Netlink framework, and then the MN informs the current AR by sending a slightly modified ICMPv6 Router Solicitation message and indicating the list of active applicative sessions (if any).
- Mounting the tunnels on the networking side and re-route traffic accordingly: On the current AR, once we receive the RS we need to inform the previous ARs about the handover and

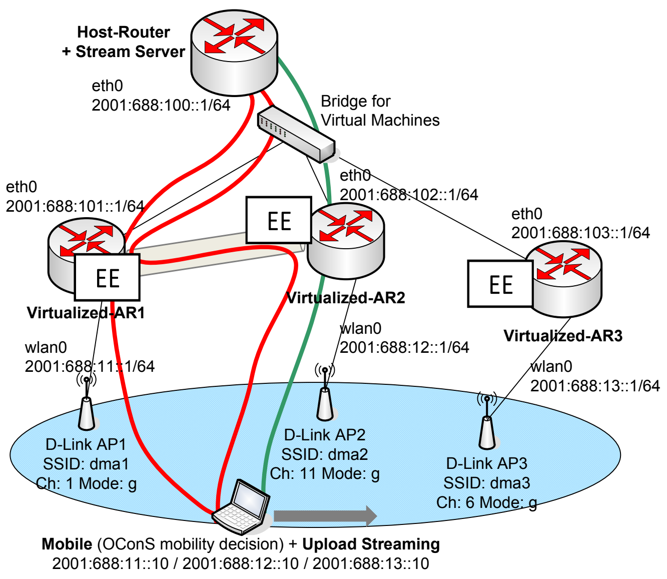


Figure 2.13: Dynamic Distributed Mobility Demo

to create the tunnels towards these ARs. This is done with a modified ICMPv6 Neighbour Solicitation message. Then we build the new tunnel by adding a virtual network interface and by indicating the two end-points of the tunnel and by attaching the tunnel to a real NIC (e.g., eth0). The routing is also update accordingly using the ip6tables and the ip -6 rule/route commands.

2.6.4 Realization details

For the networking part, the code was developed in C under Linux/Fedora distribution (kernel 3.1.8). As for the radio part, we have used D-Link access points with Atheros chipsets and running the hoastap demon in version 0.7.3.

2.6.5 Internal and external interfaces

As presented in a previous section, the internal interfaces for this mechanism reuses existing protocols, such as ICMPv6-based Router Solicitation and Neighbour Solicitation messages. As for the external interface, the Dynamic Distributed Mobility mechanism is triggered by the lost of L2 Wi-Fi connection (i.e., a GUI button-push command for the time being to deassociate from the old access point and to associate with the new one). Finally, note that this is done transparently for the applications' traffic, i.e., they continue to use the standard socket API without any modification.

2.6.6 Demonstration use case

A Linux-based IPv6 prototype was implemented and it is available in a lab environment, using Wi-Fi accesses and showing the dynamic per-flow mobility execution based on dynamic IPv6 tunnelling establishment between APs/ARs (see Figure 2.13). The future plan is to integrate it with evolved mobility decision entities using the OConS messages and the OConS orchestration functionalities currently under development (see chapter 4).

3 Analysis of OConS Scenarios

This chapter deals with the analysis of the revised use cases as developed in D.C.1-Addendum [2], section 4, and lays the basis for the evolution of the prototype building blocks towards Phase 2 till the end of the project.

We give a description of the selected sub-cases and decompose the use cases in smaller scenes, using existing "basic OConS mechanisms", or define additional ones, if required. Moreover, the abstract OConS services and mechanisms to be used in each sub-case are identified according the OConS terminology in terms of entities (IE, DE, and EE) and of service primitives (i.e., messages at the interfaces).

The scenes that will be implemented, and those parts which are only needed to set up the environment (pre-conditions) are also identified here, whereas the specific realization aspects will be described in chapter 4.

3.1 OConS for CloNe: Mobile Access and Datacentre Interconnection Use Case

Within the overall SAIL flash crowd scenario as presented in [3], we now consider a use case, where a large number of mobile users concurrently and suddenly, e.g., due to a particular event, ask for a wireless connectivity in a certain geographical region to contact their social community networks for downloading or uploading certain multimedia content.

3.1.1 Create new paths towards other instances of the same server via OConS signalling

For the future, we assume that the mobile network consisting of packet core network and Radio Access Network (RAN) will be virtualized and "cloudified" to a large extent. In the RAN case this means that the processing capacity, e.g. for baseband processing, user processing and cell related processing will be hosted in a more cloud-oriented datacentre, while at the cell site only the so called Remote Radio Heads (RRHs), containing the filter and amplifiers, are deployed.

Therefore we will use OConS concepts for connecting such datacentres (with specialized processing) and managing the load and connectivity between them, as this specific case with many users generates a flash crowd and, thus, challenges both the networking capacities as well as the available processing resources.

Let us assume the following: a number of end users wants to watch a video on their mobile phone, which in addition needs a personalized processing and adaption in a video processing node in the (mobile access) cloud. Thus an appropriate processing amount on certain processing units has to be reserved inside a mobile cloud datacentre 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 datacentres hosting the cloud application) or it will set up an additional path over less loaded processing nodes (datacentres) that are served by this OConS domain. All this is set up and managed by the cooperating and distributed DCUs servers, as entities that control the resources assigned to the respective mobile cloud datacentre resources.

The OConS mechanisms used in this use case and offering the necessary services are describe next:

- The overall reference model and control architecture for the Mobile Cloud Datacentre use case was introduced in Figure 5.3 of D.C.1; accordingly, the placement of the OConS entities on the controller-side DCU and within the DCU's clients (DCC) is shown.
- The DCU is responsible for the control of one OConS domain (cf. chapter 3.2 from D.C.1 Addendum).
- Decision (DE) and information collection (IE) entities are realized in the DCUs placed within the mobile cloud datacentre and the connecting OConS network domains.
- All three OConS entities (DE, IE, and EE) are used inside the 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 domain DCUs to exchange inter-domain information, e.g. on processing resources available remotely or link load between such processing resources.
- Several attributes (chapter 6.1.1 in D.C.1) such as resources currently offered or network quality of service (QoS), energy consumption and price (chapter 6.1.3 in D.C.1) can be used by the resource allocation algorithm.
- When the appropriate information is collected by the DCCs 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.
- In this way the network link load is influencing the instantiation of processing resources within a Mobile Cloud Datacentre and at the same time is assuring service stability.

Furthermore, the OConS Orchestration mechanisms (including its Register and the Monitoring functions) are employed to support the controller in the Mobile Cloud datacentre use case. Once the OConS node internal entities are discovered, the Orchestration processes of other OConS-enabled nodes must be discovered.

After that, all available network resources and capabilities have to be discovered and identified and have to be made available to the DCU via the node internal Orchestration functions. The discovery of a single resource, e.g. a baseband board or a general purpose CPU can run in parallel. The DCU can then start the algorithms and mechanisms that control the used resources and assign resources to incoming service requests accordingly. Then, the DCU monitors the resources for their usage, e.g. for link, storage and CPU usage; for example, a smart access control and intelligent load balancing algorithm can improve the system response and resource utilization.

The Mobile Cloud Datacentre DCU collects and identifies the available resources in its domain.

The DCU also monitors current resource usage data using the OConS protocols and Orchestration entities described above. The resource data can be the CPU usage for each processing board, the storage usage on each board, the link usage between boards and even between different Mobile Cloud Datacentre locations. Thereby the current bandwidth used, the delay, jitter and the current error rate on the link can be of interest for assigning optimal resources. In addition, via the Orchestration function, the DCU can react accordingly when application traffic changes, i.e. when users enter or leave the Mobile Cloud Datacentre. Also the mobile network operator can prepare the Mobile Cloud Datacentre in advance in case of an expected flash crowd, e.g. a soccer game or the like. This of course not only needs a control interface between the DCU and the resources in the network, but also a network management interface between the Mobile Cloud Datacentre and the operators operations and maintenance centre.

We can say that the OConS Orchestration is responsible to find the processing resources and connectivity resources that are available to the DCU; moreover, after having collected the required data it has to continuously update the current state of the processing and connectivity resources according to its usage. Thereby, in average, less processing resources shall be used for the Mobile Cloud Datacentres as in today's cellular networks. Furthermore the handling of a big flash crowd of users is not possible today without using the elasticity of a Mobile Cloud Datacentre. Finally, the interface of the end user towards the mobile system depends on the used mobile standard, like LTE, UMTS, or GSM; however the interfaces between the involved Mobile Cloud Datacentre functions are delivered within OConS.

When it comes to the implementation, we implement two end user instances each watching a video; for the video processing an appropriate processing amount on certain processing units has to be reserved inside the Mobile Cloud Datacentre. Moreover the links to and inside the Mobile Cloud Datacentre service have to be established and monitored. All this is orchestrated and managed by a Domain Control Unit (DCU). This DCU is the control entity that collects the needed information about current load of processing resources and link resources by its integrated IE. Using a resource allocation algorithm the DCU-DE decides on which processing resources the users are processed inside the Mobile Cloud Datacentre. Thereby the video flows can be switched over to other resources using other paths if needed. Connectivity changes are enforced by the DCU using the EE in the domain clients (DCCs).

In the demonstrator, we will not implement a real LTE or UMTS radio processing application. Thus we will choose the video processing analogue from above to show that tight time constraints can be met and the needed processing units can be distributed remotely in the cloud and also connected in a flexible way.

3.1.2 Facilitate users' handover by interfacing with mobility management

The OConS framework provides an enhanced way to perform access selection, according to both the current network status and user policies and preferences. In the mobile access to the *Cloud*, the end-user might be able to select a connectivity alternative between a relatively large number of options. In this sense, the information managed by OConS, together with the requirements coming from the corresponding end-user, shall be taken into consideration so as to promote an optimum selection. In addition, network and cloud operators (which in fact can be also seen as OConS users) might benefit from the available management facilities.

As a first step (in fact, this step is needed in order to call any OConS service), the various entities need to bootstrap. As a result of this, the Orchestration registry will compile information about all available entities, and services and mechanisms which can be used with them. In particular, regarding the access selection/handover procedures, all the management mobility-related functionalities (within IE, EE, and DE) shall be available within the appropriate nodes on both network and user side, so as to support the appropriate mobility management services.

In particular, these can embrace the provision of information about link quality or traffic load measurements, user profiles and requirements, thereby allowing OConS to take into account several parameters when a particular connectivity request happens. For instance, if the current connectivity configuration is not enough to address the requirements coming from the OConS user (for instance, a datacentre controller), an automatic reconfiguration (i.e., OConS triggered) shall take place without requiring the interaction with the corresponding user.

As can be seen, the OConS Orchestration plays a key role in the overall process; first, it discovers the OConS entities which will be available and the services which can be offered by them, as well as the appropriate configurations and combinations. Afterwards, the OConS entities and mechanisms need to be properly combined, so as to appropriately offer the required services. This implies the specification of adequate interfaces and signalling procedures between the involved parties.

In this particular case, OConS offers an enhanced access selection, which combines the requirements and policies of the end-user (and her services) with the particular conditions of the network, thus leading to a clear distinction between the nature of the information which is used; we further discussed below this process.

- On the user side, OConS mostly provides information concerning the type of service requested by the user (together with its particular requirements) and her preferences or profiles. The main goal is to keep the QoE, as well as ensuring the seamless continuity of the service if needed. Various end-user policies or rules are to be considered, i.e., those establishing the particular needs of the end-user according to some pre-established profiles (e.g., energy saving, technology preference, pricing, security, etc.) and those giving the service/application requirements (e.g., quality needed and minimum networking resources). Likewise, the above parameters can be handled on different ways, either in isolation or combined, according to a specific configuration (for instance, they might be weighted so as to modulate the decision token within the end-node, being part of the overall and coordinated decision).
- On the other hand, the network is able to compile information about its current status as well as considering end-users' perspective (e.g. radio-link quality such as SNR/RSSI, network delay, quality profile, etc.) and to take coordinated decisions considering a trade-off between QoS, QoE and network management goals (for instance, load balancing).

In this case, the access selection OConS service is conceived to deal with multi-homed/multi-interface devices, being able to handle flow mobility between different interfaces, to select and activate per-flow mobility anchors on network-side, and to set the right policies within user-terminals.

As can be seen from the above description, this OConS service is strongly based on the information provided by the various IEs; in this sense, the bootstrapping shall discover all the elements able to provide such information, and the mechanism shall be configured so as to establish which of them shall be considered within the decision processes.

Once this is done, the DE must be able to interact both with the IEs, to retrieve the required pieces of information; nonetheless, to cope also with multiple provider/operator cases, the appropriate decision models where suggested, for example we can configure the OConS mobility management service so that it is the end-user terminal that collects partial decisions coming from several operators providing the radio access and then it integrates them with the locally constructed decisions.

Then, the DE must be able to interact and with the EEs, so as to enforce the appropriate decisions both within the end terminals and on the network side.

The basic bootstrapping process, as well as the access selection and handover mechanisms, are currently under development. These specific functionalities of the OConS framework is generic enough so as to be used for various use cases and thus can provide benefits when the mobile access is considered in conjunction with the CloNe. Accordingly, we can imagine that these access selection and mobility functionalities interact more closely with the applications/services provided within the datacentres and the datacentres/cloud "controllers", for example, assuring service continuity when necessary, better data-path efficiency through offloading and anchor selection, influencing the instantiation of applications/services within a given datacentre, and so on.

3.2 OConS for NetInf: Multi-P and DTN for Information Centric Networks Use-Case

This scenario centres around the participants of a flash crowd that attempts to download content related to a street performer. The networks to which the flash crowd connect are based on information centric networking. The content related to the street performer is located in a number of caches within the NetInf based network. The requests for the content by the users (flash crowd participants) are resolved to a number of copies cached in the network and these different copies are retrieved by the NetInf architecture used by the user's device. The OConS mechanisms that underlay the NetInf architecture assists the NetInf mechanisms to retrieve the content in the best possible manner.

3.2.1 OConS Multi-path connectivity services for ICNs

This use case related to an integrated NetInf and OConS focuses on the benefits achieved by users of the flash crowd when using multi-path. The devices of the users in the flash crowd are NetInf based ICN devices. These users require content from different sources and the NetInf nodes contains the OConS enabled convergence layer that will retrieve content from the identified sources. The OConS enabled convergence layer adopts the best multi-path strategy to retrieve the content. The multi-path strategy adopted is determined by the OConS based convergence layer that uses the OConS framework functionality.

The users in the flash crowd are connected to a number of networks. Since the NetInf based devices are capable of maintaining multiple attachments to the networks around the flash crowd, the request for content will result in delivery of a number of locators for the copies cached in the networks.

The sequence of actions that depict the use case/scenario involves a number of steps starting from the request of the content to the final receipt of the content on the users' devices.

- The content retrieval application requests for the content by providing a set of predicates;
- These predicates are resolved by NetInf to a number of locators;
- These locators are then used in selecting the appropriate multi-path strategy by the OConS framework;
- When the content is retrieved (part by part) they are made visible by the application to the user.

Each of the networks that a user's device connects to, has different characteristics. For example, some networks maybe highly congested and some may be quite expensive to use. These characteristics affect how the content is finally received at the user's device and what multi-path strategy is adopted.

3.2.2 Improve NetInf nodes connectivity by OConS DTN routing

OConS DTN routing is based on the social encounters occurred between nodes belonging to the flash crowd scenario. We assume that some of the people gathered in front of the street performer have downloaded (or locally produced) diverse content during the event: information about the artist, videos, etc. Among this crowd, some nodes have not direct access to the Internet, so they could get these pieces of information from neighbouring nodes instead. Also, not every single node is permanently connected to every other node in the group, and so, a DTN environment can be formed. If several people start moving away from the group, they could still spread the content to new neighbours, not physically present in the flash event. DTN routing would resolve the opportunistic path available to retrieve the requested content among contacting nodes through hop-by-hop interactions, which otherwise could not have been established. Introducing one of the NetInf concepts, the DTN routing could even be exploited in combination with 'content caching' so that requests from DTN nodes to get a specific content could be automatically served by intermediate neighbours which already retrieved that same content and stored it. As a result, a benefit on the overall performance of the communication network is obtained, as well as an alternative way of spreading the flash crowd to reach a larger amount of people.

NetInf could benefit from OConS service for a DTN topology where permanent connectivity cannot be assumed and the attachment of nodes to the network might be heterogeneous and in some cases, not possible. Using the OConS DTN routing a NetInf node without a 3G connection could improve its QoE in the flash crowd scenario: it could perceive almost the same available content than if having access to the Internet. It is not based on a new service, but on the fact that proximity-based connections among neighbours are used to spread content of the flash event. As a differentiating feature, OConS DTN routing is not only based on probabilistic estimation, but also combines information regarding the quality of a neighbour. As described in section 2.4, the OConS algorithm is aimed at the rating of how good is a neighbour as candidate for "best next hop" in the path towards a desired destination. According to this approach, the social behaviour of nodes belonging to the DTN (involved in the flash crowd) influences the routing decision, so that links among neighbours are reflected in the same way of a common social network in the path rating process.

3.3 Common OConS support for CloNe and NetInf

In the previous sections, we have presented the way some particular OConS services can be individually applied for either the CloNe or the NetInf use cases. There are, in addition, some other functionalities which are common to the two use cases, able to support mechanisms which are relevant for both of them. In particular, and considering the *Event with Large Crowd* scenario requirements, there might be situations in which users want to establish connectivity by means of wireless accesses. In this situation, the OConS framework provides an enhanced way to perform access selection, according to both the current network status and user policies and preferences. In the mobile access to the *Cloud* for instance, the end-user might be able to select a connectivity alternative between a relatively large number of options. In this sense, the information managed by OConS, together with the requirements coming from the corresponding end-user, shall be taken into consideration so as to promote an optimum selection. In addition, network and cloud operators (which in fact can be also seen as OConS users) might benefit from the available management facilities. Likewise, when a NetInf user is interested in retrieving a particular content, and once the corresponding functionality has identified the potential source nodes, OConS services can be triggered, so as to support the decisions process and establish the best possible connectivity. Eventually, this might also require some mobility-related actions and therefore the corresponding service might serve as well to deal with handover situations.

As a first step (in fact, this step is needed in order to call any OConS service), the various entities need to bootstrap. As a result of this, the Orchestration registry will compile information about all available entities, and services and mechanisms which can be used with them. In particular, regarding the access selection/handover procedures, all the management mobility-related functionalities (within IE, EE, and DE) shall be available within the appropriate nodes on both network and user side, so as to support the appropriate mobility management services.

In particular, these can embrace the provision of information about link quality or traffic load measurements, user profiles and requirements, thereby allowing OConS to take into account several parameters when a particular connectivity request happens. For instance, if the current connectivity configuration is not enough to address the requirements coming from the OConS user (for instance, a datacentre controller or a NetInf user), an automatic reconfiguration (i.e., OConS triggered) shall take place without requiring the interaction with the corresponding user.

As can be seen, the OConS Orchestration plays a key role in the overall process; first, it discovers the OConS entities which will be available and the services which can be offered by them, as well as the appropriate configurations and combinations. Afterwards, the OConS entities and mechanisms need to be properly combined, so as to appropriately offer the required services. This implies the specification of adequate interfaces and signalling procedures between the involved parties.

In this particular case, OConS offers an enhanced access selection service, which combines the requirements and policies of the end-user (and his/her services) with the particular conditions of the network, thus leading to a clear distinction between the nature of the information which is used. Below we further discus below this process.

- On the user side, OConS mostly provides information concerning the type of service requested by the user (together with its particular requirements) and her preferences or profiles. The main goal is to keep the QoE, as well as ensuring the seamless continuity of the service if needed. Various end-user policies or rules are to be considered, i.e., those establishing the particular needs of the end-user according to some pre-established profiles (e.g., energy saving, technology preference, pricing, security, etc.) and those giving the service/application requirements (e.g., quality needed and minimum networking resources). Likewise, the above parameters can be handled on different ways, either in isolation or combined, according to a specific configuration (for instance, they might be weighted so as to modulate the decision token within the end-node, being part of the overall and coordinated decision).
- On the other hand, the network is able to compile information about its current status as well as considering end-users' perspective (e.g. radio-link quality such as SNR/RSSI, network delay, quality profile, cost, etc.) and to take coordinated decisions considering a trade-off between QoS, QoE and network management goals (for instance, load balancing).

In this case, the access selection OConS service is conceived to deal with multi-homed/multi-interface devices, being able to handle flow mobility between different interfaces, to invoke multi-path functionality, to select and activate per-flow mobility anchors on network-side, and to set the right policies within user-terminals.

As can be seen from the above description, this OConS service is strongly based on the information provided by the various IEs; in this sense, the bootstrapping shall discover all the elements able to provide such information, and the mechanism shall be configured so as to establish which of them needs to be considered within the decision processes.

Once this is done, the DE must be able to interact first with the IEs, to retrieve the required pieces of information; furthermore, in order to cope with multiple provider/operator situations, the corresponding decision models need to be extended; for instance, we could configure the OConS mobility management service so that it is the end-user terminal that collects partial decisions coming from several operators to integrate them with the locally constructed decisions. Afterwards, the DE must be able to interact and with the corresponding EEs, so as to enforce the appropriate decisions both within the end terminals and on the network side.

The basic bootstrapping process, as well as the access selection and handover mechanisms, are currently under development. These specific functionalities of the OConS framework is generic enough so as to be used for various use cases and thus can provide benefits when the mobile access is considered in conjunction with either CloNe or NetInf. Accordingly, we can imagine that these access selection and mobility functionalities interact more closely with the applications/services provided within the datacentres and the datacentres/cloud "controllers" or by the NetInf architecture, assuring service continuity when necessary, better data-path efficiency through offloading and anchor selection, influencing the instantiation of applications/services within a given datacentre, helping to establish the optimum nodes (and corresponding connectivity) to retrieve the requested content and so on.

4 Final Demonstrator Design and Realization Aspects (Phase 2)

The scope of this chapter is the demonstrator design and prototype realization aspects internally in WP-C during the Phase 2 (i.e., till the end of the project). It covers the prototyping activities that focus on the two use cases (OConS for CloNe, and OConS for NetInf), as well as activities related to the validation of the OConS architectural framework through the implementation of some of its functionalities, i.e., OConS orchestration bootstrapping, generic OConS protocol, and generic Information Element IE.

It will discuss which OConS "features" and innovations are validated by the prototypes and demonstrations, which functions are implemented in addition or by modifying what was described in chapter 2, how the components are realized through specific platforms and software, and what is needed as a run-time environment for the demonstrations (e.g., application used, measurement and visualization tools, and so on).

4.1 OConS support for CloNe

4.1.1 Common OConS / CloNe Concepts

This demonstration aims at showing elastic networking in cooperation with cloud management by monitoring and controlling flows with 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-depnedent resource allocation algorithms, test the protocol variants between CloNe and OConS and evaluate the performance of different decision strategies in path selection across the domains.

First we give a sketch of the common OConS/CloNe architecture. 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 Cloud Controllers (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 Distributed Cloud Management Protocol (DCMP), which is based on OCCI and OCNI. Open Cloud Computing Interface (OCCI) is the a network protocol from Open Grid Forum for managing cloud computing infrastructure like storage and compute resources but does currently not offer a means to perform network resource management. This is where Open Cloud Networking Interface (OCNI) comes into play. Open Cloud Networking Interface (OCNI) is an extension of OCCI defined by CloNe to add network resources and their management to OCCI. The DCM configures resources for the datacentre domains as well for the connecting network domain.

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

Two domains of a cloud exchange their dynamic connectivity parameters using the Distributed Cloud Protocol (DCP). The DCP comes in two flavours for two different layers. The DCP-LL 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 [7]). The DCP-NL is introduced for purposes of network aspects between the datacentre (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 DCP-LL to configure the connectivity, the actual focus in OConS is on the DCP-Network Layer protocol and its mechanisms, where DCP-LL only is needed for configuring the datacentre - network interface (CE-PE) at link level.

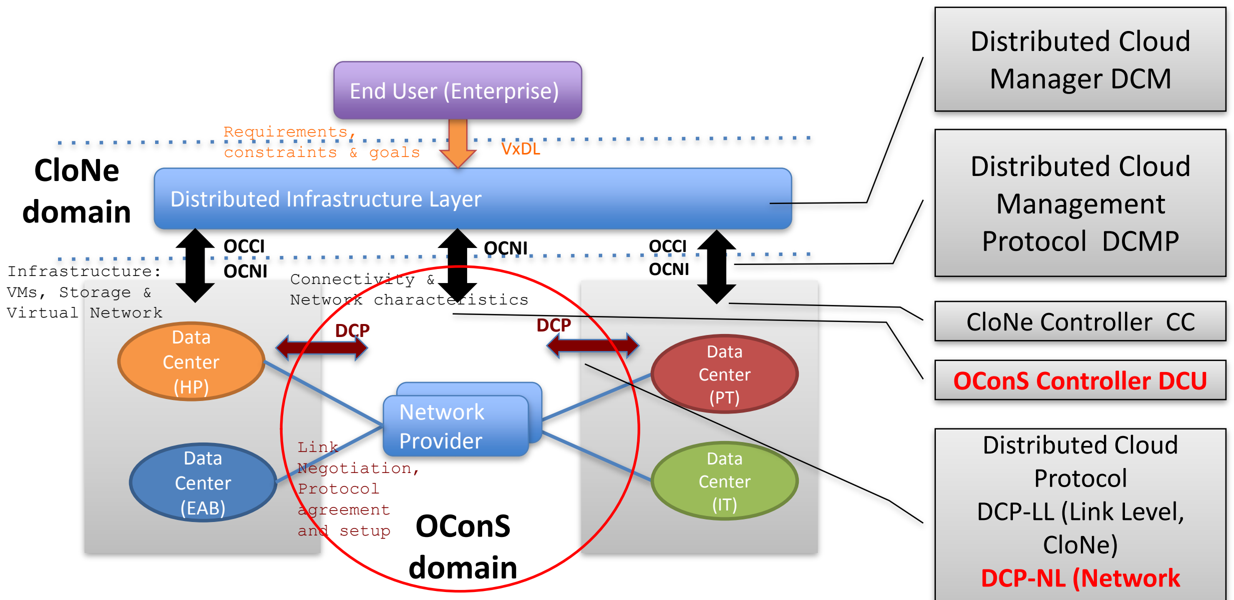

End User (Enterprise)
Requirements, constraints & goals
VxDL
CloNe domain
Distributed Infrastructure Layer
OCCI
OCNI
OCNI
OCCI
OCNI
Infrastructure: VMs, Storage & Virtual Network
Connectivity & Network characteristics
DCP
DCP
Data Center (HP)
Data Center (PT)
Network Provider
Data Center (EAB)
Data Center (IT)
Link Negotiation, Protocol agreement and setup
OConS domain
Distributed Cloud Manager DCM
Distributed Cloud Management Protocol DCMP
CloNe Controller CC
OConS Controller DCU
Distributed Cloud Protocol DCP-LL (Link Level, CloNe)
DCP-NL (Network Level, OConS)

Figure 4.1: OConS/CloNe Architecture

4.1.2 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, cf. section 2.1. Within the Mobile Cloud Datacentre, 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 Datacentre (DC) and Network (NW) domains constituting the Flash Network Slice (FNS) DCMP part. Also CloNe establishes the dynamic connectivity between DC and NW via CE-PE interface in its DCP-LL 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.

4.1.3 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.

Secondly the OConS-OConS interface between Network-CCs (intra-domain) monitors the load status of nodes and links resources, 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.

Thirdly 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 of sub-path information within the neighbour domain including its with metrics. And the inter-domain Network-CC interface performs the set up of a "foreign" domain path.

4.2 OConS support for NetInf

As OConS service realizations in support of NetInf we consider the multi-path content delivery for ICN and the mechanisms to improve NetInf connectivity by OConS DTN routing.

4.2.1 Multi-path Content Delivery for ICNs with OConS

The OConS functionality to select the best multi-path strategy is located in an extended NetInf convergence layer. This convergence layer uses the OConS framework components to decide the best paths to use. The OConS functionality includes the use of IEs that feed information to make decisions by DEs which are then implemented by the EEs located in the different NetInf based devices in the network. The orchestration of this new layer is as follows:

- NetInf based application requests for content (based on predicates)
- Part of this new convergence layer (together with existing NetInf functionality) performs the resolution phase. The resolution phase may result in a number of locations where the content is hosted.
- The OConS based convergence layer will be fed with the information from the IEs to make decisions by DEs. The decision will be in the form of an identified multi-path strategy.
- The EEs will implement the selected strategy in the NetInf nodes that are deployed with the OConS convergence layer to retrieve the content.

The NetInf implementation that operate on a client is configured to use the OConS enabled convergence layer. This will result in requests for content being received at the OConS enabled convergence layer. This convergence layer will then return the required content through the implementation of the selected multi-path strategy.

4.2.2 Improve NetInf connectivity by OConS DTN routing

A simple idea to demonstrate how OConS can support NetInf in the flash crowd scenario described, would consist on a DTN node (A) triggering a request to retrieve a video of the street performer, previously recorded by another DTN node (B) which is not present in the crowd any more. We can describe the situation in more detail, providing the necessary steps:

- Node A, which is requesting the video, has not an Internet access. It only has the possibility to establish peer-to-peer direct connections with other nodes in the vicinity (i.e via Bluetooth, Wi-Fi, etc);
- Node B, which recorded the requested video, has an Internet connection available, and its owner posted the content on his/her profile of one or several social networks;
- Some nodes of the flash crowd (C and D) have already retrieved the video, either from a friend's profile on the social network, or from a direct connection to Node B, who also shared it via Bluetooth;
- Node B went away from the spot of the street performance, and so it is not directly reachable any more. Nodes C and D are still in the flash crowd.

If a DTN routing scheme is not provided, Node A would not be able to get the video recorded by Node B, unless A specifically knows C or D and its owner asks its peers C or D to share it directly. If we have DTN routing implemented in nodes A, C and/or D together with the BPQ extension for DTN2, Node A could send a request for the video, and C or D would be able to automatically serve the content acting as NetInf caches, in a seamless way.

For the purpose of serving NetInf with the extension of DTN interactions, the DTN routing mechanism would need to rely on the BPQ extension for DTN2, so that nodes are able to act as NetInf caches and to process BPQ requests and responses for the content shared. Additionally, to spread a popular content related to the street performance, DTN caches could be prioritized when deciding which is the best next hop for a certain route or destination.

If DTN routing was to be implemented alone, the contents shared by people on the flash event could only be spread out on a peer-to-peer basis. That is, a DTN node would require to know which node has the content and ask for it to that specific owner. Combining the DTN routing with the NetInf caching on intermediate nodes, and using the BPQ request/response mechanism, a seamless communication is feasible. There is no need to be aware of the content location to trigger a request for it, since an intermediate node acting as a NetInf cache would automatically serve the content without forwarding the request to the initial destination node (the latter being one of the owners of the content, previously known by the requesting node).

4.3 OConS Framework Reusable Components

4.3.1 Orchestration Bootstrapping

Orchestration is one of the major OConS innovations since it supports the flexibility and openness of the complete framework, by allowing the dynamic configuration and instantiation of mechanisms and services based on the combination of different OConS components. In fact, for the sake of flexibility, orchestration must be present at all communication levels, thus bringing about three main orchestration levels, which are enumerated below:

- Link Connectivity Services: these orchestration services do not span further than one hop and are closely related to the physical and data-link layers.

- Network Connectivity Services: these services affect the routing and transport layers and are therefore independent from the end user application or service. They usually involve two or more nodes (i.e. end/access/core-nodes).
- Flow Connectivity Services: these are, as the previous ones, related to routing and transport layers, but in this case they show a higher dependence with the ongoing applications and services.

The orchestration functionalities are key enablers to allow the applications requesting certain service (OConS mechanisms) from the OConS framework and proceed with the appropriate configuration of the OConS components so as to offer (to the "requesters") the most appropriate service, fulfilling their needs. This orchestration task is carried out by means of some functionalities as follows:

- Service Orchestration Process (SOP) itself: it is in charge of coordinating and overseeing all the orchestrations tasks, keeping track of the mechanisms which are available OConS and what they can offer. These mechanisms are defined as a combination of OConS entities (i.e. IEs and EEs) which a particular DE requires in order to make a decision.
- Orchestration Service Access Point (OSAP): this is the external interface of OConS with its users (e.g. applications, CloNe, NetInf). The OSAP is the entry point for user requests (coming with some connectivity requirements), and the OConS can inform back the users about the available OConS capabilities or status.
- Orchestration Register (OR): this acts as a repository of all the OConS entities within the different nodes, possible considering resources from other nodes. These OConS entities can be further combined to instance various mechanisms (see [4]), which are also registered within the register.
- Orchestration monitoring: this collects the status of the OConS components and mechanisms launched within the different OConS-enabled nodes.

Some of the orchestration functionalities have been already implemented and tested [1]. The current components have been developed in *C++* and the communication between the various entities is socket-based. In this sense, each entity is an object of the appropriate class, encompassing the required functionalities; as an illustrative example, an illustrative generic class is presented further in the document (see Section 4.3.3). In terms of the corresponding connectivity services, the implementation is mostly focused on link connectivity, offering an enhanced Access Selection to the OConS user within a heterogeneous radio access environment. The orchestration of the various entities and components enable the use of an improved Access Selection service which can be integrated within each of the use cases described in Sections 4.1 and 4.2.

As depicted on Fig. 4.2, where the various F's represent a particular orchestration functionality, the implementation is focused on the registering functionalities for the time being. It is worth mentioning that we have assumed that the Access Selection service is also implicitly required by the node when it turns on, but for simplicity this is not shown in Fig. 4.2.

As was previously discussed, the OConS Access Selection encompasses three entities whose functionality has been depicted in Section 2.5. In a nutshell, this is defined by the DE in charge of selecting an interface, the IE providing the current link quality of the available connections and the EE able to enforce the connection to selected interface and access network; likewise, upon the request from the DE, the EE is also responsible for the required mobility executions functions, if necessary.

[1]The implementation described in this document is currently under development and therefore it might be adapted to the overall OConS design and specification.

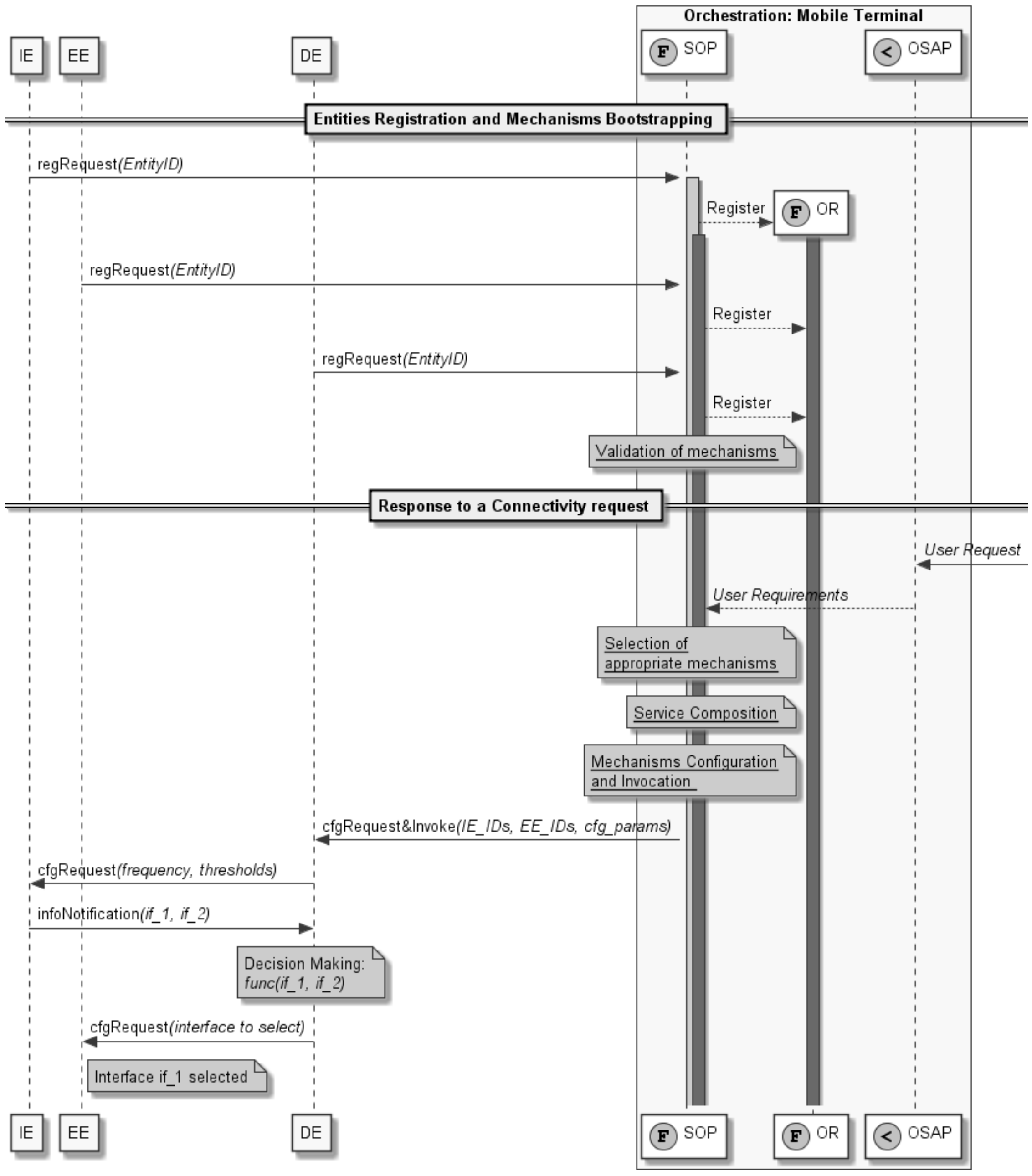

Orchestration: Mobile Terminal
IE
EE
DE
SOP
OSAP
Entities Registration and Mechanisms Bootstrapping
regRequest(EntityID)
Register
OR
regRequest(EntityID)
Register
regRequest(EntityID)
Register
Validation of mechanisms
Response to a Connectivity request
User Request
User Requirements
Selection of appropriate mechanisms
Service Composition
Mechanisms Configuration and Invocation
cfgRequest&Invoke(IE_IDs, EE_IDs, cfg_params)
cfgRequest(frequency, thresholds)
infoNotification(if_1, if_2)
Decision Making: func(if_1, if_2)
cfgRequest(interface to select)
Interface if_1 selected

Figure 4.2: OConS Orchestration

The first orchestration steps comprise the local discovery and OConS mechanism searching and identification, according to the data/information model. As can be seen, this resembles a traditional bootstrapping procedure. During this discovery process, all OConS entities register towards the SOP by using the *regRequest* primitive together with their information. This comprises both the entity type and the corresponding listening port, indicated on Fig. 4.2 as *entityID*, which are afterwards stored within the OR.

Anytime an entity is registered, the SOP checks whether there exists any DE which might be interested in that information; hence, upon the registration of any entity with the orchestration process, the SOP informs the DE about the way to reach the rest of entities (in the current implementation, this refers to the listening port) so as to enable a whole OConS mechanism or a combination of mechanisms, which could be invoked from now upon receiving the requests from the OConS users, or following a change in the network state.

Once the DE is aware of the available entities which make the mechanism feasible, it starts the normal Access Selection service procedures. In this case, the DE subscribes to the IE (by means of a *confReq* primitive) by setting the monitoring frequency as well as a threshold (these are explained in more detail on Section 4.3.3).

It is worth emphasizing that an OConS mechanism is not necessarily limited by a static number of entities, but it can be instantiated with a minimum set of them and be eventually improved with additional ones, including remotely.

Accordingly, once the basic connectivity has been reached (e.g., IP reachability), the current implementation allows the SOP within a node to trigger a remote discovery process, so as to become aware of the functionalities and entities which are available in remote nodes, as depicted on Fig. 4.3. In this case, the information gathered for the Access Selection service indicates that there exists an IE within each of the access elements to monitor the current load in the node; as a result of this new information, the 'terminal-based' access selection mechanism is able to take into account the network status.

Although the described mechanism considers that it is the mobile node which takes the ultimate decisions, the orchestration implementation does not limit this procedure to any type of node, nor a combination of them. In particular, for the aforementioned use case, the access selection decision might be jointly taken by the access nodes and the terminal itself on a distributed approach, providing more weight to one or the other so as to avoid conflicts. This could even consider the case in which the overall responsibility lies within the network nodes.

4.3.2 Generic OConS Protocol Library

The OConS Protocol Library will be in charge of providing low-level functionalities to allow the use of the OConS protocol and interfaces. It will first provide communication facilities, to make the entities able to communicate with each other. This includes management of direct communication with the local INC function, and helpers for message exchange. These are in charge of the encapsulation in the OConS headers and forwarding to the relevant entity or node through the INC. It will also support a callback-based structure.

Message content manipulation is also the realm of the library. It will abstract the on-wire TLVs structure from the entity code. This includes parsing helpers and accessors for the various data contained in a message as well functions to create and to add of data into the messages. This data can be of several basic types. Per-type setters and getters will also be provided by the library for basic OConS data type manipulation.

Finally, it will provide high-lever OConS functionalities to support the orchestration functional-

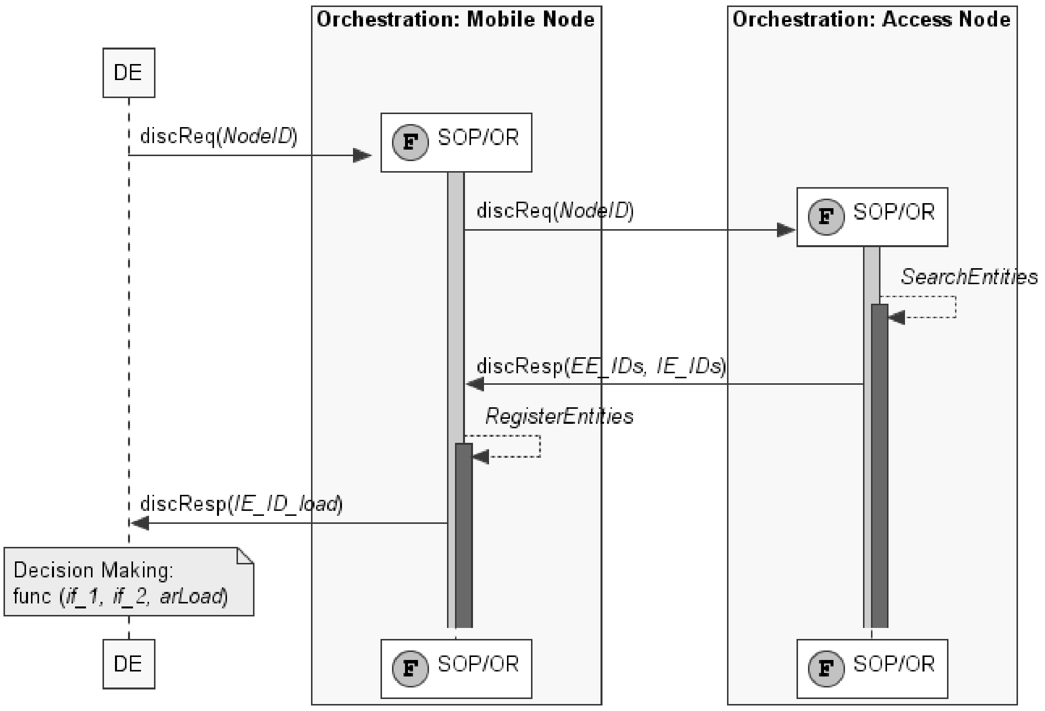


Figure 4.3: OConS remote Orchestration

ities such as mechanism identification, entities registration and bootstrap management. Conceptually, an OConS entity implemented based on this library would match the following pseudo-code:

- Register callback functions (message handling, periodic tasks, . . .) to the library;
- Create an entity-description structure;
- Open connection to INC (incl. registration and identification);
- Bootstrap.

With the message-receiving callback function being along the lines of the Algorithm 1.

Algorithm 1 Example pseudo code for message-handling callback with the OConS generic library.

```
Input: $m$, Message
  if type of $m$ is $X$ {Message parsing} then
    read first field as $f$
    if type of $f$ is $Z$ {Basic type manipulation} then
      store value of $Z$ in local data structure
    end if
  else if type of $m$ is $Y$ then
    create new message $m_2$ of type $A$
    add $Z$ into $m_2$ {Message building}
    $s \leftarrow$ sender of $m$ {Message parsing}
    send $m_2$ to $s$ {Message encapsulation and Inter-Node communications}
  end if
  do some entity-specific processing based on $Z$
```

The next section shows an example generic IE built around the services provided by the OConS protocol library.

4.3.3 Generic Information Element

According to the OConS architecture, the Information Element (IE) takes care of relevant information gathering, and later provides such information to the interested parties (in particular to the DE) according to the format which is being defined by the corresponding data/information model.

We assume that the IEs are used by one or more DEs so as to obtain the required information and are entities which have been orchestrated around a DE so as to define an OConS service (mechanism). Furthermore, they have been designed (and implemented) as both reactive and proactive entities, able to act either upon receiving a request from another entity or proactively sending notifications as configured (for instance threshold crossing, expiration of timer, etc).

- IE design: from design and implementation point of views, the IE includes a reference to the particular object responsible for providing the particular data (this inherits from a generic Data Source module (*DS*) which has been implemented as a C++ template).

  In fact, and due to the wide range of data that might be handled for the various OConS services at different levels (link, routing, flow), the IE (as can be seen from the previous discussion) has been decoupled into the OConS related procedures and the data acquisition mechanism (which might belong to a wide range of elements, nodes, technologies, and so on).

  In this sense, the outer class (so called *Information Entity*) implements the OConS interface and ensures that the configuration requested by the corresponding DE (timers, thresholds, etc.) is used, while the *DS* is responsible for getting the particular data from the original source. Figure 4.4 shows a high-level description of the structure of these two modules and their relationship.

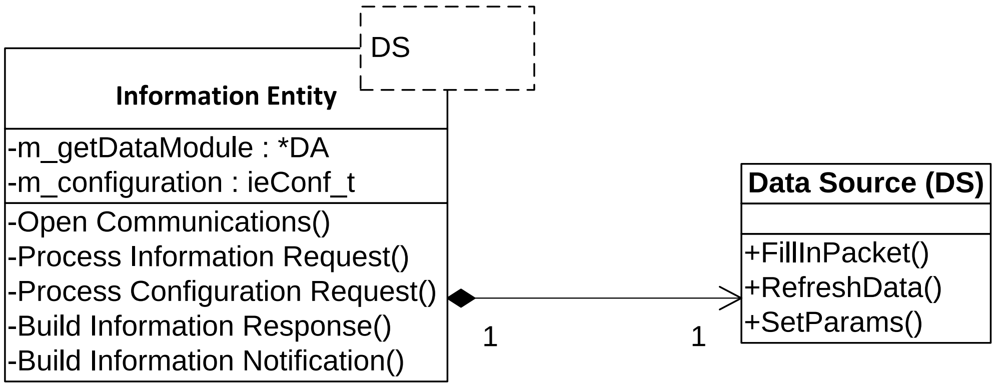

DS
Information Entity
-m_getDataModule : *DA
-m_configuration : ieConf_t
-Open Communications()
-Process Information Request()
-Process Configuration Request()
-Build Information Response()
-Build Information Notification()
1
1
Data Source (DS)
+FillInPacket()
+RefreshData()
+SetParams()

Figure 4.4: Structure of the implementation of the IE

  Last, but not least, the *DS* shall be aware of the OConS data/information model (at least *partially*) to properly represent the information (which might likely be provided by hardware/drivers/other specific elements) according to the OConS format. It shall offer the the *FillInPacket()* method to include such information in the corresponding packet at any particular moment.

- IE Configuration: Currently, the configuration procedure offers both request/response and subscription mechanisms; for the latter two possible configurations have been considered, and thus can be established within the *Configuration Request* from the “requester”, as seen below.

  1. Threshold crossing: this event happens when the metric surpasses a configurable percentage of the previous one. It does not control if a certain level has been reached but the variation of the corresponding measurements (link quality, load increase, etc.)

thus preventing hysteresis patterns. The inclusion of additional monitoring mechanisms might be straightforward[2] This threshold crossing event is checked by the *DS* whenever the *RefreshData()* method is called, so the “data” shall be first accordingly configured by means of the *SetParams()* method.

2. Notification timer: it defines an interval (in seconds) used to send periodic reports to the corresponding “requester”, this interval is set during the configuration phase.

4.3.4 Realization Plan

The following steps (milestones) are planned by end of the project, in order to develop the common prototyping support sketched above:

- OConS face-to-face meeting, 08. May 2012: discussion and alignment on orchestration and interfaces.
- Provide open entities code structure able to support low level communications functions and configuration so as to facilitate their following tailoring.
- July 2012: full specification of orchestration mechanisms/procedures and their intrefacing in deliverable D.C.2 “Architecture and Mechanisms for Connectivity Services”.
- Sept. 2012: implement orchestration capabilities in the OConS open entities and additional overall orchestration functionalities provided by the Generic OConS Protocol Library.
- OConS face-to-face meeting during the SAIL General Project Meeting, Bristol, week of 17. Sept. 2012: final alignment of the OConS-collaboration with NetInf and CloNe on use cases and interfacing aspects for a prototype supported by the Generic OConS Protocol Library.
- Final OConS demonstration of Generic OConS Protocol Library features at e.g. final SAIL General Project Meeting, January 2013, preferably for project review purposes.
- Jan. 2013: document OConS demonstration/prototyping of Generic Protocol Library 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”.

[2]This can be further discussed, but the use of relative measurements allows to limit the dependence on particular characteristics of the corresponding technology.

5 Integration and Cooperation Plan

This chapter describes the plans for the further work in the project regarding the cooperation that is envisaged for the project-wide demonstration with CloNe and NetInf.

We describe the proposed cross-WP use cases (i.e., the show case) planned for final demo, the level of integration/cooperation (e.g., conceptual, common use case, service interaction, interface interaction/interoperability, and so on) and the services and interfaces to be offered by the OConS workpackage to CloNe and NetInf.

5.1 OConS and CloNe: Elastic Video Networking

5.1.1 Use case description

The OConS - CloNe cross-WP use case that is planned for the final demo event will demonstrate the concept and the level of integration and interface interaction to show the benefits and the interoperability of the services and interfaces offered by the OConS workpackage to CloNe.

The foreseen video demo use case shows the steps necessary to run a video stream from a video server over a datacentre cloud performing the video processing to a video client.

CloNe is currently focused on the instantiation of potential cloud processing resources provided by VM containers in distributed datacentres and their interconnection using a network statically configured by Python Open Cloud Networking Interface (pyOCNI) functionality. What the OConS use case has in mind is a more dynamic instantiation of the network and an elastic modification of the connectivity based on monitoring the actually used resources by the video service (cloud application) and also the management functionality of the service soon after the network is instantiated. This monitoring can have several levels, like high level monitoring or low level monitoring and is not only used to show the current state of the network on a GUI but is also used for determining the optimal path through the network. So it is foreseen that if either the processing delay in the intermediate video processing datacentre, or the transport delay between different datacentres gets too high, the connecting flows carrying the application stream will be adapted.

5.1.2 Interaction between CloNe and OConS

The prototype implementation of the OpenFlow network controller establishes on demand OpenFlow paths in the network. These paths are defined by:

1. The entry points to the network, expressed as the interfaces for the network border routers the path will span between;
2. The network protocol or protocols that will be transported by the OpenFlow path.

The possibility to define the protocols allows the proof of concept of a 'network-as-a-firewall'-alike service. Requests for network paths are implemented with a pyOCNI server, which has been developed in the scope of the WP-D and can be used as an IE that stores the information regarding the paths established in the network. OConS needs the capacity and the utilization for links and nodes in the network. The values for different parameters will be stored at OConS side for statistics purposes and monitoring. Also OConS needs to know the network topology. As OConS sends the End Points (EPs) as IP addresses towards CloNe, there also has to be an interface to find out these EPs from CloNe.

5.1.3 CloNe and OConS Views

The application view of this use case is containing the video client, the video processing - no matter of location - and the video server. The CloNe view of this is just the video processing container available in several potential datacentres, the video server in another datacentre and the virtual connections between the datacentres via attachment points, as defined by OCNI. The OConS view is as follows: one attachment point (the CE) in the video 'cloud', two attachment points in the video processing DC and one attachment point in the video client.

5.1.4 Realization Plan

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

- CloNe-OConS face-to-face meeting, 24.April 2012: Discussion and alignment of cooperation approach, architecture and interface issues.
- May 2012: document interim state of CloNe-OConS cooperation plans in deliverable D.A.9. "Description of overall prototyping use cases, scenarios and integration points".
- OConS contribution to CloNe public demonstration at the "Future Networks and Mobile Summit", Berlin, 4.-7. July 2012: concept of inter-CloNe-OConS architecture, interfaces and data models (extended OCNI).
- CloNe-OConS face-to-face meeting and code sprint session during the SAIL General Project Meeting, Bristol, week of 17. Sept. 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".

5.2 OConS and NetInf: Multi-path Content Delivery for ICNs with OConS

5.2.1 Description

The OConS multi-path strategy selection functionality is required to be integrated within a convergence layer used by NetInf. A number of convergence layers are being conceived by WP-B to be used with NetInf. This work intends to use one of these convergence layers to perform the multi-path strategy adoption.

The multi-path functionality is mainly contained in the convergence layer itself. The name resolution phase in NetInf, also done through the convergence layer, results in obtaining a number of locators. These locators are then used by the convergence layer to retrieve the content over the multiple attachments that are under the control of the convergence layer.

WP-B is currently focused on defining and developing a UDP-based convergence layer. This convergence layer is planned to be designed in a way where content are retrieved as segments (chunks). Therefore, the multi-path functionality defined in WP-C aims to utilize this segmenting capability of the UDP convergence layer to perform multi-path actions.

5.2.2 Realization Plan

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

- NetInf-OConS presentation and discussions, 18. April 2012: Presentation of 2 alternative proposals on how OConS functionality will be integrated into NetInf.
- May 2012: document interim state of NetInf-OConS cooperation plans in deliverable D.A.9. ”Description of overall prototyping use cases, scenarios and integration points”.
- 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:
- Final NetInf-OConS demonstration at e.g. final SAIL General Project Meeting, January 2013, preferably for project review purposes.
- Jan. 2013: document NetInf-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”.

5.3 OConS and NetInf: OConS routing for DTVideo

5.3.1 Description

The OConS implementation of a DTN routing mechanism is based on the social interactions among 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. The final purpose of integrating this mechanism 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 requires the DTN routing mechanism to rely on the BPQ extension for DTN2, so that nodes are able to act as NetInf caches and to process BPQ requests and responses for the content shared.

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

- Individual prototype implementation: NetInf has implemented 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 the implementation of the DTN routing 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).

If DTN routing was to be implemented alone, the contents shared by people on the flash event could only be spread out on a source/destination basis. That is, a DTN node would require to know which node has the content and specifically ask for it to that source (either the content owner/generator or a node publishing its available content files). Combining the DTN routing with the NetInf caching on intermediate nodes, and using the BPQ request/response mechanism, a seamless communication is feasible. Furthermore, if BPQ extension is individually considered, the social behaviour of DTN nodes would not be exploited as routing information, which could result in a sub-optimal performance due to the demanding nature of a flash event. If some people present in the flash crowd belong to some social network and are connected with other friends, they would presumably have historical contact information about reciprocal encounters, and that is definitely useful for routing decisions in the flash crowd scenario.

5.3.2 Realization Plan

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

- NetInf-OConS presentation and discussions, 23. April 2012: Discussion and alignment of cooperation approach, prototype goals and interface issues.
- May 2012: document interim state of NetInf-OConS cooperation plans in deliverable D.A.9. "Description of overall prototyping use cases, scenarios and integration points".
- June-September 2012: work in the implementation of OConS DTN routing so as to serve the DTVideo application, based on BPQ extension. Maintain discussions all along this period in order to combine social based routing with an adapted NetInf client for DTN environments (focus on flash crowd scenario).
- NetInf-OConS face-to-face meeting and code sprint session during the SAIL General Project Meeting, Bristol, week of 17. Sept. 2012: finalized descriptions of inter-NetInf-OConS demonstration, interfaces and realization of the prototype. Align use case realizations across WPs.
- Final NetInf-OConS demonstration at e.g. final SAIL General Project Meeting, January 2013, preferably for project review purposes.

- Jan. 2013: document NetInf-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 Conclusion

We have started by presenting the prototyping and experimentation clusters based on partners' activities, the use cases from first phase of the project as introduced in D.A.1 (i.e., the bottom-up approach), and then focusing mainly on the two updated use cases OConS for CloNe and OConS for NetInf, as described in D.C.1-Addendum.

Accordingly, we have described in detail the prototyping activities as implemented in the first phase based on the partners' contribution. We have thus explain the component architecture for each prototype and given its realization details, i.e., sub-components, modules, platform, programming language and protocols used or adapted.

We have then continued by decomposition and analysis the two main OConS use cases from D.C.1-Addendum, identifying the functionalities and the prototyping components to be employed in order to support and bring into reality these use cases. Based on this analysis, we have come up with a cooperation plan for the next phase of the project, and have presented the functions that are to be implemented in addition or by modifying what was done by the WP-C partners in the first phase.

Finally, we have also planned the future work needed for demonstrating project-wide scenarios, thus continuing and increasing the cross-WPs cooperation from Cloud Networking and, respectively, NetInf perspectives.

Among the lessons learnt, we have seen that it is important to have quite early in the project not only the clusters with inter-related prototyping topics coming from the partners, but also to focus on a couple of driving use cases which are commonly agreed among several partners.

Another lesson learnt is the fact that if a certain level of prototyping integration is sought, then we need to boil down to common interfaces and information model in the initial stage of the prototyping task.

When it comes to the next steps, we will have in a few months the refinement and the finalisation of the OConS architecture in D.C.2 deliverable, so we further expect that this will influence the prototyping activities by bringing in further coherence and clarification, notably for the re-usable components such as the autonomous OConS orchestration process and intra-/inter- domain OConS interfaces.

Likewise, the results from the experimentation and prototyping activities will be collected and presented in the D.C.5 (due at the end of the project) demonstrating concrete applications for Connectivity Services. Moreover, our work on prototyping will contribute to the project-wide deliverable D.A.9 where we intend to focus more on how prototyping pieces and components from NetInf, OConS, and CloNe fit together in a common project-wide story.

Before that, we have also planned intermediate demonstrations to show our prototypes related to the two main use cases, i.e., OConS for CloNe at the FuNeMS event in July 2012, as well as OConS for NetInf at the MONAMI conference in Sept. 2012.

List of Acronyms

AR	Access Router
AJAX	Asynchronous JavaScript and XML
ASF	Advertising Supporting Function
BPQ	Bundle Protocol Query
CCN	Content-Centric Networking
CC	Cloud Controller
CE	Customer Edge
CloNe	Cloud Networking
DCC	Domain Control Client
DCC-B	Domain Control Client - Border Forwarding Node
DCC-I	Domain Control Client - Interior Forwarding Node
DCM	Distributed Cloud Manager
DCMP	Distributed Cloud Management Protocol
DCP	Distributed Cloud Protocol
DCUP	Domain Control Protocol
DCU	Domain Control Unit
DC	Datacentre
DE	Decision Making Entity
DFC	Dynamic Flow Control
DTN	Delay Tolerant Network
EE	Execution and Enforcement Entity
EP	End Point
FNS	Flash Network Slice
GUI	Graphical User Interface
IaaS	Infrastructure as a Service
IDE	Integrated Development Environment
ICN	Information-Centric Networks
IE	Information Management Entity
INC	Inter-Node Communication
JSON	JavaScript Object Notation
MN	Mobile Node

NetInf	Network of Information
NE	Network Element
NW	Network
OCCI	Open Cloud Computing Interface
OCNI	Open Cloud Networking Interface
OConS	Open Connectivity Service
OE	Orchestration Entity
OR	Orchestration Register
OSAP	Orchestration Service Access Point
OSGI	Open Service Gateway Initiative
PCE	Path Computation Entity
PE	Provider Edge
PRoPHET	Probabilistic Routing Protocol based on Historical EncounTers
pyOCNI	Python Open Cloud Networking Interface
QoE	Quality of Experience
QoS	Quality of Service
RAN	Radio Access Network
REST	Representational State Transfer
RPE	Request Processing Entity
RRH	Remote Radio Head
SAIL	Scalable Adaptive Internet Solutions
SOP	Service Orchestration Process
TED	Traffic Engineering Database
TLV	Type Length Value
VM	Virtual Machine
WP	workpackage

List of Figures

Bibliography

[1] SAIL. The SAIL project web site. http://www.sail-project.eu/.

[2] SAIL. 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.

[3] SAIL. 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.

[4] SAIL. 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.

[5] SAIL. 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.

[6] SAIL. Content Delivery and Operations. Deliverable FP7-ICT-2009-5-257448-SAIL/D.B.2, SAIL project, May 2012. Available online from http://www.sail-project.eu.

[7] SAIL. 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.

[8] Fariborz Derakhshan, Heidrun Grob-Lipski, Horst Roessler, Peter Schefczik, and Michael Soellner. On converged multidomain management of connectivity in heterogeneous networks. *Future Network and Mobile Summit (FuNeMS2012), Berlin.*, July 2012.

[9] Jeff McAffer and Jean-Michel Lemieux. Eclipse rich client platform: Designing, coding, and packaging java applications, 2005.

[10] Beacon OpenFlow controller. http://www.openflowhub.org/display/Beacon/. Last seen on Wed, 09 May 2012.

[11] Bob Lantz, Brandon Heller, and Nick McKeown. A network in a laptop : Rapid prototyping for software-defined networks. *Electrical Engineering*, pages 1–6, 2010.

[12] Jon Watson. Virtualbox: bits and bytes masquerading as machines. *Linux J.*, 2008(166), February 2008.

[13] Paul Barham, Boris Dragovic, Keir Fraser, Steven Hand, Tim Harris, Alex Ho, Rolf Neugebauer, Ian Pratt, and Andrew Warfield. Xen and the art of virtualization. In *SOSP '03: Proceedings of the nineteenth ACM symposium on Operating systems principles*, pages 164–177, New York, NY, USA, 2003. ACM Press.

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

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

[16] J. M. Cabero I. Urteaga S. Perez-Sanchez, N. Errondosoro and I. Olabarrieta. Human routines

optimise routing in disrupted networks: the hurry protocol. *ACM MobiCom Workshop on Challenged Networks CHANTS 2012*, Submitted.

[17] Communication System Design. Android application version 3 of the project bytewalla, 2010.

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