ABSTRACTION LAYER FOR IMPLEMENTATION OF EXTENSIONS IN PROGRAMMABLE NETWORKS

Collaborative project co-funded by the European Commission within the Seventh Framework Programme

Grant agreement no:	317880
Project acronym:	ALIEN
Project full title:	"Abstraction Layer for Implementation of Extensions in programmable Networks"
Project start date:	01/10/12
Project duration:	24 months

Deliverable D5.1
Usage Scenario Description

Version 1.2

Due date:	31/03/13
Submission date:	19/04/13
Deliverable leader:	UCL
Author list:	Richard G. Clegg (UCL), Jon Matias (UPV/EHU), Matteo Gerola (CreateNet), Eduardo Jacob (UPV/EHU), Maider Huarte (UPV/EHU), Artur Binczewski (PSNC), Bartosz Belter (PSNC), Krzysztof Dombek (PSNC), Artur Juszczyk (PSNC), Łukasz Ogrodowczyk (PSNC), Iwo Olszewski (PSNC), Damian Parniewicz (PSNC).

Dissemination Level

☒	**PU:**	Public
☐	**PP:**	Restricted to other programme participants (including the Commission Services)
☐	**RE:**	Restricted to a group specified by the consortium (including the Commission Services)
☐	**CO:**	Confidential, only for members of the consortium (including the Commission Services)

Abstract

This document describes the use cases for the Content Centric Network (CCN) based experiments within the ALIEN project. The experiments will combine CCN with OpenFlow using the existing work from the COntent NETwork (CONET) project. This provides integration between CCN and OpenFlow. The document has two purposes within the ALIEN project: firstly to define what OpenFlow functionality needs to be implemented within each piece of ALIEN hardware to successfully complete each experiment and secondly to describe the experiments in sufficient detail as to give guidelines as to the work which needs to be achieved in order to successfully conduct experiments using CCN and OpenFlow on ALIEN hardware.

Project:	ALIEN (Grant Agr. No. 317880)
Deliverable Number:	D5.1
Date of Issue:	*19/04/13*

Table of Contents

Executive Summary 6

1 Introduction 7

2 Introduction to Content Centric Networking 8
2.1 Information Centric Networking Schemes 8
2.1.1 Data Oriented Network Architecture (DONA) 9
2.1.2 Publish / Subscribe for Internet Perspective (PSIRP) 9
2.1.3 Internet Indirection Infrastructure (i3) 9
2.2 Content Centric Networking 10
2.2.1 Content Centric Networking addresses and routing 10
2.2.2 Content Caching in CCN 11
2.3 CCNx reference implementation 11

3 The COntent NETwork (CONET) 12
3.1 The CONET proposal 12
3.2 CONET implementation with OpenFlow 16
3.3 CONET implementation deployed in the OFELIA testbed 18
3.4 ALIEN extensions to CONET 21
3.5 OpenFlow functionality required by CONET 22

4 Testbed description 27
4.1 ALIEN slice at OFELIA 27
4.2 Testbed resources and connection to OFELIA 29
4.2.1 PSNC testbed 30
4.2.2 UPV/EHU testbed 31
4.2.3 PUT testbed 32
4.2.4 Dell Force10 testbed 34
4.2.5 UCL testbed 35

5 OpenFlow CCN use cases 36
5.1 Basic scenario 38
5.2 Alternative routes scenario 39
5.3 Cache scenario 40

5.4 Caches and alternative routes scenario 40

6 Requirements 41

6.1 OpenFlow requirements 41

6.2 Requirements from CONET extensions to the Floodlight controller 42

6.3 Experimental harness requirements 43

7 References 44

Figure Summary

Figure 2-1 The CCN protocol stack compared to the traditional IP stack 10
Figure 3-1 CONET Architecture [OF ICN] 13
Figure 3-2 Sequence of operations [ICN SDN OF] 19
Figure 3-3 CONET testbed in OFELIA [ICN SDN OF] 20
Figure 3-4 ALIEN improvements to the current CONET development 22
Figure 4-1 OFELIA Facility 27
Figure 4-2 ALIEN slice topology 28
Figure 4-3 CCN Overlay 29
Figure 4-4 Interconnections and internal details of PSNC Network Processor laboratory 31
Figure 4-5 Interconnections and internal details of PUT laboratory 33
Figure 4-6: Force 10 testbed 34
Figure 5-1 Schematic diagram for the CCN use case 37
Figure 5-2 CCN acts as an overlay topology over the ALIEN test network 38
Figure 5-3 Alternative routes scenario 40

Table Summary

Table 3-1 OpenFlow capabilities 26
Table 4-1 Connection to OFELIA 30

Executive Summary

The ALIEN hardware at each partner site will be tested over the OFELIA testbed using a common control interface based on OpenFlow. All partners will either implement the OFELIA control framework locally (in some cases this is already done) or connect their testbed directly to an OFELIA island. The test application (Content Centric Networking) will be implemented building upon the work from the existing work on COntent NETworks (CONET) [CONET] which integrated OpenFlow and CCN.

Four test scenarios for CCN are described, a base scenario with content delivery between end nodes at different partner sites; a caching scenario where auxiliary caches are deployed; a flow diversity scenario where CCN and non-CCN flows are pushed onto separate paths by the OpenFlow controller and a final scenario which combines the auxiliary caches and the flow diversity.

The requirements for the experiments to succeed are twofold. The most important requirements are on the OpenFlow implementation by each partner. These are critical to the success of the project and this document list those OpenFlow properties which must be implemented in order for the test scenarios to function. Secondary requirements are extensions to the OpenFlow controller which will be necessary in order for the CONET CCN extensions to work in the way required by the project.

1 Introduction

This document describes the scenarios which will be tested during the ALIEN project. The project will develop OpenFlow capability for a number of pieces of network equipment (dubbed ALIEN hardware). The functionality of the OpenFlow support will be tested using the scenarios described in this document. The chosen application for testing was Content Centric Networking (CCN) and this is described and placed in context in section 2. Section 3 describes the CONET project which allows an interaction between OpenFlow and CCN. It is this project which will be key to the CCN deployment within the ALIEN project. Section 4 will describe the partner testbeds, how they will connect to the OFELIA facility (both at data plane and control plane) and how this will be used to provide tests across all test beds within the project. Section 5 will describe the OpenFlow capabilities to be tested and the CCN scenarios which will test them. Finally, section 6 will give a description of the requirements for the tests to succeed. These are requirements both for the OpenFlow implementation at partner sites and for the OpenFlow controller which will provide control for the hardware and a testbed harness for the experiments.

2 Introduction to Content Centric Networking

This chapter discusses content centric networking (CCN). The term is used in three separate ways which this document distinguishes. Firstly, and most generally, the term CCN is used to refer generically to the idea of networking based upon addresses related to content and not to the machine or machines where the content is stored – phrases like Information Centric Networking, Named Data Networking and Content Based Networking are used to mean the same thing. Secondly, the term is used to refer to a specific architecture for this developed at Palo Alto Research Centre (PARC). Finally the term is used to refer to a specific reference implementation of the CCN protocol, CCNx [CCNx] which is a software implementation with API. This chapter will begin by in section 2.1, reviewing other schemes for information centric networking. In section 2.2 the CCN scheme will be considered and, finally, in section 2.3 the CCNx implementation will be briefly discussed. To distinguish between them, the phrase Information Centric Networking will be used to mean the generic idea of networking based upon content names, the term CCN will be used to mean the PARC architecture for this and CCNx will be used to mean the specific C and Java implementation developed by PARC.

Major issues which are identified as important for this type of scheme are:

- Availability: Ensuring content reaches the user quickly from the nearest possible location on the network.
- Security: Ensuring that the content which the user retrieves matches the content which the user named when making the request.
- Location dependence/persistence: Ensuring that content remains available even when the original location which provided the content is reconfigured or removed.

2.1 Information Centric Networking Schemes

The core idea of Information Centric Networking is the idea that a user is normally interested in accessing information, not in accessing a specific machine on a network. When the user, for example, looks for a web page by URL, the user is interested in the information on that web page. The fact that the URL translates to an IP address and hence to a specific machine and then (via the http protocol) to a specific file (or files) on that machine is a historic accident. Increasingly, thanks to mechanisms like peer-to-peer networking, content distribution networks and other technologies mean that a given piece of content is available in multiple physical locations on the network. Information Centric Networking is a paradigm which, in essence, routes user requests based upon a content identifier to the most convenient copy of that content. This section covers some other Information Centric Networking architectures.

2.1.1 Data Oriented Network Architecture (DONA)

DONA was proposed in [DONA] and uses flat addresses and self-certifying names to authenticate data. DONA replaces the TCP/IP solution of Domain Name Servers (DNS) with resolution handlers (RHs). RHs use content requests (FIND) from content consumers and content provisioning messages (REGISTER) from providers. Because of the flat namespace, each RH holds a large forwarding table which provides next hop information for every possible piece of content. The forwarding tables are used to locate the nearest location for the content and then standard IP routing is used to deliver content to the consumer.

2.1.2 Publish / Subscribe for Internet Perspective (PSIRP)

PSIRP solves a similar problem to CCN and uses a publish/subscribe approach [LIPSIN][PSIRP]. PSRIP aims to address security, routing, wireless access and mobility. Instead of users and providers, PSIRP uses the terminology publishers and subscribers to refer to creators and consumers of content. PSIRP relies on components for forwarding, routing, rendezvous and caching. Information identifiers are a vital part of PSIRP, there are four classes:

- Application identifiers – used directly by publishers and subscribers.
- Rendezvous identifiers – bridge between different levels of identifier.
- Scope identifiers – limit which parts of a network certain information can be routed through.
- Forwarding identifiers – define transit paths which routes information across a network.

In PSIRP, nodes are split into branching nodes (route subscription messages and cache content), topology nodes (intra domain topology and routing information including load balancing), forwarding nodes (forward information) and rendezvous points (which locate published information within the network).

2.1.3 Internet Indirection Infrastructure (i3)

The i3 architecture [I3] is based upon a rendezvous abstraction upon which communication services can be built (e.g. multicast, anycast and mobility). It tries to combine IP layer solutions with deployable overlay based solutions. The architecture is similar to IP multicast but uses Chord Distributed Hash Tables (DHT) and packet identifiers. The DHT is used to identify the appropriate provider for the data. Because the sender inserts identifiers in the data and sends them via a rendezvous point, a receiver can continue retrieving the data even if their IP address changes. Similar mechanisms use this identifiers for multicast, anycast or even (via stacked identifiers) service composition (where the data must undergo transformation before returning to the consumer).

2.2 Content Centric Networking

CCN was introduced in the paper [NDN] by Van Jacobsen et al. The aim is to have a new "waist" to the network consisting only of content chunks and independent of the underlying transmission mechanism. By doing this correctly certain properties are "built in" from the ground up, for example security (in the sense that the content received is cryptographically guaranteed to be the content requested).

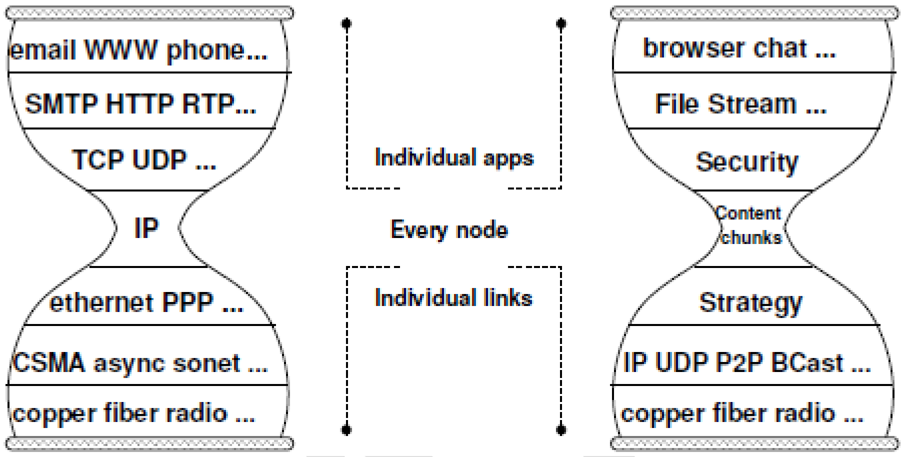


Figure 2-1 The CCN protocol stack compared to the traditional IP stack

Figure 2-1 shows how CCN compares to the well known TCP/IP protocol stack. CCN packets come in two main types, Interest and Data. The Interest packet is generated by a consumer who is searching (by name) for content. The data packet is generated by a CCN server in response and returns that data requested by the interest packet.

2.2.1 Content Centric Networking addresses and routing

CCN names are hierarchically structured but opaque to the network in the sense that routers do not know the meaning of a name only where name boundaries exist. Name conventions are agreed between produces and consumers and names do not need to be globally unique unless global reach is required for their propagation. Each data packet has a digital signature based upon its name which guarantees that the packet has been produced by the publisher. The issue of trust management has been given considerable attention in CCN.

Packet routing and forwarding is based upon the packet names which avoids problems related to NAT traversal, mobility, address space exhaustion and address management. A router announces name prefixes covering the data that router can serve. The hierarchical nature of the names means that route aggregation can take place and each router does not require a full entry for each content packet. In the full paper [NDN] attention is given to the issue of scalability.

Routing occurs through two mechanisms. The Forwarding Information Base (FIB) is populated by the routing protocol and tells each router where content may be found. This may be down multiple routes. Interest packets are routed using the information in the FIB. The Pending Interest Table (PIT) tracks all requests for content a CCN router has made but not yet satisfied and the interface (in fact the term Face is used as the interface is generalised and may be, for example, a multicast group) from where the request came.

2.2.2 Content Caching in CCN

Key to CCN is the concept of on-path caching. Each CCN node in the system contains a content store. As the node forwards packets according to received interests, it caches a copy. This is justified by the fact that modern routers already contain considerable memory space (which is not expensive) and, therefore, it is not unreasonable for content to be cached at a number of intermediate nodes.

The cache is organised by a simple Least Recently Used (LRU) policy. Every time a piece of content is requested it is effectively “refreshed” in cache and that content least recently refreshed drops out of memory if new content arrives. In this way it is hoped that the CCN nodes will be able to propagate popular content to where it is requested on demand in a self-organising manner.

2.3 CCNx reference implementation

CCNx is a specific reference implementation of CCN created at PARC and hosted at http://www.ccnx.org/. The code is in several parts. The main part, a CCN daemon is C based and runs on unix as ccnd. This is a complete implementation of the CCN protocol over an existing networking cache. The basic daemon listens on a port for CCN protocol messages and responds appropriately. Other helper programs are used to set up and control the daemon, for example, causing interface (faces) on the daemon to connect to another ccnd running on a different machine. In this way, the CCN daemons can run as a layer 7 overlay network over the existing TCP/IP network. Faces can be created using either TCP or UDP sockets. Protocol messages are in xml (highly compressed for efficiency).

Control programs are used to configure, connect and request information from the running ccnd. These programs can also request content through the CCN overlay. For example `ccngetfile ccnxname filename` requests to the daemon that a given piece of named content (named ccnxname) is downloaded from the overlay and saved as filename.

The Java library provides a more complete interface with much more functionality and easier ways to access functions of CCN but requires the daemon written in C to implement a CCN node. An android implementation also exists.

3 The COntent NETwork (CONET)

This section describes the COntent NETwork (CONET) project, how this approach has been adapted to be implemented in an OpenFlow scenario and how the CONET work will be extended in the ALIEN project. Firstly some details are given about CONET itself. Section 3.1 describes the CONET proposal, section 3.2 the CONET implementation with OpenFlow, and section 3.3 the CONET implementation deployed in the OFELIA testbed. Section 3.4 describes the extensions to CONET which the ALIEN project will require and section 3.5 describes those OpenFlow functions which CONET requires and which, therefore, must be implemented on ALIEN hardware during the project.

3.1 The CONET proposal

CONET [CONET] is one of the solutions proposed for Information Centric Networking (ICN). In fact, the CONET is an ICN framework able to integrate different specific solutions for the different ICN components. In general, an ICN solution should support at least the following procedures: (1) Content centric request/reply paradigm for data distribution; (2) Route-by-name operations; (3) In-network caching; and (4) Security embedded in the content. The CONET framework is modular and supports different solutions for fundamental issues like: (1) data naming; (2) name-based routing; (3) name-based interest forwarding; (4) content-data forwarding; or (5) transport protocols.

In the CONET framework, each piece of content (also known as chunk of data) has a unique name. The users need to request each chunk of data by means of an interest message. The forwarding process of this interest message is based on the name of the requested content. When finally the data-content is sent back to the user, the intermediate nodes can cache the chunk of data. On subsequent request of the same chunk of data, any intermediate node which has already cached that specific chunk is able to send the data back. The interest message is not further propagated.

The coCONET [ICF SDN] is a specific implementation of ICN based on the CONET framework. In this solution, specific procedures and alternatives have been selected. For instance, the forwarding mechanism of the interest messages relies on the lookup-and-cache strategy. In such strategy, the network nodes send a query to the Name Routing System (NRS) to obtain the next-hop node and cache the response for later on. The NRS provides the necessary routing information to all the nodes. The network nodes only perform forwarding operations, while the NRS is responsible for the creation and maintenance of the forwarding rules and the control plane logic.

Regarding the CONET approach, some of its main features are described next. It is stateless, which means that the network nodes do not need to maintain any state. It is able to limit the size of the name-based routing tables, which is relevant for addressing scalability issues. CONET proposes to integrate the content awareness in the IP networks as a new header option in IPv4 or as an extension header in IPv6.

Concerning the network architecture (Figure 3-1), CONET is a system that interconnects one or several CONET Sub Systems (CSSs). Each CSS contains CONET nodes, which will be defined later. Inside a CSS, the data is transferred by means of an under-CONET technology.

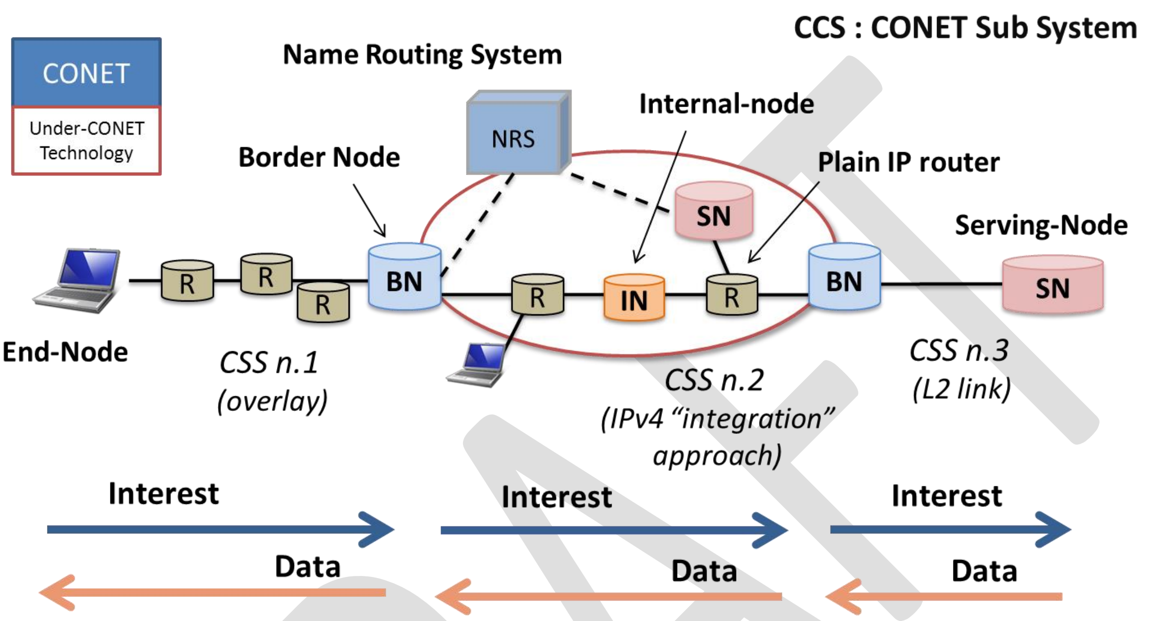


Figure 3-1 CONET Architecture [OF ICN]

With regard to the protocol stack, each and every CONET node has two layers, the CONET layer and the under-CONET technology. In the case of end-nodes, they also include the transport-level functionality. Starting from the lower layer, the under-CONET technology is the underlying network technology related to the CSS with an autonomous and homogeneous addressing space. On top of this, the CONET layer is a connectionless technology, which handles CONET Information Units (later discussed) and carrier packets. It also provides caching functionalities and security at content level. In the case of end-nodes, there is an extra layer, the transport layer, which adds reliability and flow control. It also provides an API to the upper layers (application).

Depending on the selected under-CONET technology, the approaches can be categorized in three different groups (1) the overlay approach, which uses point-to-point links (e.g. PPP or UDP/IP overlay tunnels) to build the logical topology of the ICN; (2) the clean slate approach, which relies in layer-2 networks (e.g. Ethernet or MPLS LSP); (3) the integration approach, which is built on top of layer-3 networks (e.g. IPv4 or IPv6).

There are two different types of CONET Information Units (CIU)s: the Interest CIU and the Named-data CIU. The Interest CIUs are used by users to request for named-data or a set of bytes of a named-data CIU. The Named-data CIUs are used to send each chunk of a named-data. In order to get a better performance from the under-CONET technology (e.g. improving the forwarding speed or avoiding further segmentations due to the MTU), the carrier-packets are introduced as the CONET data units used to carry all the CIUs.

In order to give a top-down vision, the named-data is the whole content (e.g. file, movie,…) that the users want to obtain. The named-data is divided in one or several chunks of data, each of which is encapsulated in a named-data CIU with additional information. The named-data CIU is divided in one or several carrier-packets, which are under-CONET technology specific, to improve the overall performance. In fact, the carrier-packets are the data-units of the under-CONET technology.

As previously mentioned, the CONET nodes can be logically classified in the following entities (Figure 3-1), with specific functionalities to implement: (1) the end-nodes; (2) the serving-nodes; (3) the border-nodes; (4) the internal-nodes; and (5) the name routing system. The end-nodes (ENs) are the final entities which request for the named-data by sending Interest CIUs. The serving-nodes (SNs) are responsible for storing, advertising and providing the named-data. The SN split the named-data in one or more named-data CIUs, which are transferred in one or multiple carrier-packets. The border-nodes (BNs) are located at the border between the CSSs. They are responsible for caching the named-data CIUs and forwarding the carrier-packets by means of CONET routing mechanisms (e.g. routing-by-name or inter-CSS source routing, later discussed). The internal-nodes (INs) are located inside the CSS. They are responsible for caching named-data CIUs internally in the CSS and forwarding carrier-packets by using under-CONET routing mechanisms. Finally, the name routing system (NRS) assists CONET in performing the routing operations. As previously mentioned, the NRS replies to network nodes with the information about the next-hop node when requested, and it also creates and maintains the forwarding rules.

In order to provide a better understanding of this approach, an operation model in a very basic scenario is described next. The most basic scenario could be one in which just one request packet and one delivery packet are exchanged. The operations of both the upward and the downward path are described next.

On the one hand, the tasks performed in the upward path are the following ones. First, the end-node sends an interest CIU, which includes the network-identifier of the named-data. The interest CIU is encapsulated in a single carrier-packet (I). Then, the carrier-packet (I) is routed-by-name by the end-node and the border-node. By means of the routed-by-name process, both the EN and the BN obtain the address of the next BN towards the serving-node in the current CSS (i.e. the output BN). This CSS address should be consistent with the under-CONET technology of the CSS. Therefore, the carrier-packet (I) is encapsulated in the under-CONET data-unit with the CSS address as destination, and routed by means of this technology. All the CSS addresses in the path towards the content (i.e provided by the SN, BN or IN) are appended within the path-info field. The path-info is used by the content issuer to obtain the reverse-path towards the end user. The internal-nodes parse the carrier-packet (I) and forward it by using the under-CONET routing engine.

On the other hand, the tasks performed in the downward path are the following ones. The first in-path CONET node (i.e BN, IN or SN) with the Named-data CIU requested by (I), sends the content back and avoids further propagating the carrier-packet (I). The named-data CIU is encapsulated in another carrier-packet (C). On the way back to the user, the packet (C) traverses the same CSS than that packet (I), but in the reverse order. When dealing with an inter-CSS routing, both the SN and the BN get the next CSS by using the reverse path routing. Reverse path mechanisms are later discussed. When dealing with an intra-CSS routing, the under-CONET technology is in charge of routing the packet by using traditional traffic engineering. All the BN and IN in the reverse path may cache the named-data CIU for later requests of the same content.

Concerning the reverse path mechanisms, two alternatives are discussed next: the source-routing and the pending interest table. The former allows obtaining the reverse-path from the path-info field, whereas the later makes use of a Pending Interest Table at the BN to store the "pending" requests and trace the reverse path. However, the use of source-routing on reverse-path avoids maintaining "pending" states in the network nodes.

One important function related to CCN is the caching procedure. There could be different approaches and caching strategies, such as what to cache, which content to replace or expire, and so on. First of all, the Named-data CIUs are identified as the data-units to be cached by the network nodes (i.e. BN and IN). There is some control information that should be associated with the caching procedure: (1) the network-identifier; (2) the chunk number; (3) the time-related data; and (4) the security data. Starting from the first element, the network-identifier consists of a tuple of two elements: <namespace ID, name>. The namespace ID identifies the format of the name field in order to contextualize the parsing of this name. The name is a unique identifier on each namespace, which defines its own rules to get this uniqueness. In this case, the name consists of two elements: <hash (principal), hash (label)>. The principal and the label are flat names, whereas the hash function is used to get a fixed size (i.e. number of bytes) from both names. The principal identifies the owner of the named-data, whilst the label is the identifier chosen by the principal to assign a unique named-data. The second element is the chunk number, which identifies the portion of content inside the whole named-data. The third element is the temporal-data, which adds time information to the cached chunk. This information can be used as expiry date and implement a kind of digital forgetting mechanism to automatically delete old entries. The last element is the security data, which provides essential information to validate the named-data CIU before the caching procedure. This method will avoid the caching and further distribution of fake content.

As previously described, the named-data CIUs are transported in one or several carrier-packets adapted to the under-CONET technology. Then, in order to validate and cache the named-data CIUs, the carrier-packets should be previously reassembled by BN and IN. The EN also needs to reassemble the carrier-packets into named-data CIUs to get and validate the chunks, and then a further process is needed to reassemble the named-data CIUs into named-data.

Since the carrier-packets have been mentioned in the context of reassembling the named-data CIUs, the structure of a carrier-packet is described next. A carrier-packet can have the following elements: (1) the header field, (2) the payload-header, (3) the payload, and (4) the path-info field. The header field is the minimal set of control information of any CIU, which consists of the network-identifier, the chunk number and the CIU type. The payload-header specifies the byte boundaries of the segment in the context of the whole content. The payload field is only present in the named-data CIUs and it is the actual chunk of data. Finally, the path-info was previously introduced when describing the procedure to obtain the reverse-path. Each next-hop CSS address traversed in the upward path is added to this path-info field.

Another important function provided by CCN is the content-based routing. CONET includes several routing proposals. Some of them have been already introduced, such as routing-by-name and inter-CSS source routing mechanisms. One important issue when dealing with a content-based routing is the scalability of the solution, and more precisely the scalability of the routing tables. CONET introduces a name-based routing method called “lookup-and-cache”. This mechanism deals with the update of the CONET name-based routing tables, which are used by both the EN and the BN to route-by-name the Interest CIUs in the upward direction. The entries of such routing tables consist of a tuple of four elements: <network-identifier, mask, next-hop, output-interface>. These entries look like common routing entries in which the net-prefix elements have been replaced by name-prefixes (mask). In this case, the name-prefix could be a couple of <network-identifier, mask>. The next-hop element contains the CSS address of the next BN towards the SN.

One relevant concern is the dissemination of these name-prefixes. When dealing with content-identifiers (i.e. names), the prefix dissemination is an issue because the aggregation of these names is not effective. This is the reason for proposing the lookup-and-cache technique. In general, a CONET node (i.e. EN and BN) has a fixed number of rows of a named-based routing table as a route cache. By means of this procedure, when a certain entry (i.e. information about a certain content) is not available, the node will look up in a name-system (DNS like), recover the information and insert it in the route cache. When all the rows are filled in, one possible strategy is that the new rows will replace the old entries. Each name-system is devoted to a single CSS and a specific namespace. Moreover, if the SN is located inside the same

CSS, the name-system returns the CSS-address of the SN. However, if the SN is in another CSS, the CSS-address of the egress BN towards the SN is returned to the node. Nevertheless, the prefix-dissemination strategy and the lookup-and-cache proposal can be combined.

3.2 CONET implementation with OpenFlow

This section focuses on the implementation of CONET with OpenFlow as the under-CONET technology in the CSS. The CONET solution is based on several extensions to the CCNx project [CCNx]. The main enhancement that CONET contributes on is the development of a different transport layer protocol (Information Centric Transport Protocol, ICTP). The aim of this layer is to enhance the performance from the under-CONET technology. In this way, the CIUs are divided into one or several Carrier Packets to properly fit in the typical MTU avoiding further fragmentation processes. Therefore, the compatibility with CCNx still remains in the CONET implementation, since the API towards the application layer has not changed. Then, CCNs applications can be reused in the CONET testbed.

From this point on, the problem to solve is what happens when one of the CSS is an OpenFlow network. Depending on the OpenFlow version available at the CSS, there is a short term and a long term solution [ICF SDN]. This section focuses on the short term proposal, which is the one already implemented and available. The short term proposal assumes that the underlying OpenFlow network implements the specification version 1.0 and tries to adapt the CONET approach to such environment.

When dealing with OpenFlow version 1.0 rules, it is not possible to inspect the headers at arbitrary positions. Thus, there is no way to match packets and take actions based on the IP options. As previously mentioned, CONET has been defined as a new IP option (both for IPv4 and IPv6). In fact, in OpenFlow version 1.0, only a predefined set of fields in the packet header can be inspected.

Therefore, the proposed solution is to map the content carried in the IP option (i.e. the ICN-ID) into a tag which is valid for OpenFlow version 1.0. The CONET proposal adapted to OpenFlow, reuses the transport level ports (i.e. the source and destination TCP/UDP ports, with a total length of 4 bytes) to map the ICN-ID. It is important to notice that the CONET Carrier Packets use a new IP transport protocol, which is not TCP or UDP. The transport protocol is replaced to carry the ICN identifier. This means that the ICN-ID, which contains the content name with a variable or fixed arbitrary length depending on the naming schema, needs to be mapped into a 4 bytes fixed length tag.

In a general vision, the OpenFlow enabled CSS will be connected to other CSS based on different under-CONET technologies. Consequently, gateway nodes will be essential to perform the translation between the OpenFlow-based and non-OpenFlow CONET CSS. For later descriptions, it is assumed that the OpenFlow CONET network corresponds to a single CONET CSS. In this context, the gateway BN is analysed when Interest Carrier Packets are received. First, the mapping between the ICN-ID and the tag is performed. Then, the fixed 4 bytes tag is carried in the new transport level ports. In the case of the outgoing BN, it only needs to remove the tag. No further action is needed since the ICN-ID field remains the same in the IP option. One important aspect is that the mapping between the ICN-ID and the tag should be identical in all the BNs in the CSS for consistency reasons. The NRS node is responsible for performing this mapping. For scalability reasons, the tags can be reused (i.e. the tags expire) by means of a dynamic mapping, since 2^32 tags (4 bytes) do not scale to all possible content names. However, there is a concept of “active” contents, which means that there is a set of live mappings which are valid in a particular moment in the CONET Sub System. The first time that an Interest Carrier Packet enters in the BN, a request is sent to the NRS asking for the adequate mapping tag. This mapping is cached in the BN for subsequent Carrier Packets. At the same time, the BN also needs to request for the next-hop to the NRS. Therefore, the same request message can be used for both: mapping tag and next-hop.

In order to better understand the whole process, the protocol operations are going to be described [OF ICN]. Basically, the idea is to analyse the operations performed both when an Interest Carrier Packet and a Named-data Carrier Packet traverse an OpenFlow-based CSS. Before doing so, there is a previous consideration that should be taken into account. The first distinction when a packet comes into the BN, is to differentiate between non-CONET packets and CONET packets. The former are identified by using an IP protocol different from CONET and the OpenFlow network should use standard mechanisms to route the packets inside the CSS. However, the later are identified by the CONET IP protocol and two different actions are performed depending on the specific type of Carrier Packet (CP): Interest CP and Named-data CP.

On the one hand, when an Interest CP is received at the incoming BN, this node should identify the adequate outgoing BN and give the appropriate format to the CP in order to be matched by OpenFlow. First, the incoming BN queries the NRS (i.e. the controller) for the right outgoing BN. To perform this query, the header of the packet with the CONET IP option is sent to the controller. The NRS running in the controller checks whether the same Interest CP has been recently received in the CSS, which means that the mapping is active. If this is the case, the same tag and outgoing BN is assigned. If the Interest CP has not been previously issued, the NRS assigns a new 32 bit flow identifier (i.e. tag) to such Interest and identifies the most appropriate outgoing BN. When this new mapping is performed, the controller creates a new rule with this flow identifier towards the outgoing BN. Then, a new entry at the incoming BN is installed towards the outgoing BN or a local Cache Server if the content is stored in a local cache inside de CSS. The incoming BN also needs to encapsulate the Interest CP with the information received from the NRS. In this way, the packet format is adapted to the OpenFlow matching: (1) the IP protocol is set to the CONET value; (2) the transport layer ports are filled in with the flow identifier (i.e. the tag); and (3) the destination IP address is set to the IP address of the outgoing BN (or the IP of the Cache Server). Once the packet is correctly encapsulated, it is sent to the OpenFlow enabled switch located at the incoming BN, which sends the Interest CP to the appropriate network interface or the local Cache Server (if it is directly connected to the switch and already has the requested content cached). In the case of sending the Interest CP to the local Cache Server, this CP is no longer propagated through the network. All the OpenFlow enabled switches towards the outgoing BN handle the packets the same. When the Interest CP arrives at the outgoing BN, the tag is removed and the packet is forwarded towards the SN to the next CSS.

On the other hand, different actions are performed when the Named-data CP are received. The first difference with Interest CP is that the NRS is no longer used to find the path back to the user, since the users are not managed the same as content. The reverse path can be built with the information contained in the path-info field. This field stacked all the BN chosen and traversed in the upward path. Another option is to implement a “Pending Interest Table”, which provides this information by storing some state about the Interest CP upwards path. Once the outgoing BN is obtained, the incoming BN encapsulates the Named-data CP into an IP packet with the destination address set to the IP address of the outgoing BN. In the downward direction, the Internal Nodes decide or not to store a copy of the chunk in the local cache. There could be different caching strategies depending on the available resources and demanded requirements. The content cache could be updated with a Least Recently User (LRU) algorithm. It the end, the outgoing BN restores the original Named-data CP and forwards it to the end node.

There are some architectural components of an OpenFlow-based CSS that must be highlighted: (1) the forward-by-name; (2) the data forwarding; (3) the content routing; and (4) caching. The forward-by-name is used to forward the Interest CP to the adequate output interface relying on a name-based forwarding table. The data forwarding is used to send the content back to the user (i.e. deliver the data). The main difference from the forward-by-name is that the users are not addressed by the content routing plane; thus, the same strategy is not possible for upwards and downwards. The content routing is responsible for disseminating the information about the actual location of the contents. Therefore, this process is essential for the proper setup of name-based forwarding tables. Finally, the caching is the ability of ICN nodes to cache data and reply to incoming content request.

Although, as previously mentioned, the long term solution is not part of this study, some ideas about this approach are going to be introduced. The long term solution for CONET adaptation to OpenFlow-based CSS relies on some OpenFlow extensions to be added. The reason for being out of the scope of this section is that there is no actual implementation available. Basically, the idea behind it is to make use of the flexible match support up to the IP options (not available with OpenFlow version 1.0) and to define new ICN specific methods and operations. Therefore, the OpenFlow API should be extended to handle the concept of "content" and support content related methods and operations, such as key management, caching of content, routing-by-name, and so on.

3.3 CONET implementation deployed in the OFELIA testbed

This section focuses on the specific adaptations that have been done to the current CONET implementation in order to deploy this CCN application over an OFELIA slice [ICN SDN OF]. So, this also covers the deployment of CONET in the OFELIA testbed. The OFELIA project has been introduced in [D2.1 Chapter 6].

The overall view is a scenario in which the OpenFlow based network is part of a larger ICN network. The OpenFlow based CSS is, in fact, the OFELIA slice devoted to the CCN application. Inside the slice, the OpenFlow features are exploited for content delivery and thus, the under-CONET technology is OpenFlow in this case. The whole slice can be shown as a complete OpenFlow domain. The compatibility between the OpenFlow and non-OpenFlow domains is supported by the so-called InterWorking Elements (IWE). These elements translate the regular ICN operations to OpenFlow-based ICN operations and vice versa. This function is performed by the BNs of the OpenFlow based CSS.

The content is stored and returned by the ICN Cache Servers, which are CONET nodes with caching support (i.e. BN and IN). Due to the lack of caching support in the OpenFlow switches, the Cache Server and the caching functionality are implemented in an external node. In the future (long term solution), this functionality will be integrated in the OpenFlow switches.

The interface between the NRS node (i.e. an application running in the controller) and the OpenFlow Switch is the OpenFlow protocol version 1.0. However, OpenFlow is not the interface between the NRS and the Cache Server.

Some simplifications have been assumed and some of them are imposed by the OFELIA testbed. The switching equipment is OpenFlow 1.0 and it is basically off-the-shelf equipment. Moreover, the OpenFlow domain is a single IP subnet which operates at layer-3. It is also possible to have a set of IP subnets interconnected by OpenFlow routers.

Regarding the Interest packets, when one of those packets are received by the BN, there are two alternatives depending on whether the content has been cached or not. If the content has been already cached within an "in-network" Cache Server, the Interest packet is directly forwarded to the Cache Server. If the content has not been cached yet, there is also a distinction if the content is at a local or external ICN server. If the content is at a local server, which is internal to the OpenFlow domain, the Interest packet is forwarded to the SN. If the content is not at a local server, the Interest packet is forwarded to the outgoing Edge Node (i.e. outgoing BN) on the path to the target SN.

Regarding the Data packets that are transmitted from the content server (i.e. SN) towards the end host, all the traversed OpenFlow switches with an associated Cache Server (local or remote) duplicate the packets and a copy of these packets are cached at the caching server.

The operations involved in these processes are described next in more detail (Figure 3-2). When an Interest packet enters in the BN (i.e. incoming Edge Node), the incoming BN resolves the ICN name to an IP address as the ICN next-hop. This next-hop can be a content server (i.e. SN) or the outgoing Edge Node (BN) towards the SN, which is obtained by using the ICN routing information. Then, the Interest packet is forwarded through the OpenFlow domain by means of the underlying technology, in this case, the specific logic located at the controller. Finally, the Content Server or the outgoing BN sends back the data packets to the incoming BN.

Due to the fact that with OpenFlow version 1.0 it is not possible to read and process the content names located at the CONET IP option (i.e. ICN-ID), the forwarding function is based on tagging by means of tag-based routing. The ICN BN performs the mapping of content from the name to the tag. The NRS node helps in this process and is logically centralized to ensure the unequivocal mapping across the whole domain. Then, the outgoing ICN BN removes the tag and caches the association between the content name and the tag. When the content data packet returns to the same BN, the tag is directly added since this mapping has been previously cached. Once the packet is tagged, it is forwarded based on this tag through the OpenFlow domain.

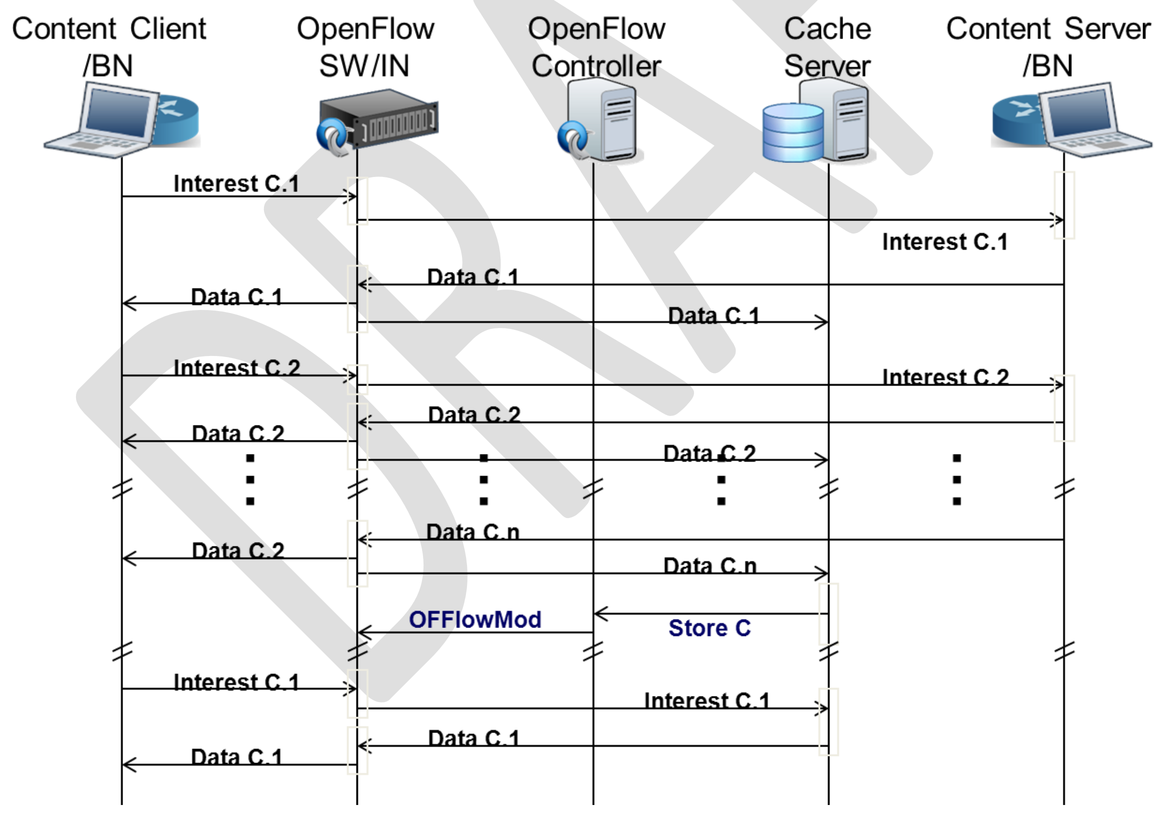


Figure 3-2 Sequence of operations [ICN SDN OF]

When the data packet returns in the downward direction, it is duplicated towards the Cache Server. Once the full chunk of data is completely cached by the node, the Cache Server notifies the controller. Then, the controller adds the flow information into the switch. In this way, further requests of the same chunk of data are directly forwarded to the Cache Server.

The CONET tag is located in the first 4 bytes of the IP payload, which is where the source and destination TCP/UDP ports are located. TCP and UDP protocols are the only IP protocols defined by the OpenFlow specification version 1.0. Therefore, one of them has been chosen to transport the CONET tag, the protocol type 17 (i.e. UDP). However, it is important to distinguish the actual UDP from the CONET packets (which uses UDP as IP protocol for implementation reasons). In order to clearly differentiate both uses of the same IP protocol value, a set of reserved IP addresses have been selected. Consequently, all the ICN nodes use specific IP addresses for CONET. In the end, the unambiguously identification of the actual content comes from the combination of a reserved IP address and the source and destination UDP ports (where the tag is located).

At the OFELIA slice (Figure 3-3) the OpenFlow switches identify the CONET IP addresses (due to the limitations imposed by OpenFlow version 1.0). Thus, the CONET Interest packets are identified by 192.168.1.8, whereas the CONET data packets are identified by 192.168.1.23. In a clean slate solution, a specific IP protocol is defined for the CONET packets.

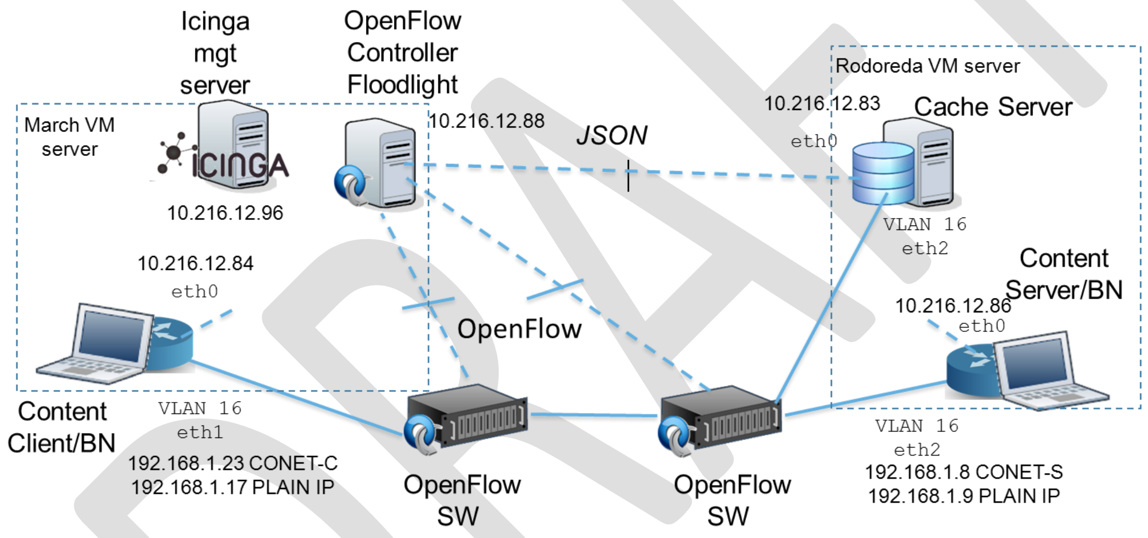


Figure 3-3 CONET testbed in OFELIA [ICN SDN OF]

Regarding the Cache Server and the caching strategies, there are two options: one-to-one and one-to-many associations. In the one-to-one association, there is a direct link between the Cache Server and the OpenFlow enabled switch. In the one-to-many, the Cache Server is remotely located and many OpenFlow enabled switches are associated to the same Cache Server in order to cache the content.

Currently, only a first simple configuration has been implemented over the OFELIA slice, which consists of just one client, one ICN Server, one Cache Server, one OpenFlow controller (i.e. FloodLight) and two OpenFlow enabled switches. Moreover, the client and the server internally implement the IWE functionality. Therefore, the CONET packets are sent with the tag (mapping the actual content name).

Initially, some tests have been performed by means of a request generator located on the end node, which is continuously requesting the same six content elements. In the tests, there are two steps. In the first step, the ICN caching is disabled and initially the SN provides the content. In the second step, the caching functionality is enabled and, in this case, the Cache Server provides the content instead of the SN.

3.4 ALIEN extensions to CONET

The current development of CONET has been deployed over an existing testbed in OFELIA to provide verification of the functionality and defined interfaces. In order to simplify the scenario, only one client, one content server and one Cache Server have been considered. This imposes some restrictions and a more realistic scenario with several entities of each type should be considered, including multiple clients, multiple content servers and multiple Cache Servers. Since the whole setup uses OpenFlow as under-CONET technology, only one CSS is necessary. This means that just one NRS entity is required for the whole CSS. Thus, only in case of defining multiple CSS, multiple NRSs would be needed.

The ALIEN project will actively contribute to the CONET development in cooperation with the team that is currently supporting the code. Due to the fact that the ALIEN project deals with OpenFlow, the consortium focuses mainly on the OpenFlow related part of the CONET development. This is the development of the Controller (i.e. the FloodLight modules), where the NRS function is located. Currently, some areas have been already identified and agreed between both teams as possible areas for enhancement in which ALIEN can positively contribute. By adding those functionalities the CONET development will be improved in key aspects which were already in their planning.

Basically, the ALIEN efforts will be concentrated in two main areas: the support of multiple nodes of the same type (i.e. multiple Cache Servers) and the adding of topology awareness to the Controller.

On the one hand, a realistic scenario will take advance of multiple Cache Servers geographically distributed (Figure 3-4 (a)) along the CSS to reduce both the latency of content delivery and traffic load in the core links. This will improve the performance of the content distribution and the scalability of the system through several local Cache Servers distributed close to the clients. In order to test the improvements, multiple clients and content servers will also be distributed through the whole setup. This will affect the controller, since supporting multiple nodes of the same type impacts on the data modelling of the whole implementation.

On the other hand, topology awareness (Figure 3-4 (b)) will be added to the Controller. This feature is important for several reasons. For instance, it could be used to enforce loop avoidance enabling redundancy support in the CCN deployment. Currently, it is not possible since only loop free topologies (i.e. tree topology) are supported. Moreover, topology awareness also enables the future support of traffic engineering. In this context, there could be some routes reserved to CCN traffic or it would be possible to route CCN and non-CCN traffic following different paths. These alternatives are studied for the definition of use cases.

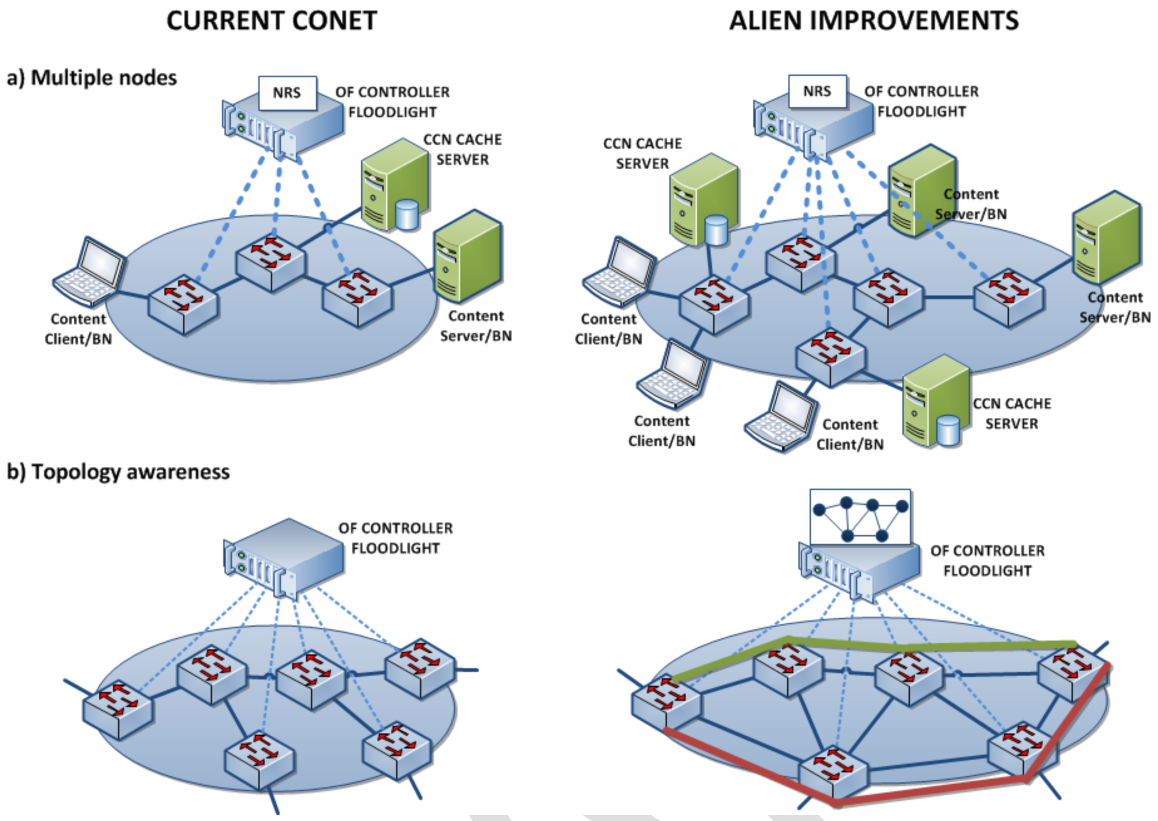


Figure 3-4 ALIEN improvements to the current CONET development

Both the support of multiple Cache Servers and the topology awareness have impact on the proactive phase, where the caching is configured, and the reactive phase, where routing decisions are made based on the topology.

Additionally, ALIEN will also try to overcome the IP addresses decoupling. Currently, one IP range is used for CONET and another IP range is used for legacy traffic. This issue is not trivial due to the limitations imposed by the OpenFlow version 1.0, but several alternatives will be considered to solve it at another layer.

3.5 OpenFlow functionality required by CONET

Table 3-1 shows the basic OpenFlow functionalities for version 1.0 and 1.3, describes them and shows whether or not those functions are required by the CONET system. This table, therefore, is an important input into what OpenFlow functions need to be supported by the ALIEN hardware in order for a test-bed to implement the experiment required by the use cases.

OF basic functionalities	OpenFlow versions	Description	Use within CONET
One Flow Table	OF1.0, OF1.3	A flow table is the main element of OpenFlow switch and contain flow entries	All data plane CONET nodes support switching

More than one Flow Tables	OF1.3	Packet can be processed multiple times with different sets of flow entries	
Group Table	OF1.3	Perform specific operations like load-balancing (send to random port of group), failover (send to first live port of group), multicast (send to all port of group), etc	
Meter Table	OF1.3	Perform QoS operations (traffic shaping) over flows	
Cross-connect Table	OF1.0 circuit ext v0.3	Contains a set of circuit flow entries, which detail the cross-connections between input and output channels	
Forward to physical port (Output)	OF1.0, OF1.3	Processed frame is directly sent out by given data plane interface	Interest packets are sent from content client to content server direction; data packets are sent from content server to content client direction; without QoS functionality
Forward to all physical ports (Output ALL)	OF1.0, OF1.3	Processed frame is directly sent out by all interfaces (excluding input port of frame)	Required for CONET to work in "learning switch" mode
Forward along spanning tree ports (Output FLOOD)	OF1.0, OF1.3	Processed frame is directly sent out along the minimum spaning tree interfaces (excluding input port of frame)	The CONET controller is sending out frame to all ports (FLOOD) if it has not learned the port for the dest MAC/VLAN for frame that was received in Packet-in notification
Forward to controller (Output CONTROLLER)	OF1.0, OF1.3	Processed frame is sent to the controller	
Forward to a table (Output TABLE) (Goto-Table)	OF1.3	Start processing a frame in a specified flow table	
Forward to input port (Output	OF1.0, OF1.3	Processed frame is sent out by the input interface	

IN_PORT)			
Multiple forwards	OF1.0, OF1.3	A single action list can contain multiple forward actions	Used to duplicate packets, one to the actual destination and another to send content into cache Server.
Enqueue (Set-Queue)	OF1.0, OF1.3	Forward frame through a queue attached to a port	Interest packets are sent from content client to content server direction; data packets are sent from content server to content client direction; possible QoS functionality.
Drop	OF1.0, OF1.3	Matched frame should be dropped	Used for all unsupported packets
Modify-Field actions (Set-Field)	OF1.0, OF1.3	Modify-Field actions allows to add, modify, remove most important fields, structures in processed frame headers	Rewriting MAC addresses used to redirect the duplicated packet to the Cache Server
Group	OF1.3	Process the frame through the specified group in group table	
Matching against ingress port	OF1.0, OF1.3		
Matching against Ethernet MAC addresses and Ethernet type	OF1.0, OF1.3		Clients are recognized by combination of MAC and VLAN identifiers; flow entries for CCN traffic use dst MAC addresses
Matching against VLAN id and priority	OF1.0, OF1.3		VLANs are used for isolating experiments in OFELIA (VLANs are not transparent for CONET). This is why clients are recognized by combination of MAC and VLAN identifiers; flow entries for CCN traffic use VLAN tags
Matching against IPv4 addresses, protocol and ToS	OF1.0, OF1.3		The CCN application uses specific IPv4 addresses to distinguish the CCN traffic from regular traffic
Matching against IPv6 addresses,	OF1.3		

protocol and ToS			
Matching against TCP/UDP ports	OF1.0, OF1.3		The content-names are mapped into UDP ports, and specific flows are installed in the switches to redirect all Interest Carrier Packets of already cached chunks to the most appropriate or local Cache Server.
Modify-State	OF1.0, OF1.3	Controller is able to add/modify/delete flow entries	To instruct switch about new cached content or about content location changes and content deletion from the server
Read-State	OF1.0, OF1.3	Controller is able to collect statistics (counters values) from flow tables	Extended functionality for network traffic measurements
Send-Packet (Packet-out)	OF1.0, OF1.3	Controller is able to send packet out of specified port	The CONET controller is sending out frame to all ports (FLOOD) if it has not learned the port for the dst MAC/VLAN for frame that was received in Packet-in notification
Packet-in	OF1.0, OF1.3	Notify the controller about a frame not matched in any existing flow entries (by default first 128 bytes of frame are sent to the controller)	All unknown (not matched) CCN packets (interest or content) are sent to the OF controller
Flow-removed	OF1.0, OF1.3	Notify the controller about expiration of the flow entry in the switch	Flow entries will expire if client with matching src MAC disconnects a device from a port
Port-Status	OF1.0, OF1.3	Notify the controller about port configuration status change	For CCN network monitoring
Error	OF1.0, OF1.3	Notify the controller about problems within the switch	For CCN network monitoring
Per table counters	OF1.0, OF1.3	Collect table related information as active entries, packet lookups and matches performed	
Per flow counters	OF1.0, OF1.3	Collect flow related information as received	

		packets, received bytes, durations, etc	
Per port counters	OF1.0, OF1.3	Collect port related information as received/transmitted packets, received/transmitted bytes, drops, errors, collisions, etc	
Per queue counters	OF1.0, OF1.3	Collect queue related information as transmitted packets, bytes, errors	
Per group counters	OF1.3	Collect group related information as flow entries count, packet count, byte count, durarion	
Per meter counters	OF1.3	Collect meter related information as flow count, packet count, byte count, duration	

Table 3-1 OpenFlow capabilities

4 Testbed description

This section describes how the ALIEN testbed will be deployed in cooperation with the OFELIA facility. All partners are providing testbeds for the project. The project use case experiments will take place over this ALIEN hardware and will allow these testbeds to interact, sometimes via dedicated connections between testbeds and at other times via connections over the general internet.

4.1 ALIEN slice at OFELIA

The OFELIA experimental facility is based on OpenFlow, a networking technology that allows virtualization and control of the network environment through secure and standardized interfaces. It is based on a heterogeneous set of ten islands that allow experimentation on multi-layer and multi-technology networks provided by the different islands (TUB – Berlin, iMinds – Ghent, ETH – Zurich, i2CAT – Barcelona, UNIVBRISTOL – Bristol, Create-Net – Trento, CNIT – Rome, Pisa & Catania, UFU – Uberlândia) (additional information in [ALIEN D2.1] Chapter 7).

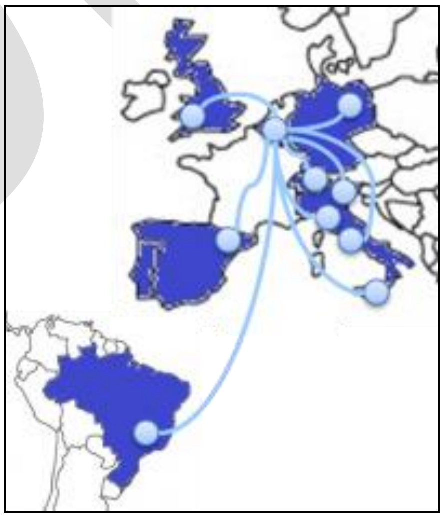


Figure 4-1 OFELIA Facility

To test the CCN Scenario portrayed in this deliverable, a network slice will be assigned to the ALIEN project, consisting of the following resources from OFELIA:

- Virtual machines as end-hosts deployed on physical servers (at least 6 VMs, distributed all over the facility)
- A virtual machine to deploy the CCN Controller (FloodLight).
- A set of OpenFlow switches distributed over multiple OFELIA islands. Currently the interconnection is both provided via dedicated circuits (1Gpbs) and via best-effort connection through Internet (L2 tunnels), all crossing through the central hub in Ghent, in a star topology.

An example topology of the ALIEN slice is represented in Figure 4-2. The connections between the node elements refer to the data plane, e.g. the user traffic (CCN and not) between the end hosts. Each OFELIA Island is equipped with multiple servers, which host user's virtual machines, connected with the OpenFlow switches.

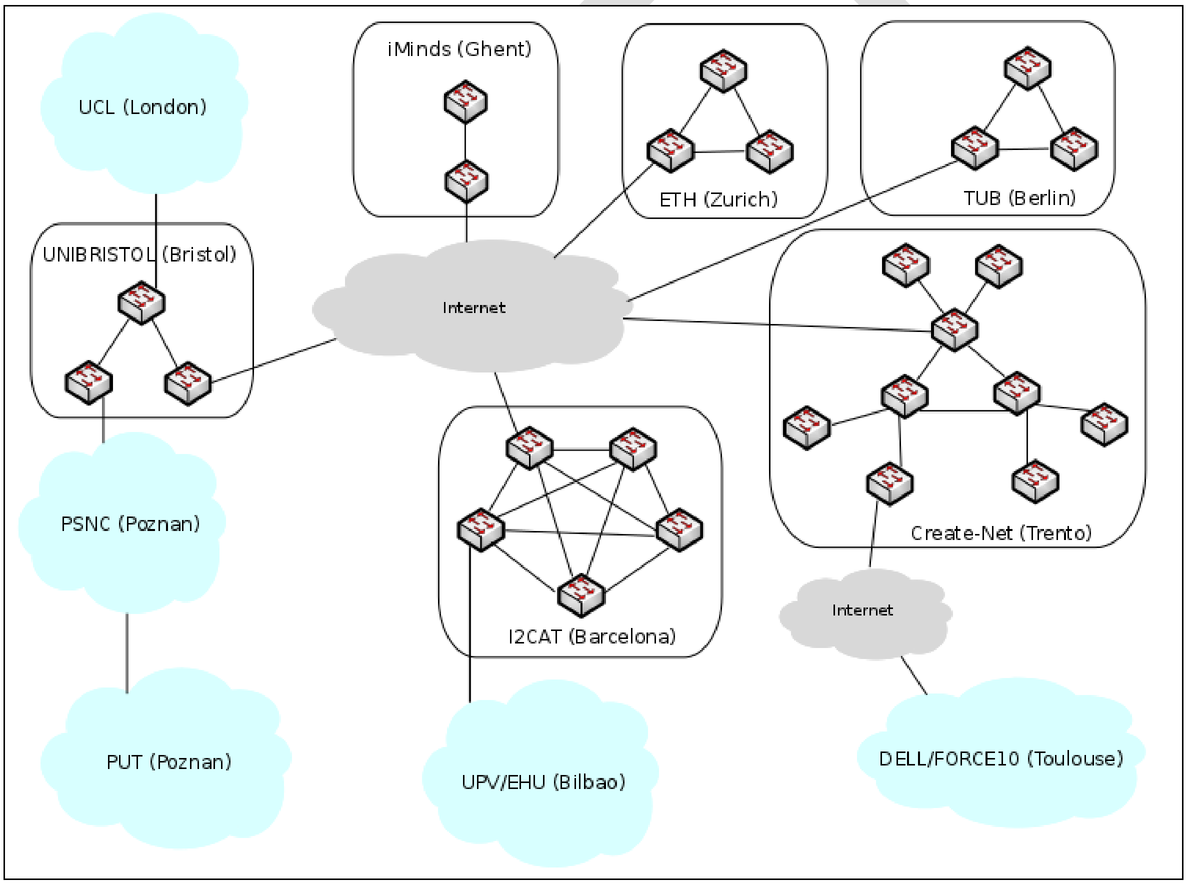


Figure 4-2 ALIEN slice topology

The ensure isolation between each slice (e.g. one slice cannot control another's traffic) OFELIA adopted FlowVisor tool. FlowVisor is an open-source special purpose OpenFlow controller developed by ON.Lab that acts as a transparent proxy between OpenFlow switches and *multiple* OpenFlow controllers.
Slices can generally be defined by any combination of switch ports (layer 1), src/dst ethernet address or type (L2), src/dst IP address or type (L3), and src/dst TCP/UDP port or ICMP code/type (L4).

Moreover, to establish a unique slicing mechanism for all the facility, the OFELIA consortium chooses the L2 VLAN field. For this reason, all the traffic belonging to a range of VLAN (at least 10) will be forwarded to the CCN controller, independently from the other L2-L4 fields.

Figure 4-3 represents the CCN overlay that will be deployed in the ALIEN slice. The different elements (cache servers, CCN servers, clients and OpenFlow controller) will be placed both by the OFELIA facility and by the partners' testbeds.

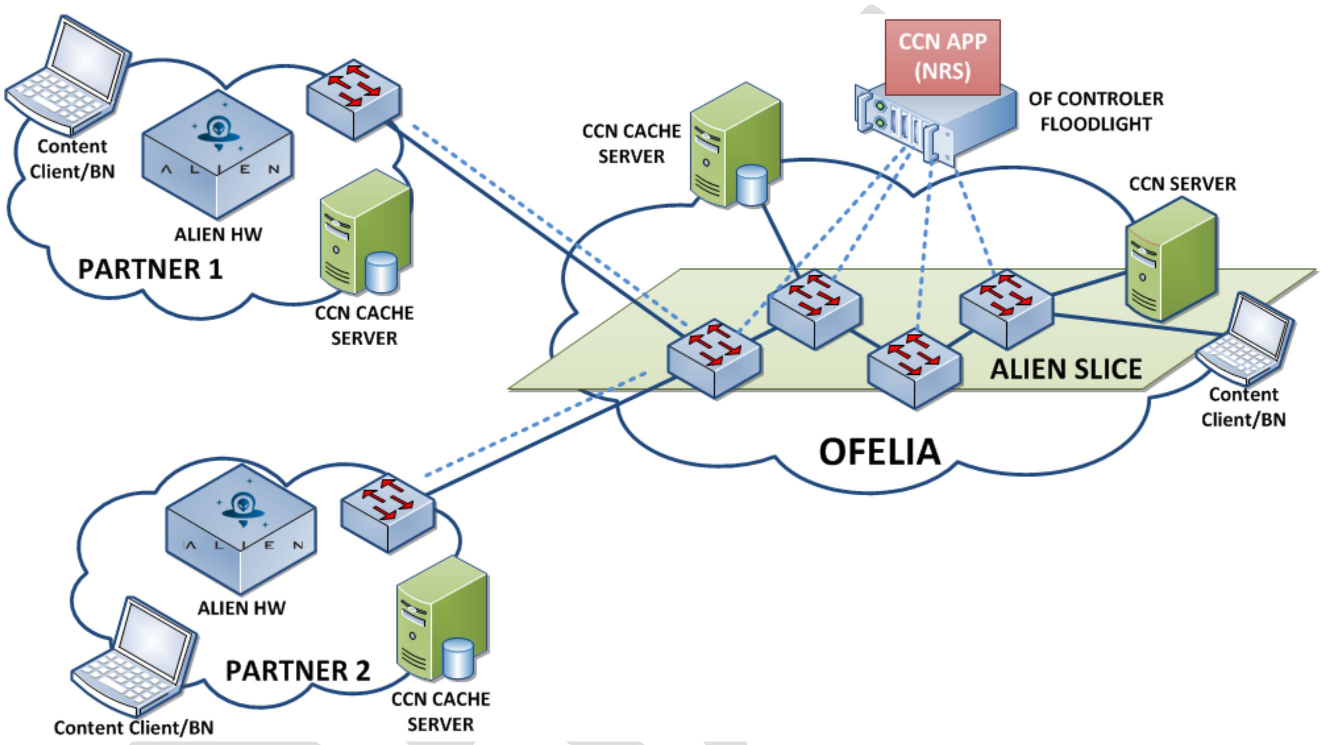


Figure 4-3 CCN Overlay

4.2 Testbed resources and connection to OFELIA

In this chapter each partner lists all the ALIEN resources in its testbed, and how these resources can be interconnected with the OFELIA facility. Three partners are already part of the OFELIA facility (EICT, University of Bristol and Create-Net). Fuller descriptions of the testbeds are already given in [ALIEN D3.1].

The following table summarizes how all the partners have planned their connection to OFELIA:

Partner	OFELIA island with OCF	Connection to OFELIA – Control / Data plane		
		NREN	Type of connection	Home Island
PSNC	Yes	GÉANT	Layer2 (VLAN tag)	UNIBRISTOL
UPV/EHU	Yes	RedIRIS	Layer2 (VLAN tag)	i2CAT

Create-Net	Already	-	OFELIA island	-
PUT	Yes	PIONIER	Layer1 (optical link) to PSNC	PSNC / UNIBRISTOL
UNIBRISTOL	Already	-	OFELIA island	-
UCL	No	JANET	Layer2 (VLAN tag)	UNIBRISTOL
DELL	Yes	RENATER / GÉANT	(3x) VPN Layer2 tap	Create-Net

Table 4-1 Connection to OFELIA

4.2.1 PSNC testbed

In order to connect PSNC Network Processor laboratory to OFELIA, the existing connections over GÉANT to Bristol University will be used. To realize the connectivity, there will be used GÉANT Plus Service instances that are now established between PSNC and Bristol University premises (VLAN tags: 600-620). Some of these VLANs may be used for ALIEN project, depending on experiment requirements. The available bandwidth between PSNC and Bristol University is 1 Gbps (note that all bandwidth is shared between all existing VLANs).
PSNC Network Processor laboratory is managed via PL-LAB (see the details in [ALIEN D3.1]). PL-LAB Web Portal offers functionality to create private virtual networks (OSI Layer 2 services) through interconnecting devices and ports available in PL-LAB network.
The process of creating the experiment within PL-LAB starts with configuring VLANs as well as assigning Gigabit Ethernet ports to particular VLANs in the local switch Juniper EX4800. Each port taking part in an experiment can be set in untagged, single tagged and trunk modes. Remote management access to any device is provided by predefined separate management VPN. It allows to manage the device even if the experimental virtual network is not available. Each device has exactly one port (mostly it is ‘eth0’) statically assigned to management VPN (IP address of a such port is predefined). Other ports of devices are free to be configured and used in any way for experiments purposes. Details of PSNC testbed are presented in Figure 4-4.

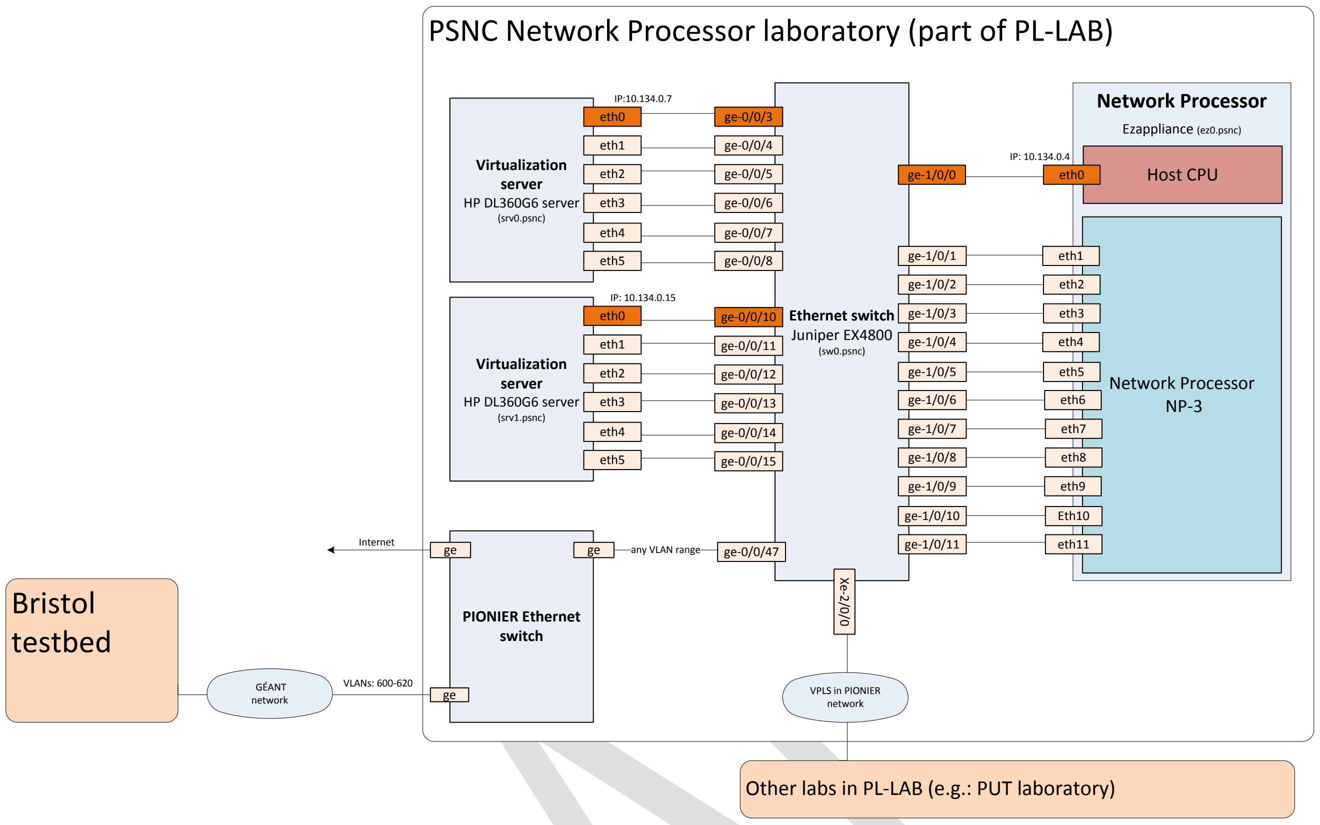


Figure 4-4 Interconnections and internal details of PSNC Network Processor laboratory

PL-LAB virtual laboratory can be composed of resources that are located in PSNC as well as in other PL-LAB partners in POLAND. One of the ALIEN partners – Poznań University of Technology (PUT) is also participating in PL-LAB federation and can interconnect the local testbed thru PSNC to OFELIA islands (via Bristol). The total available bandwidth between PSNC and PUT is 10 Gbps (it is shared between all experiments in which PSNC and PUT are participating).
The ALIEN project participants will be able to create experiments composed of programmable packet processors devices (at least one EZappliance device available in PSNC) and virtualization servers (at least one server). PUT resources may be part of these experiments if it will be necessary. Within PL-LAB, a public IP address can be granted and assigned for any network resource taking part in an experiment. This public IP address may be used for accessing an experiment in PL-LAB from Internet (e.g.: establishing a connection between OpenFlow controller deployed in OFELIA testbed and OpenFlow agent for EZappliance in PSNC).

4.2.2 UPV/EHU testbed

Currently, the EHU OpenFlow Enabled Facility (EHU-OEF) is the core of the UPV/EHU testbed related to the OpenFlow activities (described in [ALIEN-D3.1] Section 2.4). The EHU-OEF has the ability to create new slices for the experiments. Therefore, the first idea was to connect a subset of the resources from this facility to OFELIA. However, the EHU-OEF and OFELIA are incompatible both at control framework level and, more important, at data plane. Both facilities have a different approach to the virtualization of the data plane and the definition of the network slices, and cannot be directly connected. OFELIA relies on VLANs to virtualize the network, whereas the EHU-OEF makes use of Ethernet MAC prefixes to define the network slices. This is a problem for inter-island connectivity at data plane, since there is not a common end-to-end definition of the

slice. A kind of gateway should be needed to adapt both data planes and ensure the isolation of experiments when traversing the island's border. Having different control frameworks is an issue, but dealing with two control frameworks is a barrier that can be solved.
Because of these problems, and in order to become part of OFELIA as a new island, a new set of resources has been devoted to OFELIA and will be controlled by the OFELIA Control Framework (OCF). Initially, the island will consist of two NEC switches IP8800/S3640 OpenFlow enabled and computational resources for deploying virtual machines. This new EHU-OFELIA island will be fully compatible with OFELIA. The interconnection of resources located at the EHU-OEF to OFELIA, such as the DOCSIS platform, will be solved internally at the UPV/EHU testbed.
One of the remaining actions to complete is the connection of the EHU-OFELIA island to the OFELIA facility. The idea is to connect our research laboratory to the i2CAT island at OFELIA through the Spanish NREN, RedIRIS. One of the points of presence (PoP) of RedIRIS, is located at the premises of the University of the Basque Country. A 10 Gbps link will be provided by the new optical infrastructure of RedIRIS-NOVA to be shared by all the research projects that needs a special and dedicated connectivity at UPV/EHU. A 1 Gbps interface will be devoted to the connection of the EHU-OFELIA island to OFELIA. Internally at RedIRIS this service will be tagged with a specific VLAN. This service will provide 1 Gbps layer-2 connectivity between our EHU-OFELIA island and the i2CAT island at OFELIA.

4.2.3 PUT testbed

PUT is a testbed which needs to create a connection to the OFELIA islands. In order to realize this connectivity existing connections will be used. Currently, PUT laboratory has access to the PIONIER (Polish Optical Internet) network. Due to that fact, connectivity between PUT and PSNC is established and available bandwidth is 10 Gbps. This bandwidth is shared between all experiments in which PUT and PSNC are involved. Connectivity with PSNC provides opportunity to connect to the OFELIA islands over GÉANT network.
PUT laboratory can be managed via PL-LAB system. Using Web Portal of PL-LAB it is possible to establish a virtual private network (VPN) using devices and ports available in PL-LAB network. This functionality provides opportunity to establish an experiment (or testbed), which, in turn, can be used in ALIEN project. Administrator has access to all declared and connected devices in PL-LAB. Procedure of creating a new experiment within PL-LAB begins with configuring of VLANs. Ports that will take part in experiment should be added to appropriate VLAN. After this, devices can be added to the created experiment. Every device has a dedicated port used to the VPN management (mostly it is 'eth0' port). The rest of the ports of the device are free and can be configured according to the needs of the experiment requirements. Details of PUT testbed can be seen in Figure 4-5. More information can be found in [ALIEN-D3.1].

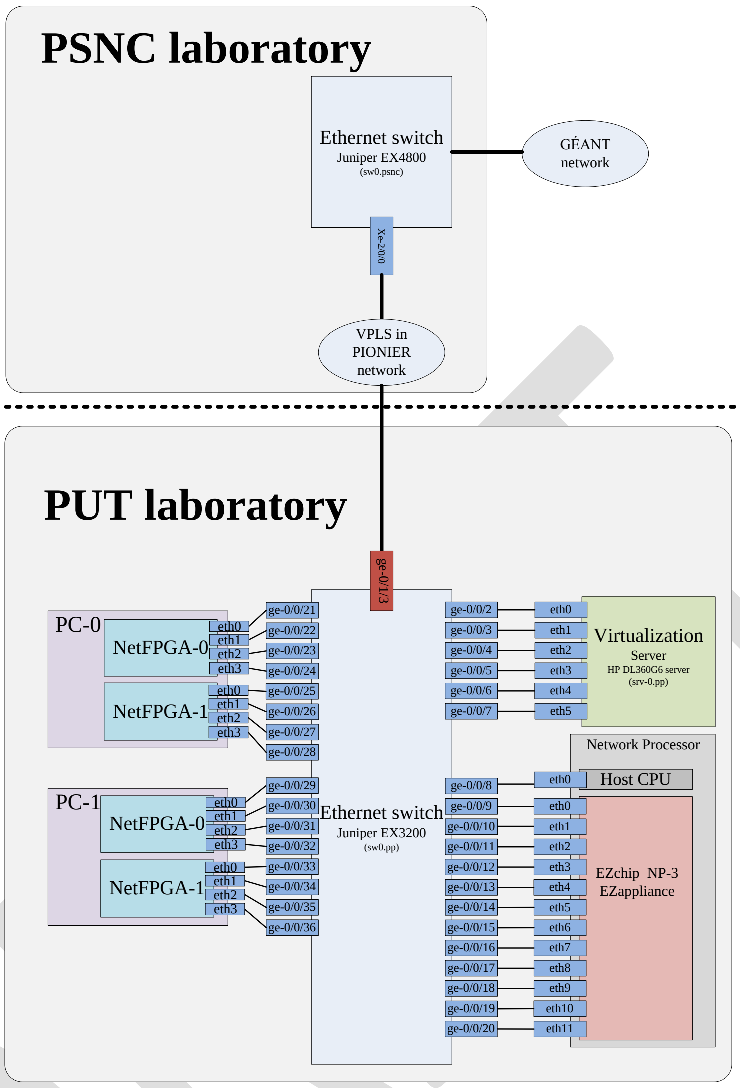


Figure 4-5 Interconnections and internal details of PUT laboratory

PL-LAB is a very useful virtual laboratory, which collects resources from all PL-LAB partners in Poland. One of PL-LAB partners is PSNC. This virtual laboratory is a set of tools whereby the new experiments between partners can be established in a very easy way.
PUT laboratory will provide access to the following devices:

- EZappliance – network processor (NP-3 processor),
- HP DL360G6 – virtualization server,
- NetFPGA cards (at least 2 computers equipped with 2 cards will be accessible; if necessary PUT laboratory will be extended by additional NetFPGA cards – depends mostly on available ports in switch(es)).

PUT laboratory is equipped with 12 x 1 Gbps NetFPGA cards and in the nearest future they will be extended by few 10 Gbps NetFPGA cards. By request devices from PUT laboratory can constitute a part of PSNC experiments.

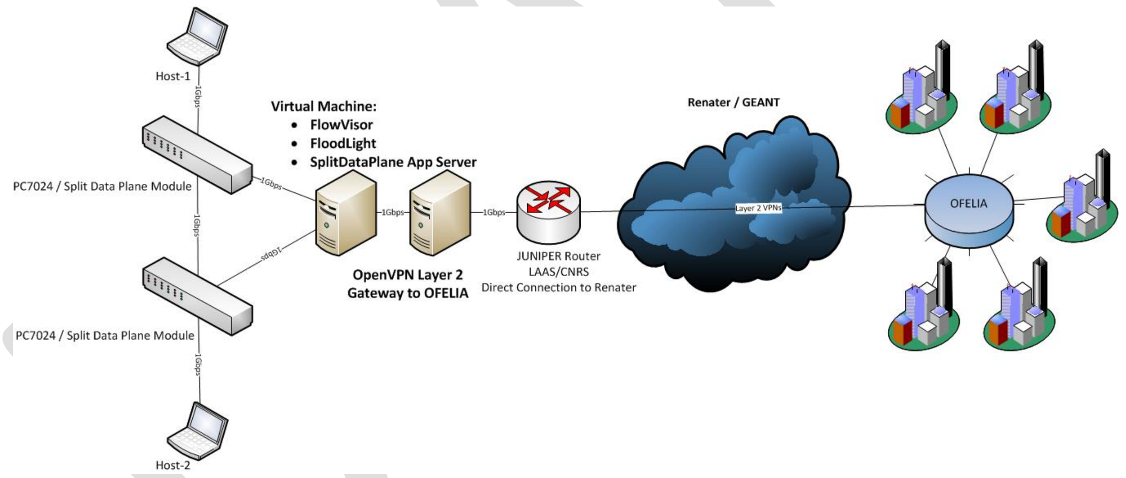
4.2.4 Dell Force10 testbed

DELL France/FORCE ALIEN testbed as described in [D3.1 Chapter 2] will be connected to OFELIA. DELL France has a partnership with a French Academic Lab, the “Laboratoire d’Analyses et d’Architecture des Systemes LAAS/CNRS” based in Toulouse.
The intention is to interconnect the FORCE testbed equipement directly in the LAAS/CNRS network, who is connected to the French National Research IP Network RENATER (http://renater.fr).

The LAAS/CNRS has a network dedicated for experiments and totally separated from the Local Area Network of the LAAS Campus. This network is also directly connected to RENATER and via RENATER to GEANT.

The testbed will use VPN Layer2 TAP tunneling functionality to be able to interconnect to OFELIA. OFELIA will provide a dedicated VLAN number and we will use a server to terminate the Layer2 Tunnels and send the traffic to the specific VLAN allocated for the data plane and the control plane.
The ALIEN members will be able to access the testbed composed of two PowerConnect 7024 with each a Split Data Plane modules and dedicated Virtual Machines running FloodLightand FlowVisor. The ALIEN/OFELIA members who like to develop specific application on the Split Data Plane, will be able to use a Split Data Plane install script to deploy easily new code on the Split Data Plane Module.

Diagram of the FORCE testbed OFELIA inter connection:



Host-1
1Gbps
PC7024 / Split Data Plane Module
1Gbps
Virtual Machine:
FlowVisor
FloodLight
SplitDataPlane App Server
1Gbps
1Gbps
1Gbps
OpenVPN Layer 2 Gateway to OFELIA
JUNIPER Router LAAS/CNRS Direct Connection to Renater
Renater / GEANT
Layer 2 VPNs
OFELIA
PC7024 / Split Data Plane Module
1Gbps
Host-2

Figure 4-6: Force 10 testbed

Two physical servers will be deployed:

- A dedicated server with 2x1Gbps running as OpenVPN gateway to transport Layer2 frames with VLAN Tagged with 802.1q for the slicing made by FlowVisor. Three VPN will be connected to the OFELIA network:
    - A VPN for the data.
    - Two VPN for the OFELIA control, one active and one passive using OSPF as protocol for detecting active link failures and to bring up the back VPN.
- A Server with multiple VMs connected with 2x1Gbps :
    - A VM with FloodLight as primary OpenFlow Controller.
    - A Split Data Plane management VM.
    - A FlowVisor VM.

- A VM for future applications

4.2.5 UCL testbed

Currently the UCL testbed is not connected directly to any OFELIA island. The testbed is part of a larger testing laboratory for passive optical networking equipment. The connection to OFELIA will be made by activating a dedicated optical “Lightpath” connection between University of Bristol and University College London. This will give a direct 1GB connection between the Bristol OFELIA island and the UCL testbed. The equipment at UCL will expose the ALIEN hardware (GEPON) via its OpenFlow controller and also several client machines which can accept logins over ssh. In this way the UCL testbed will not appear as an OFELIA controlled island itself but instead as extra equipment available at the ALIEN slice of the Bristol island. This will allow the UCL equipment to appear as part of the OFELIA Control Framework.

5 OpenFlow CCN use cases

This section describes the use cases which will be used to test whether the implementation of OpenFlow has been successful on the ALIEN hardware and the utility of OpenFlow in networking scenarios. The application chosen is CCN. Various potential scenarios will be tried. Section 5.1 describes the base case scenario. Section 5.2 describes the split routing scenario. Section 5.3 describes the cache assisted scenario. Section 5.4 describes the combined scenario.

Figure 5-1 shows a schematic view of the CCN/CONET experiments to be carried out in all scenarios. There must also be a Name Routing System (NRS) and Floodlight OpenFlow controller as described in section 3.4. These are not pictured for simplicity. The CCN nodes will typically (though not always) be end hosts in the system. In the deployed testbed they will be running on virtual machines (VMs) controlled through the OFELIA control framework. This should provide a consistent deployment of identical VMs on which to install the software. The non CCN nodes will similarly be end hosts running normally on virtual machines. The only role for these machines is to provide competing traffic for the CCN nodes for the experiments described in sections 5.2 and 5.4. CONET supports a single extra cache machine which provides extra storage for CCN content and the ALIEN project will extend the capabilities of CONET as described in section 3.4 in order that extra cache nodes can be used. These cache nodes will be used in the experiments in section 5.3 and 5.4

In the currently envisaged set up, all test beds which are OFELIA islands will connect to each other (the current topology is a star centered on a single island but this may change as the experiment progresses). The UCL testbed will connect, not as an OFELIA island but as external equipment hosted via an OFELIA island. These situations are all shown in the figure as a connection via the general internet.

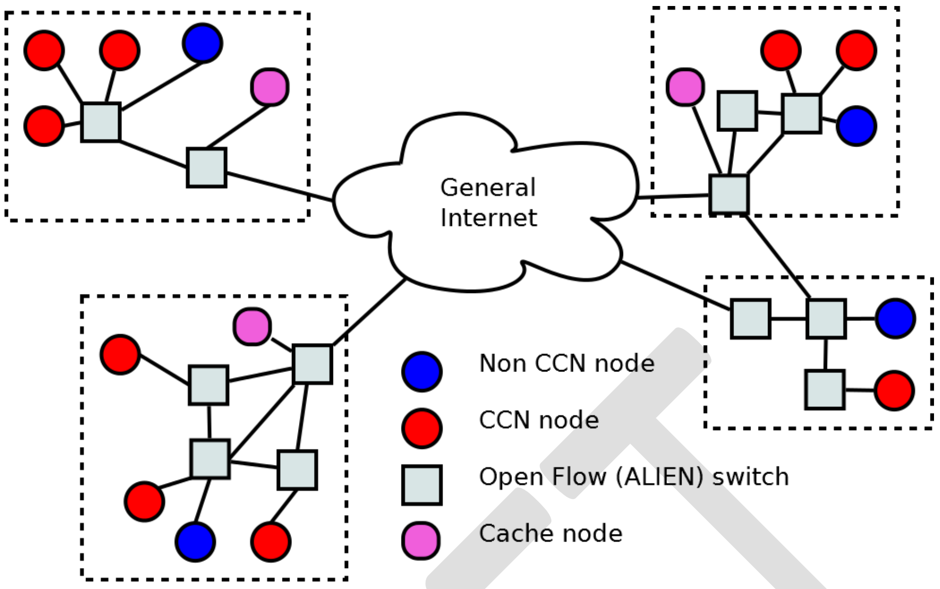


Figure 5-1 Schematic diagram for the CCN use case

It is important to note that CCN operates as an application layer overlay. While the topology in Figure 5-1 makes it appear as though CCN traffic will never travel via intermediate CCN nodes (because they are implemented on VMs which are “leaf” nodes), Figure 5-2 makes it clear that, in fact, this can happen. The CCN nodes act (using CCNx) as an application layer overlay and route traffic to each other using the routing provided by the underlying network layers. The CCN nodes are simply told the IP addresses of their “faces” (see section 2.2.1) and send the traffic to those IP address according to the arrival of data and interest packets.

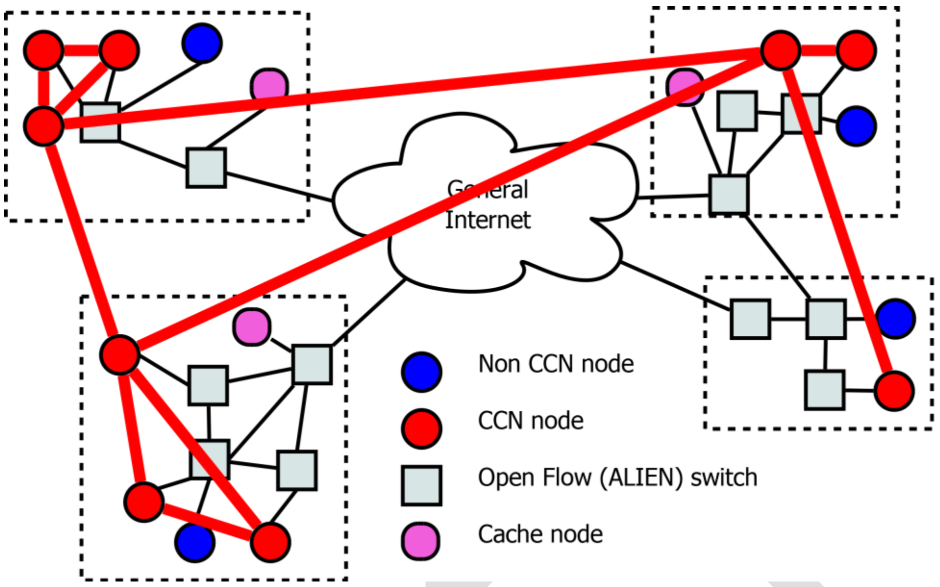


Figure 5-2 CCN acts as an overlay topology over the ALIEN test network

5.1 Basic scenario

In the basic experimental scenario, the cache nodes are not used and the routing is over default routes. This scenario has several requirements which are common to all four scenarios.

- Testbed logical topology – this describes how the whole experiment is to be set up within the OFELIA control framework and which virtual machines are available to be CCN nodes.

- CCN overlay topology – given a list of virtual machines which are to be used to host CCN nodes, these machines must then have the ccnd daemon running and each ccnd must be informed of the IP addresses of other ccnd daemons to which it will connect.

- Initial content – each CCN node will be populated with some initial "content" (this can, in fact, be randomly generated) for the purposes of the trial. This content will need a name, a length and so on.

- Content popularity distributions – each CCN node will then begin to request content from other nodes. A hypothetical popularity distribution will be given for content. Requests will be generated for some content at each CCN node with exponential interarrival times for requests. The exact content to be requested will be determined by the popularity distribution.

The experimental harness must measure the following in the base scenarios and all the scenarios that follow:

- Cache hits and misses within each CCN node (in this scenario there are no dedicated cache nodes but the CCN nodes always have a limited amount of cache).

- Levels of traffic generated by the experiment on each link connected to an OpenFlow Switch and through each OpenFlow enabled switch.

The experiment can vary the number of content items, the popularity distribution and the cache size at each CCN node to determine the effect on the experiment.

The following OpenFlow requirements are made by the CONET application without caches:

- One flow table – to support CONET switching.
- Forward to physical port (Output) – to send interest packets from client to server.
- Matching against Ethernet MAC and type – to recognise clients.
- Matching against VLAN id and priority – because VLANs are not transparent for CONET.
- Drop – for unsupported packets.
- Read-state – for network traffic measurement.
- Port-status – for network monitoring.
- Error – for network monitoring.

In addition the following OpenFlow functions will be useful for the test harness to monitor the network:

- Per table counters.
- Per flow counters.
- Per port counters.

Finally, CONET works in “learning switch” mode, that is to say the switches respond to packets initially by flooding to learn the topology. If CONET is to work in this mode then also needed are the following:

- Forward to all physical ports (Output ALL)
- Forward along spanning tree (Output Flood)

5.2 Alternative routes scenario

In this scenario the OpenFlow routers are used to give alternate paths to CCN and non CCN traffic. This scenario is shown in Figure 5-3. In this scenario the experimental harness is also capable of setting up simple interfering traffic between non CCN nodes in the testbed. This traffic could be as simple as ftp flows between these nodes. This requires the following additional OpenFlow functionality:

- Match against IPv4 address -- to distinguish CCN traffic from regular traffic.

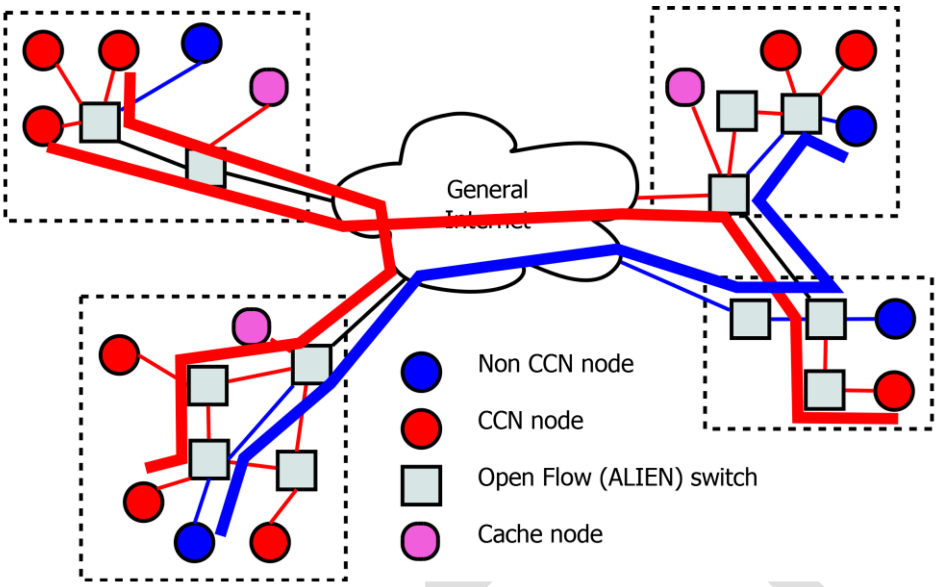


Figure 5-3 Alternative routes scenario

5.3 Cache scenario

In this scenario the extra cache nodes from CONET are used. These nodes provide larger cache storage than the typical CCN nodes. Data traffic between CCN nodes is replicated nearby cache stores in order that the store can populate its cache. This scenario requires extra OpenFlow functionality compared to the base-case scenario (section 5.1):

- Multiple forward – to duplicate the packets (one to the client, one to the server) and cache content into the server
- Modify-field – to rewrite MAC addresses in duplicated packets to the cache server
- Match TCP/UDP ports – content names are mapped to UDP ports
- Modify-state – to instruct the switch about new content.

5.4 Caches and alternative routes scenario

This scenario is a combination of that in 5.2 and 5.3. It should not require any extra functionality other than that of the previous sections combined. However, by combining the experiments it may provide a more demanding test of the OpenFlow implementations.

6 Requirements

This section gives the requirements which the use cases will impose both on the OpenFlow implementation on the ALIEN hardware and also on the extensions which must be made to the CONET module at the FloodLight controller. Section 6.1 lists those requirements imposed by the use case scenarios from the previous chapter which are requirements for OpenFlow functionality which must be implemented in the ALIEN hardware to take part in each use case. Section 6.2 lists requirements on the FloodLight controller which must be made to extend the CONET functionality as described in section 3.4 and. Section 6.3 describes the requirements to create the experimental harness as described in section 5.

6.1 OpenFlow requirements

The CONET CCN application puts some requirements on the OpenFlow implementation of each partner. These capabilities must be present within the OpenFlow implementation on each piece of ALIEN hardware in order to participate in the testbed experiments. The requirements are derived from Table 3-1.

To ensure that the base scenario (section 5.1) works, the implementations of OpenFlow must have the following capabilities:

- One flow table.
- Drop.
- Matching against Ethernet MAC and type – to recognise clients.
- Matching against VLAN id and priority – because VLANs are not transparent for CONET.
- Read-state.
- Port-status.
- Error.

For monitoring in all scenarios the following are also useful:

- Per table counters.
- Per flow counters.
- Per port counters.

For “learning switch” mode:

- Forward to all physical ports (Output ALL).

For the alternative routes scenario (sections 5.2 and 5.4) then the ALIEN hardware must also implement:

- Match against IPv4 address.

If that partner’s ALIEN hardware is to work with the CONET caches (sections 5.3 and 5.4) then the following further capabilities will be required:

- Forward to physical port.
- Multiple forward.
- Modify-field.
- Match TCP/UDP ports.
- Modify-state.

6.2 Requirements from CONET extensions to the Floodlight controller

As well as requirements at each partner site, the testbed experiments will require extensions to the CONET module in the Floodlight controller to handle the extra requirements imposed by the experiments within ALIEN. The following functions are required by extending the CONET functionality:

- Ability to handle multiple caches within the same CONET system.
- Ability to handle topologies with loops within a CONET system.
- As a consequence of both the above requirements, some degree of topology awareness within the CONET system.

6.3 Experimental harness requirements

This section describes the requirements which will be necessary for the experimental harness which will control experiments over the testbed and which will collect the results for further analysis.

- For setting up the CCN topology the following will be required:
    - Read a list of the IP addresses of VMs which will run CCN.
    - Read the structure of the overlay topology formed by these VMs.
    - Install of ccnd on each.
    - Connect ccnd instances to form the CCN topology.
- For populating the content cache:
    - Generate the list of content and which CCN node(s) will initially host each
    - Send the appropriate content to the appropriate nodes.
- For running the experiments:
    - Trigger CCN nodes to request appropriate content at appropriate times.
    - Trigger non CCN nodes to make downloads from each other at appropriate times.
    - Collect data about cache hits and misses.
    - Collect data about traffic on network links monitored by OpenFlow enabled hardware.

7 References

[ALIEN D2.1] http://fp7-alien.eu/files/deliverables/D2.1-ALIEN-final.pdf

[ALIEN D3.1] http://fp7-alien.eu/files/deliverables/D3.1-ALIEN-final.pdf

[CCNx] The CCNx project, http://www.ccnx.org/

[CONET] A. Detti, N. Blefari-Melazzi, S. Salsano and M. Pomposini, “CONET: A Content Centric Inter-Networking Architecture”, ACM SIGCOMM Workshop on Information-Centric Networking (ICN-2011), Toronto, Canada, 2011

[DONA] T. Koponen, M. Chawla, B-G. Chun, A. Ermolinskiy, K. H. Kim, S. Shenker and I. Stoica, “A Data-oriented (and Beyond) Network Architecture,” in Proc. ACM SIGCOMM ’07, Kyoto, Japan, Aug. 2007.

[i3] I. Stoica, D. Adkins, S. Zhuang, S. Shenker and S. Surana, “Internet Indirection Infrastructure,” in Proc. of ACM SIGCOMM ’02, pp. 73-86, Pittsburgh, PA, USA, Aug. 2002.

[ICF SDN] L. Veltri, G. Morabito, S. Salsano, N. Blefari-Melazzi and A. Detti, “Supporting Information-Centric Functionality in Software Defined Networks”, SDN’12 Workshop on Software Defined Networks (ICC 2012), Ottawa, Canada, 2012.

[ICN SDN OF] N. Blefari-Melazzi, A. Detti, G. Morabito, S. Salsano, and L. Veltri, “Information Centric Networking over SDN and OpenFlow: Architectural Aspects and Experiments on the OFELIA Testbed”, http://arxiv.org/abs/1301.5933, January 2013.

[LIPSIN] P. Jokela, A. Zahemszky, C. E. Rothenberg, S. Arianfar and P. Nikander, “LIPSIN: Line Speed Publish/Subscribe Inter-networking”, in Proc. ACM SIGCOMM ’09, Barcelona, Spain, August 2009.

[NDN] V. Jacobson, D. K. Smetters, J. D. Thornton, M. F. Plass, N. H. Briggs, and R. L. Braynard, “Networking Named Content,” in CoNEXT ’09. New York, NY, USA: ACM, 2009, pp. 1–12.

[OF ICN] N. Blefari-Melazzi, A. Detti, G. Mazza, G. Morabito, S. Salsano and L. Veltri, “An OpenFlow-based Testbed for Information Centric Networking”, Future Network & Mobile Summit 2012, Berlin, Germany, July 2012.

[PSIRP] Tarkuma, Sasu et al., “The Publish/Subscribe Internet Routing Paradigm (PSIRP): Designing the Future Internet Architecture”, IOS Press, 2009.