# OFELIA

ICT-258365

## Deliverable 9.1

## EXOTIC final architecture and design

| | |
|---|---|
| Editor: | *Stefano Salsano (CNIT/University of Rome Tor Vergata)* |
| Work Package (leader) | WP9 (Stefano Salsano, *CNIT/University of Rome Tor Vergata*) |
| Deliverable nature: | Report (R) |
| Dissemination level: (Confidentiality) | Public (PU) |
| Contractual delivery date: | 31/10/12 |
| Actual delivery date: | 21/12/2012 |
| Version: | 1.0 |
| Total number of pages: | 42 |
| Keywords: | OFELIA, FP7, Information Centric Networking, OpenFlow |

 

Disclaimer



 

**Imprint**

| [Project title] | *OpenFlow in Europe – Linking Infrastructure and Applications* |
|---|---|
| [short title] | *OFELIA* |
| [Number and title of work package] | *WP9 – Support of Content centric networking functionality* |
| [Document title] | *D9.1 EXOTIC final architecture and design* |
| [Editor] | *Stefano Salsano (CNIT/University of Rome Tor Vergata)* |
| [Work package leader] | *Stefano Salsano (CNIT/University of Rome Tor Vergata)* |
| [Task leader] | *Stefano Salsano (CNIT/University of Rome Tor Vergata)* |
| [PM (estimated)] | *10* |
| [PM (consumed)] | *14* |

**Copyright notice**

# Executive summary

The goal of the OFELIA WP9 is to design, implement and validate a solution for supporting Information Centric Networking using Software Defined Networking. In particular, the solution has to be demonstrated on the OFELIA testbed.

*Information Centric Networking* (ICN) has been proposed as a new networking paradigm in which the network provides users with content instead of communication channels between hosts. The *Software Defined Networking* (SDN) approach promises to enable the continuous evolution of networking architectures. In this light, an SDN enabled network could support ICN functionality without the need to re-deploy new ICN capable equipment.

In this document, we propose and discuss solutions to support ICN using SDN concepts. We focus on an ICN framework called CONET, which grounds its roots in the CCN/NDN architecture. We address the problem in two complementary ways. First we discuss a general and long term solution based on SDN concepts without taking into account specific limitations of SDN standards and equipment. Then we focus on an experiment to support ICN functionality over the OFELIA large scale SDN testbed based on OpenFlow. The current OFELIA testbed is based upon OpenFlow 1.0 equipment from a variety of vendors. Therefore, we design an experiment taking into account the features that are currently available on off-the-shelf OpenFlow equipment.

 

# List of authors

| Organisation/Company | Authors |
|---|---|
| **CNIT** | Stefano Salsano, Nicola Blefari-Melazzi, Luca Veltri, Stefano Brogi, Andrea Araldo, Fabio Patriarca, Marco Bonola, Salvatore Signorello |
| **Lancaster Univ.** | Matthew Broadbent |
| | |
| | |

 

# Table of Contents



 

# List of figures and/or list of tables



 

# Abbreviations

| | |
|---|---|
| CONET | COntent NETwork |
| iC | Interfaces between two Controllers (NRS Nodes). It can be an inter-domain interface |
| iM | "North-bound" interface between a Controller/NRS Node and a node implementing Management / OSS / Orchestrator / UI functions |
| iN | Interface between an ICN Node and the Controller (NRS Node), it is an extended OpenFlow interface. |
| iS | Interface between a Serving Node and the Controller (NRS Node), it is an extended OpenFlow interface. |
| IC-AS | Information Centric – Autonomous System |
| ICN | Information Centric Networks |
| NRS | Name Routing System |
| OF | OpenFlow |

# 1 Introduction

The shift from “host-centric” networking to “Information Centric” or “Content-centric” networking has been proposed in several papers (e.g. [1][2]) and is now the focus of an increasing number of research projects ([3][4][5][6]). In short, with Information Centric Networking (ICN), the network provides users with content, instead of providing communication channels between hosts. With ICN, the network becomes aware of the content that it is providing, rather than just transferring it amongst end-points. As a result, it can introduce features such as in-network caching or content-based service differentiation. The biggest advantages of ICN can turn to reality when network nodes natively support ICN mechanisms. Deploying ICN capable equipment into existing networks is a critical issue as it may require the replacement or update of existing running equipment. In this context, we believe that Software Defined Networking (SDN) is an important opportunity as it promises to enable the continuous evolution of networking architectures. This is the first important motivation to provide ICN functionality using SDN. An SDN enabled network could facilitate the introduction of ICN functionality, without requiring re-deployment of new ICN capable hardware.

A second motivation for the integration of these two paradigms can be found in our specific approach to ICN. The framework we have defined, called CONET (COntent NETwork) [8] [9], is based on the interaction of the forwarding nodes with nodes belonging to a routing layer (called NRS: Name Routing System). We believe that such architecture can help in overcoming some critical issues of ICNs, such as scalability. The forwarding nodes are instructed by the NRS nodes and this approach of separation between the forwarding plane and a “control plane” naturally maps into the SDN architecture.

Probably one of the most popular implementations of the SDN paradigm is OpenFlow [16][17]. In OpenFlow, network nodes are called OpenFlow Switches and are responsible for forwarding, whilst routing decisions and control functions can be delegated to a centralized element called a “Controller”. OpenFlow Switches communicate with the Controller through the OpenFlow protocol. The OpenFlow protocol is already implemented in a number of commercial products, and available in several research and prototyping projects. Started as a research/academic project, OpenFlow evolution and standardization efforts are now covered by the industry forum named the *Open Networking Foundation* (ONF).

In this document, we first consider support for ICN using SDN concepts in a medium-long term perspective, with no reference to the limitation of current OpenFlow specifications and available equipment. We assume an OpenFlow like architecture, in which the Controller nodes can process events coming from forwarding nodes and can instruct the forwarding nodes both in a reactive and in a proactive way. In this approach, we define the set of supported operations and the capability of forwarding nodes according to the needs of the ICN scenarios. In the definition of the proposed enhancements, we start from the latest OpenFlow specification (v1.3). An initial description of the issues relating to the long term approach has been presented in [10]. A testbed implementation of the proposed solutions is ongoing, using software based switches (Open vSwitch).

 

In section 6 of this document, we consider a solution to support ICN using the OpenFlow v1.0 specification and equipment, as was first discussed in [11]. This solution has been fully implemented and tested in the OpenFlow v1.0 testbed provided by the OFELIA project.

# 2 The CONET ICN solution

Figure 1 shows an overview of the CONET architecture. The "terminals" are called E*nd-Nodes* and *Serving Nodes*. End-Nodes require content using CONET protocols, using the name of the content to be retrieved. Serving Nodes are the originators/providers of the content.

Using the same terminology as CCN [2], the content requests are called "interests" or "interest packets". The content requests are forwarded over the ICN network, taking into account the requested content-name. When a node that contains the content receives the content request, it will reply with a "data" packet that will be sent back towards the requesting node. Using more specific terminology, the data units exchanged by the CONET nodes are called "CONET Information Units" (CIUs): the End-Nodes request for content using *interest* CIUs which are sent towards the Serving Nodes; the Serving nodes deliver the content using chunks (i.e., part of files, videos, etc.), called *named-data* CIUs. The key entities involved in these operations are the *Border Nodes*, which: i) forward content-requests from End Nodes to Serving Nodes, implementing the ICN "forward-by-name" mechanism; ii) deliver content data from Serving Nodes to End Nodes, implementing the ICN "data forwarding" mechanism; iii) may cache content and therefore provide it to End Nodes without forwarding the requests to Serving Nodes. In this case, the Border Nodes perform security checks as not to store and redistribute unauthorised content.

We now introduce the concept of the CONET sub-system, denoted as CSS. The nodes (End-Nodes, Serving Nodes and Border Nodes) that belong to the same CSS can directly exchange interest and data packets. The nodes that belong to different CSSs will be interconnected by other Border Nodes. To form an analogy with IP networks, CONET Sub-Systems are the equivalent of IP subnets, while Border Nodes are the equivalent of IP routers. A CSS exploits underlying technology to transfer requests and data among its nodes and CONET is an inter-network that interconnects CSSs. A CCS can be, for example: two nodes connected by a point-to-point link, a layer 2 network like Ethernet, a layer 3 network (such as a private IPv4 or IPv6 network or a whole Autonomous System), or an overlay UDP link (socket) among two nodes.

An Information Centric Autonomous System (IC-AS) includes a set of CSSs under the same administrative domain. The interconnection of different IC-ASs is performed by special Border Nodes called Border Gateway Nodes (BGN). Maintaining the analogy with IP networks, the IC-AS are the equivalent of the IP Autonomous System that constitute today's Internet.

The *NRS Nodes (Name Routing System)* assist the Border Nodes in performing the forward-by-name operation. They are responsible for the content-routing mechanisms, both at the intra-domain and inter-domain level (as discussed below). The NRS functionality can be seen as logically centralized within one IC-AS. Scalability and reliability issues will obviously drive the deployment of NRSs in a distributed and redundant way, but these aspects are not yet addressed in our prototypes. The introduction of NRS Nodes in the architecture is fully aligned with the SDN approach of using "controllers" to drive the forwarding behaviour of switches/routers and to enforce an explicit separation between a data forwarding plane and a control plane. Though we do not necessarily rely on OpenFlow technology for deploying CONET architecture, we have experimented using a CONET implementation based on such technology.

 

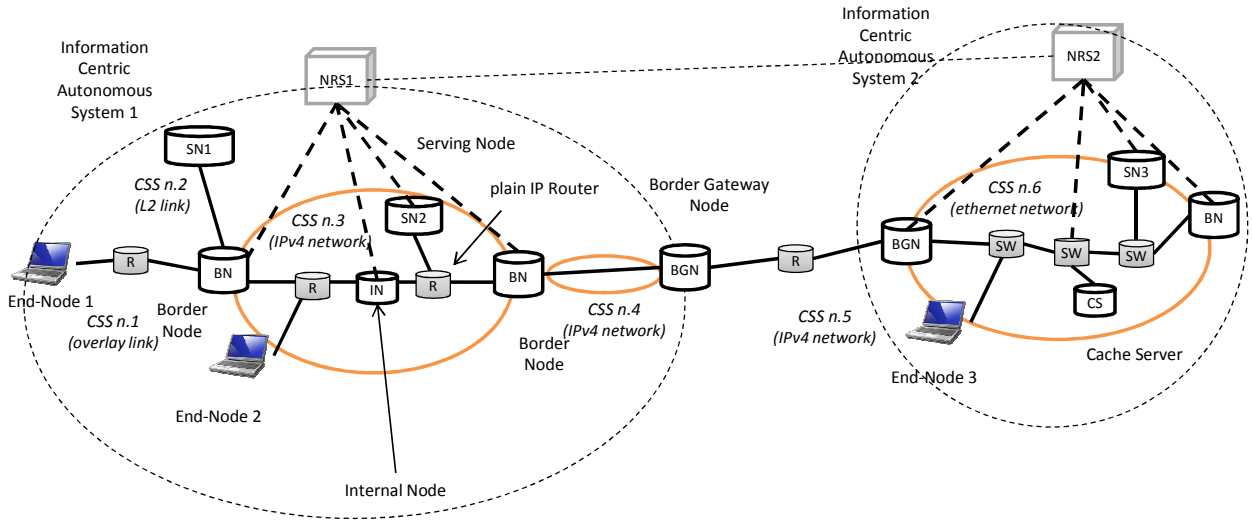


**Figure 1: CONET Architecture**

As shown in Figure 1, Border Nodes interconnect different CSSs. Therefore the end-to-end forward-by-name process can be seen as the process of finding a sequence of Border Nodes from the End-Node up to the Serving Node.

When a CSS is an IP network (e.g. CSS n. 3 in Figure 1), the downstream Border Node performs a forward-by-name procedure to resolve a content name and gets as a result the IP address of the upstream Border node. The interest packet is sent using such IP address. An IP CSS can be composed of an arbitrary number of plain IP Routers that do not perform ICN operations. Moreover, we define *Internal Nodes* (IN). These can be seen as "enhanced" IP routers that can perform content caching but are not able to perform "forward-by-name" (only plain IP routing).

In the case of CSS n. 6, there is a "Cache Server" node. This node is used, as a temporary solution, when it is not possible to replace legacy nodes with ICN nodes that are capable of caching. In the experiments section, we will present a use case involving such Cache Servers.

## 2.1 Naming

The issue of naming in an ICN, i.e. which identifiers should be used to identify and then route content, is a very hot topic and different proposals are still under consideration. The proposals belong to two different "families": one foresees that the names should be "human readable"; another one foresees that the name needs not to be human readable and therefore they may have other interesting security properties (such as being "self-certified"). We believe that our solution should be agnostic in this respect and to design an ICN that can support different naming schemas. Therefore, we define the name as a tuple <*namespace ID*, *name*>. The *namespace ID* determines the format of the *name* field. Thus, the *name* field is a namespace-specific string. Each namespace follows its own rules to release unique *names* with its own format. ICN-ID namespace values to be used in the future public Internet should be assigned by an authoritative registration authority, such is this case with IANA and global IP address allocation.

In our current prototype, we support the naming proposed by the CCN/NDN architecture. The *content name* is a string, such as: www.cnn.com/football. This can be human readable, such as in this example. We can see it as the composition of a *principal* (www.cnn.com) and a *label* (/football) [1]. Such naming schema allows us to aggregate routes and to perform longest prefix matching. We also support *self-certifying*

*names* [1], by using a public key as the string of the principal. Therefore CONET supports both human-readable and self-certifying names.

## 2.2 Forward-by-name operations – Lookup and cache

CONET forward-by-name operations are based on a name-based lookup table and on a prefix matching algorithm. When interest packets arrive, the name-lookup operation has to be performed at line-speed, using a table that stores the associations between name prefixes and next hop. This table is called FIB (Forwarding Information Base), like in IP routers. Moreover, another table, called RIB (Routing Information Base), must be handled by the system. This is needed to handle the exchange of routing information with other nodes. The RIB does not need to be accessed at line speed, and can be larger (e.g. two or three times) than the FIB, as further prefix aggregation can be performed when evaluating the FIB from the RIB. Let us consider a worldwide CONET deployment, in which CONET would be used to fetch current Web content and principal identifiers are the current second-level domain names (www.cnn.com). With these hypotheses, we need to store in the FIB/RIB tens of billions of name-based routes, due to the high volume of Web content and the limited potential for aggregation with their names. Consequently, re-using the existing architecture of an IP router would result in two severe scalability problems. First, the current FIB technology is unable to contain all name-based routes (see [28]). Second, implementing large RIBs requires prohibitively expensive hardware.

In order to overcome these scalability problems, we propose the use of a FIB Forwarding Engine to cache routes and employ a centralized routing logic that serves all the nodes of a sub-system, managing the full routing table (RIB).

A typical sequence of Lookup-and-Cache forwarding operations is shown in the upper half of Figure 2. Border Node N1 receives an Interest message for “ccn.com/text1.txt/chunk1, segment 1”. Since the FIB lacks the related route, the node temporarily queues the Interest message, looks up the route in a remote RIB, gets the routing information from the RIB and stores it in the FIB. It can then forward the Interest message. In what follows, we discuss the rationale underlying this approach.

**FIB as a route cache**– The relative frequency with which Web content is accessed follows Zipf’s law, while the interest in specific Web content is mostly restricted to time and space. Therefore, a large proportion of Interest messages that a CONET node would have to concurrently forward-by-name, only refers to a small set of content. We propose to use the FIB of a CONET node as a route cache, which should contain, at a minimum, the entire set of active-routes. When the FIB lacks a route, the node looks up the route in a “remote” RIB and caches the route in its FIB. When all FIB entries are filled, new routing entries replace old ones, according to a specific route replacement algorithm, e.g. LRU.

**Centralized Routing Engine**–In our model, all named-based routes are contained in a RIB, which logically serves all nodes of a sub-system and runs on a centralized server, called the Name Routing System (NRS) node. Thus, the expensive Routing Engine can be logically centralized rather than replicated on all the forwarding nodes (see for example [22], which proposed the so-called Routing Control Platforms). Since many Interest flows use a small set of active-routes, the temporal dynamics of active-routes is slower than the flow dynamics. This means that routes can be used for longer than the duration of a single flow, limiting the lookup rate that a centralized Routing Engine has to support.

 

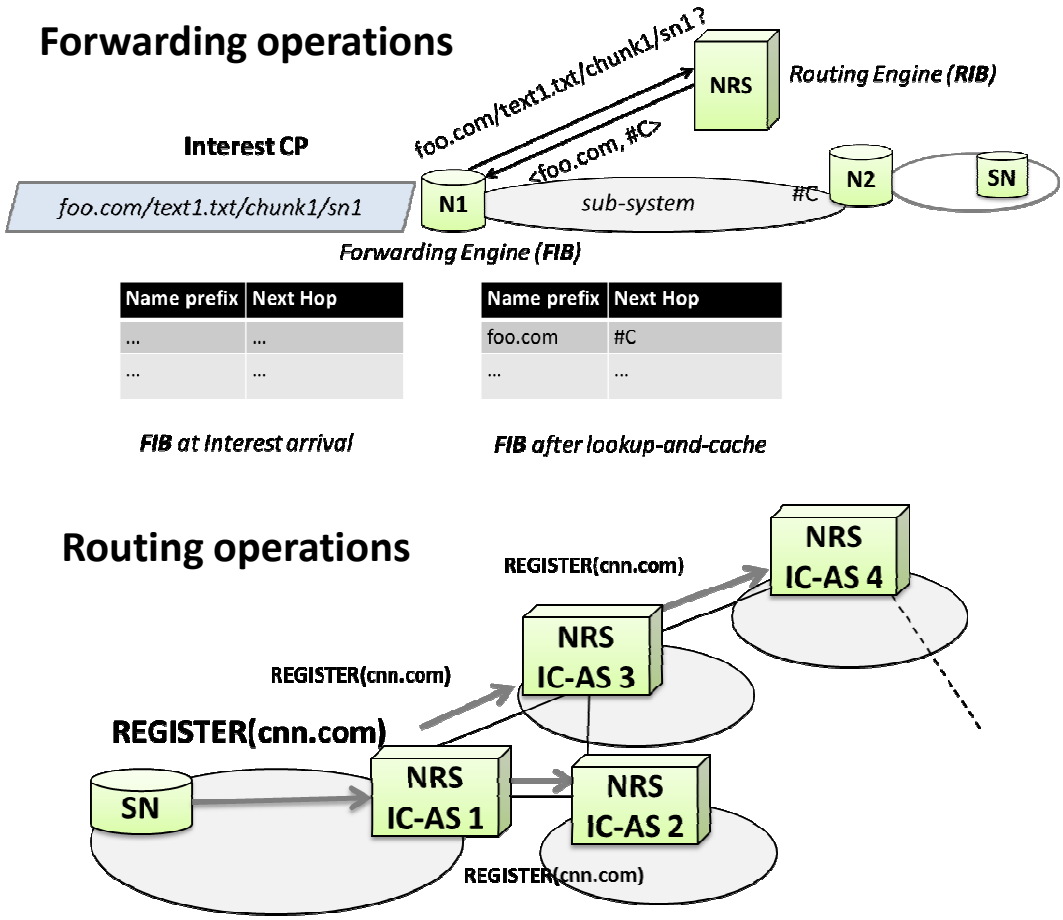


**Figure 2- Lookup and cache: forwarding and routing operations**

## 2.3 Content routing (inter-domain and intra-domain)

In the previous section we have described the "forwarding-plane", i.e. the procedures carried out to forward Interest messages. However, the Lookup-and-Cache architecture also needs "routing-plane" procedures that run on NRS nodes and whose goal is to establish the RIBs.

Intra-CSS routes connect Border Nodes with Border or Serving Nodes of the same CSS. Inter-CSS routes connect BNs of different CSSs. As depicted in Figure 1, a single NRS node may serve the whole set of CSSs administrated by a single IC-AS. The NRS also distributes inter-domain routes, which consist of Inter-CSS routes connecting the Border Gateway Nodes of different ASs. The computation of intra-domain routes, i.e. intra-CSS and inter-CSS routes internal to the AS, is done by the NRS based upon its knowledge of the AS network topology. It may also take into consideration proprietary name-based traffic engineering strategies. For instance, the content that has the cnn.com prefix could be routed on a path which may be different from the path used to route youtube.com, albeit with identical egress Border Gateway Nodes.

The CONET inter-domain routing approach is conceptually similar to the current BGP inter-domain routing, in which the different ASs advertise the reachability information of a network prefix, along with the AS path used to reach that prefix. The use of a BGP-like approach for name-based routing was already proposed in [23]. Recently, it has been proposed to extend BGP to disseminate advertisement information about content objects in the form of URIs [24]. We are currently investigating the possibility of directly extending BGP to disseminate inter-domain routing information. Our current CONET implementation is based on a

simplified approach with respect to the full BGP protocol. It disseminates name-prefixes using “REGISTER” and “UNREGISTER” messages. A similar mechanism was proposed by the DONA architecture [1].

In the bottom half of Figure 2, a CONET Serving Node (SN) REGISTERs the name-prefixes (e.g., “cnn.com”) of the content it provides in the local NRS node. The local NRS node, in addition to subsequent nodes, forward the REGISTER messages toward their peers. Using REGISTER and UNREGISTER messages, a NRS node can compute the next BN on the path toward the destination and accordingly configure the RIB.

The scalability of this approach to name-based inter-domain routing is still a research topic under investigation. It is worth noting that in our approach, we do not request that all Border Nodes exchange routing information and maintain a full copy of the RIB. In this case, they delegate this functionality to the logically centralized Routing logic in the NRS node, which can be properly dimensioned to manage the name-based inter-domain routing information.

## 2.4 CONET segmentation and transport protocol

Content to be transported over an ICN can be very variable in size, from few bytes to hundreds of Gigabytes (and it is not easy to set an upper bound for content size). Therefore, it needs to be segmented into smaller data units, typically called *chunks*, in order to be handled by ICN nodes. A chunk is the basic data unit to which caching and security is applied. This means that: i) a single chunk of a large content object can be requested by an End-Node; ii) single chunks will be *signed* by the origin Server-Node for security reasons; iii) Border and Intermediate Nodes will authenticate chunks and store them in their caches as needed.

Each chunk contains an amount of overhead, mostly due to the security information (e.g. the signature needed to authenticate the chunk), which is approximately independent of the chunk size[1]. Taking the CCNx [7] implementation as reference, this overhead is in the order of 700 bytes per chunk. Each signature and authentication operation can be computationally expensive. These considerations on the overhead and on the number of cryptographic verification operations to be performed by caching nodes suggest the chunk size to be rather large: e.g. from tens of KBs up to 1 or 2 MBs. When chunks of such relatively large size are exchanged amongst ICN nodes, they need to be fragmented to match the Maximum Transfer Unit imposed by level 2 networking technologies. As shown in Figure 3, we propose to handle the segmentation of content with two levels: at the first level, content is segmented into chunks, and we define data units called CIUs (CONET Information Units). At the second level, chunks are segmented into smaller data units (*Carrier-Packets*). The transfer of Carrier Packets is regulated by our proposed “Information Centric Transport Protocol” (ICTP) [25] [26]. ICTP performs reliability and congestion control, much like TCP does for the transfer of a byte stream in current TCP/IP networks. ICTP is based on the principles described in [2]. It is a receiver-driven approach implementing the same algorithms as TCP (slow-start, congestion avoidance, fast retransmit, fast recovery). ICTP is TCP friendly and can achieve fairness amongst multiple competing ICN content downloads. Furthermore, ICN can fairly handle ICN content downloads competing with regular TCP flows. In [25] we analyze the limitations of the current implementation of the CCNx transport protocol, and evaluate the performance gains possible with ICTP. For example, and as shown in Figure 4, the goodput of the CCNx transport protocol sharply decreases with packet loss, due to the fact that chunks are segmented at IP level. The loss of a single segment implies the loss of the full chunk that

[1] For example the cost is mostly independent from the chunk size if asymmetric cryptography of chunk hashes is used.

 

must be retransmitted by the transport protocol. Furthermore, ICTP handles the retransmission of single segments, similar to TCP. In a competition with regular TCP flows, CCNx flows will be starved due to their excessive reaction to these loss events.

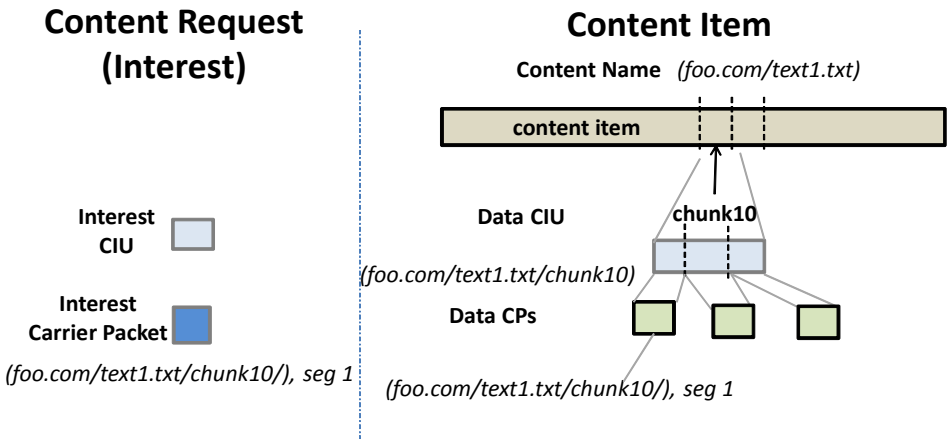


**Figure 3- CONET Information Units (CIUs) and Carrier-Packets**

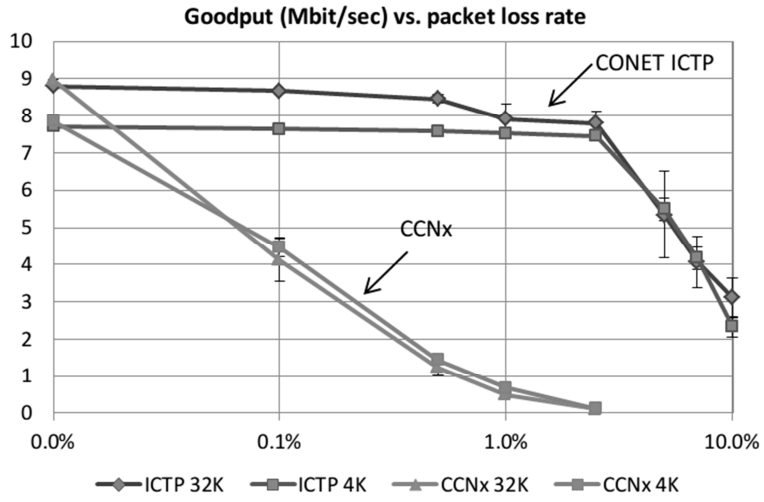


**Figure 4- Transport protocol: CONET ICTP vs. current CCNx implementation**

## 2.5 Considerations on different approaches for ICN design and deployment

The "clean slate" approach for ICN aims at fully replacing the existing IP layer, i.e. assuming that the ICN layer will sit over layer 2 networks like Ethernet. On the other hand, we see two main drivers for an "integration" approach towards ICN, in which ICN solutions operate on, and interwork with, IP networks. In order to present the first driver, let us define the "content-oriented" and "conversational" communication patterns.

A "content-oriented" communication pattern takes place when a user (or a user application) needs a specific content irrespective from where the content is stored. The content is associated with an identifier (the "content name") unique over a certain namespace. The goal of an ICN is to provide native and efficient support for this communication pattern. A "conversational" communication pattern takes place when a user / user application wants to exchange information in a bidirectional way with a remote entity. This pattern clearly applies to audio/video calls, but also a range of current applications and services deployed on the web. For example, most social networking applications require a continuous exchange of

information from the client (the browser) to the server (or among clients). As such, they exhibit characteristics akin to "content-oriented" communication patterns, as well as conversational patterns.

In our opinion, the first driver for the "integration" approach is that the *conversational communication pattern still remains very important* because of the number of applications and use cases that rely on it. Both the "content-oriented" and the "conversational" patterns need to be addressed in future Internet designs, with the aim to provide an efficient support to both ones. Promoters of clean slate approaches for ICN argue that the conversational pattern would be supported by ICN mechanisms (as in [27]). We do not see any apparent benefits in changing the way that this communication pattern is supported by current IP based networks. The second, and maybe most important driver, is that having an *evolutionary path from existing IP networks could be the only path for a realistic adoption of ICN*. Existing prototype implementations of "clean slate" ICN solutions, such as CCNx [7], mostly rely on an "overlay" approach to run on top of existing IP based networks. This overlay approach works by tunnelling ICN information units within TCP or UDP flows running over IP. The authors of [7] suggest that this is a short-term temporary approach, needed to deploy ICN implementations over current networks. Indeed, we believe that this demonstrates the need for an ICN solution to operate on, and to be compatible with, existing IP networks. It is unrealistic to believe that existing IP networks will simply be dismantled overnight in order to migrate to ICN. Therefore we propose an ICN design that evolves from current IP technology and even leverages existing IP infrastructure.

In principle, CONET may be deployed following all three approaches, which are not mutually exclusive but can be combined:

- *overlay approach*: built on top of the IP layer, CSSs are coupled nodes connected by overlay links, e.g. UDP/IP tunnels (see the CSS n.1 in Figure 1); this is the approach used in CCNx implementation [7]
- *clean slate approach*: built on top of layer-2 technologies (e.g. Ethernet, PPP, MPLS LSP), CSSs are nodes connected by layer-2 links/networks, with ICN replacing the IP layer (see the CSS n.2 in Figure 1);
- *integration approach*: ICN functionality is integrated into the IP layer (see the CSS n. 3 to 6 in Figure 1). This can be done either by means of a novel IP option defined for IPv4 and IPv6 [12], or by including ICN information in an ICN transport layer.

Taking into account the previous considerations, we support and focus on the integration approach.

## 2.6 CCNx and CONET compatibility

Our CONET implementation is based on the CCNx code [7]. Considering that the CCNx implementation is widely used in the ICN research community, we wanted to ensure compatibility with it. In particular, we maintained the same API towards the applications. This allows reuse of existing CCNx applications, as shown in the option 1) of Figure 5. Under option 1), the CCNx client running in the ICN client and the CCNx server running in the ICN server use the CCNx API implemented by the CONET module, whilst the communication from the client through the ICN node towards the server is done with the CONET protocols.

 

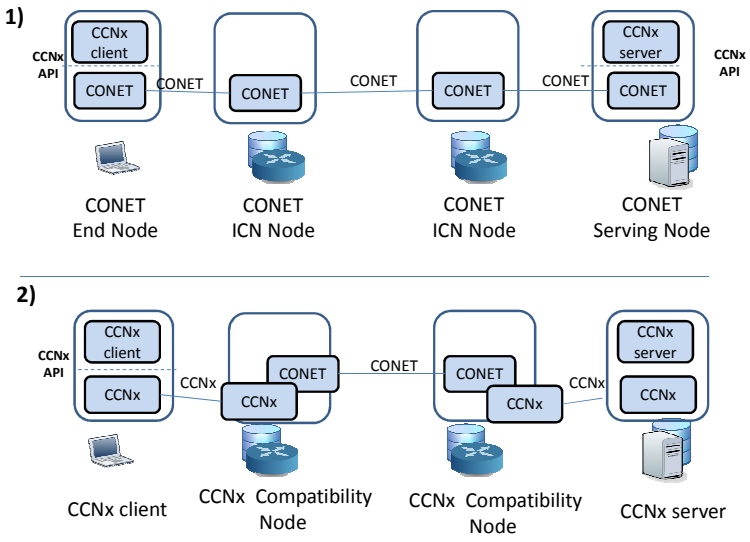


**Figure 5- Two different compatibility models with CCNx**

We also support a second compatibility model with CCNx, shown as option 2) in Figure 5. In this model, CCNx clients and/or servers can use the CCNx protocol "over the wire" to communicate with a CCNx Compatibility Node that acts as a gateway towards other ICN nodes using CONET protocol.

# 3 Extending OpenFlow to support CONET ICN

Following the SDN approach, we consider an OpenFlow-based ICN architecture in which the intelligence of the ICN is de-coupled from the forwarding (of interest and data packets) and caching functions. As shown in Figure 6, this results in an architecture that is composed by two different planes: i) a data plane with the Serving Nodes (i.e. the content producers), the End-Nodes (i.e. the content requesters/consumers) and the different types of ICN network nodes we have discussed in section 2; ii) a control plane that includes both the Name Routing System (composed by NRS Nodes) and a security infrastructure (e.g. a PKI - Public Key Infrastructure). The two planes communicate through an *extended OpenFlow* interface, used by the NRS nodes (acting as OpenFlow Controllers) to control one or more ICN nodes. In this architecture, the name-based routing intelligence needed to implement ICN functionality runs in the NRS, implemented as a set of OpenFlow controllers. We also introduce in the architecture an "Orchestrator/OSS/UI" node that will talk with NRS nodes. The controllers/NRS Nodes offer a "North-Bound" API towards this element that orchestrates the overall behavior of the network.

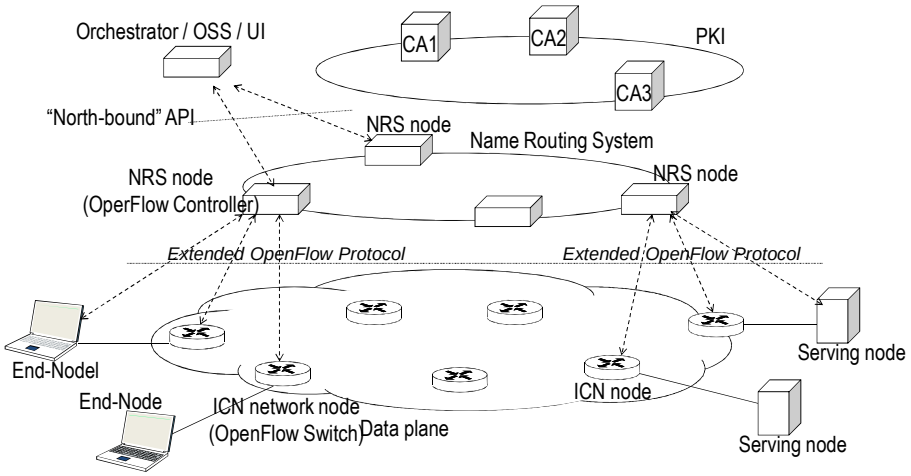


**Figure 6- Information Centric Network based on extended OpenFlow**

## 3.1 OpenFlow protocol extension

Currently, the OpenFlow protocol is used to configure a set of matching rules in a switch flow table and the actions to be performed on the packets. In our approach, we support a broader set of capabilities and we need to extend the semantic of the protocol. Therefore we need to define new messages that, extending the current message types provided by OpenFlow, allow the support of ICN-related operations and parameters. These protocol extensions will be discussed in detail in section 5.

Hereafter, we will classify and discuss the functional requirements of an ICN node that we support, like reactive or proactive caching, intelligent forwarding-by-names of interest messages and security.

We extend the OpenFlow protocol operations in order to handle the concept of "content" and to support new content-related methods, such as key management, caching, routing-by-name, etc.. The new set of ICN-specific operations that can be supported by an ICN node and by the corresponding control protocol, can be classified (and hereafter summarized) as: "content-related", "routing-related", "tagging-related" and "security-related".

 

### 3.1.1 Routing-related operations

An ICN adopts an addressing scheme based on names, which do not include references to their location; as a consequence, interest messages have to be routed toward the closest copy of the content, based on a "destination" content-name. The main routing-related operations are:

*Forwarding-by-name* – When receiving a new interest message, the ICN node is expected to forward it to a next-hop node on the basis of the targeted content name. This forwarding operation is performed through the name-based Forwarding table present within the ICN node that maps possible content names to the corresponding next-hops (the FIB described in section 2.2). The insertion of new entries in such forwarding table is the responsibility of the NRS nodes/controllers. When new content is requested with a name that is not in the local forwarding table, the CONET Node requests the name-to-next-hop lookup to the controller, using the "lookup-and-cache" mechanisms described in section 2.2. If a route is found and returned, this is stored in the forwarding table. Particular care should be taken when the forwarding table is full and a previous entry has to be deleted and replaced by the new one. Such decision may be assisted by the controller (for example through some priority information), or can be autonomously taken by the node on a basis of fixed and simple rules (e.g. "do remove the oldest entry").

*Forwarding table management* – Such forwarding table is dynamically updated each time a name-to-next-hop is requested to and returned from the controller. However the controller is expected to have the possibility to populate and modify the forwarding table according to some upper level strategy (for example by distributing the forwarding-by-name information for the most popular contents, or for contents that require some form of prioritization). Such operations can be driven by the controller/NRS node and executed asynchronously with respect to the incoming packet events.

*Forwarding table exportation* – In an instance where the controller does not keep a copy of the routing table of each node attached to it (e.g. for scalability reasons), it is required that the ICN nodes send to the controller their current routing-by-name table.

### 3.1.2 Content-related operations

*Content publications* – An ICN Serving Node needs to publish, using the Name Routing System, the content names that it can serve. This information will be used by the NRS in the routing procedures.

*Caching decisions* – An ICN node may provide caching functionality for achieving a native, in-network caching function. This may drastically improve multicast content distribution (when the same content has to be distributed from one source to multiple destinations), and more in general, allows for a more efficient content delivery in both fixed and mobile environments [30] (when the same content is successively requested by other destinations). Due to the large amount of packets (and content) that a node is requested to forward, it is expected that a node decides to cache only a portion of the overall forwarded contents. Such decision of which content is to be cached and which is not, can be made locally inside the node or, more likely, by relying on the NRS Node. A further decision related to caching, is on which content has to be removed from cache when the cache is full and new content has to be added. This decision could be delegated to the external controller/NRS Node.

*Caching notification* – It could be expected that the controller is notified when a particular chunk of content has been cached by a node. This creates an updated map within the controller of the availability of content.

*Proactive caching* – The controller can proactively push content to an ICN node, therefore anticipating the "automatic" caching procedures that are the sole solution if a purely distributed approach (i.e. without OpenFlow) is used. These "proactive push" approaches could prove very useful in the case of live content distribution within the ICN (e.g. audio/video real-time streaming).

 

*Interest handling* – An ICN node handles incoming content interest messages that have to be routed based on a forwarding-by-name strategy. During this phase, the node will also keep track of requested content (for example, if a "Pending interest" state is used to forward back the content data) or it may keep track of the requests to optimize the processing of further interest requests for the same content received from other interesting end-nodes. The ICN node could keep track locally or delegate such operation to an external controller. In case of stateless interest processing (no information is maintained for the already processed interest requests), the node may still want to inform the controller about the received requests in order to better influence caching decisions. This may either be performed : i) by notifying the controller on a per-request basis (every time a new interest message arrives) or ii) on a batch basis (sending to the controller periodic summary information reports about the requested content). Thus, the controller can build ordered lists of the most popular content: useful for advising subsequent caching decisions.

### 3.1.3 Tagging-related operations

In general, ICN operations rely on name based information stored within the packets. In addition to the information of "global" significance which is inserted by End-Nodes and Serving Nodes, information of "local" validity within a given domain can be used. Some information "tags" can be added by ingress nodes when packets enter a given domain and then removed by egress nodes when packets go out of the domain.

*Name-to-tag mapping*– An ICN node can request a tag for a given piece of content from the NRS node. Moreover, an NRS node could asynchronously "inject" a name-to-tag mapping into an ICN node.

### 3.1.4 Security-related operations

An ICN node is expected to exploit security information embedded in the content to avoid the diffusion of fake versions of content and also protect the content. This is opposed to exploiting connection-based or application-based security [31].

*Security enforcement* – Content (or content chunks, like in CONET) are cryptographically protected in order to assure content (and content generator) authenticity and data integrity. This security service is provided through digital signature signing and can be verified through the public key associated to the private key of the content (or of the content generator). Every ICN node should verify such signature before forwarding the content toward the interested end-nodes, in order to protect the network against DoS or other similar attacks. Such function in turn requires that the ICN node obtain the public key associated to the content. One solution involves the NRS node (which acts as controller) providing the public key together with routing information (see next point). Other possible solutions could be to use identity-based cryptography, or self-certifying names. The investigation as to which solution fits best with the ICN paradigm and requirements is out the scope of this paper. The CONET framework can support all of these solutions.

*Key management and distribution* – In case human readable names are partially used, and an association between names and public keys is required, this should be executed by the controller NRS node, according to a proper key management mechanism. This would typically be through the use of a public key infrastructure, such as PKI or Web of Trust. The result of such use will inform the NRS node as to the correct name-to-key associations and enable it to pass this information to the ICN nodes.

*Key revocation* – In parallel to a suitable key management and distribution mechanism, a key revocation mechanism should also be implemented. This allows the revocation of compromised or withdrawn keys. As a part of this mechanism, the NRS node will still have the authority to communicate such revocation information to the ICN node.

 

## 3.2 Reference Interfaces

Figure 7 shows the set of reference interfaces for the design of our solution.

The interface iN lies between the NRS Node and an ICN Node, which according to Figure 1 can be a Border Node, an Internal Node, or a Border Gateway Node. Figure 7 shows two NRS Nodes, and represents a scenario with two domains (two IC-ASs).

The two NRS nodes are interconnected by the iC interface, which is not an OpenFlow interface. As such, the definition of the iC interface is out of the scope of this document.

The interface iS is placed between the ICN Serving Node and the NRS node. This interface is used by the Serving Nodes to publish into the NRS the information related to the hosted contents. We have designed the interface iS as an extended OpenFlow interface.

We assume that an NRS Node/OF controller offers the iM interface towards a "Management" node playing the role of orchestrator and offering User Interfaces towards network managers. This is often referred as "North-bound" interface in the SDN jargon. In our work we implemented the iM interface extending the REST API provided by the Floodlight OpenFlow controller.

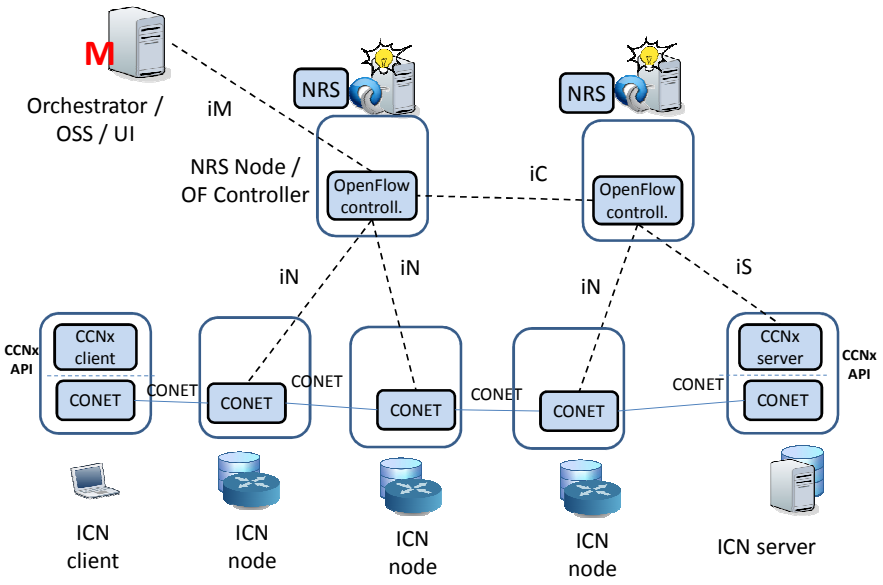


**Figure 7- References Interfaces**

In Figure 7 we assume that a common CONET protocol is used throughout the ICN network ( and across different domains) and that the ICN node can exploit the information carried within the CONET protocol to perform the required operations. On the other hand, within an OpenFlow based domain, we may want to add further information that can be processed by OpenFlow capable equipment within that specific domain. This information needs to be removed outside of the domain. In Figure 8 we have added "InterWorking Elements" (IWE) within ICN "ingress/egress" Nodes. We refer to this process as "Tagging" and we denote with iN+T an extension of the iN interface that also covers this functionality.

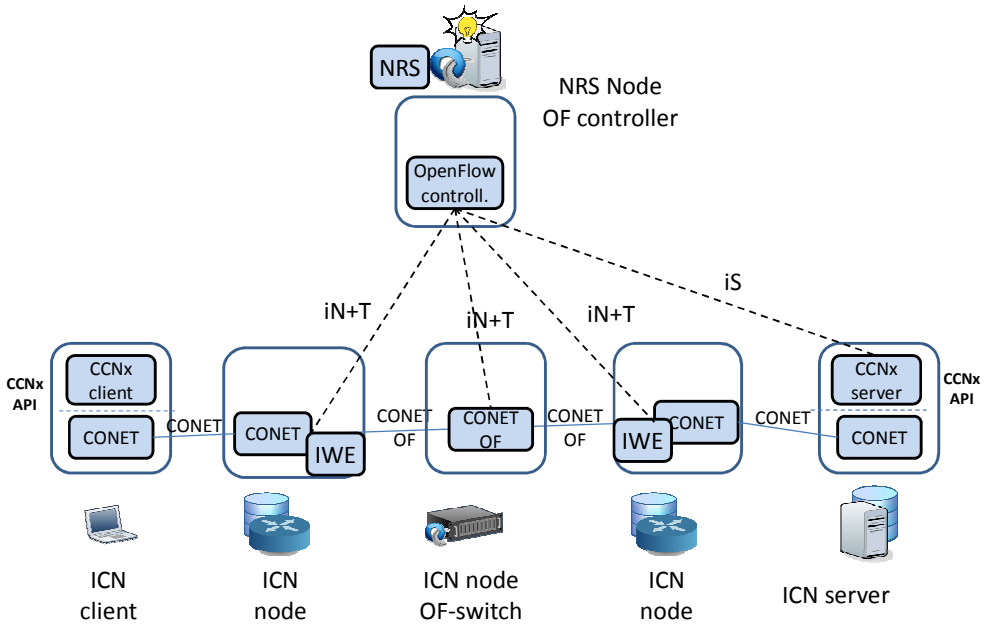


**Figure 8- Reference Interfaces for intra-domain specific SDN functionalities**

 

# 4 CONET packet format

Figure 9 provides a simplified representation of the structure and content of interest CIUs and data CIUs. Interest and data CIUs are transported using CONET Carrier Packets. Therefore, the data CIUs, which correspond to content "chunks" of arbitrary size, will need to be segmented into a set of Carrier Packets. Interest CIUs will be one-to-one mapped into Carrier Packets.

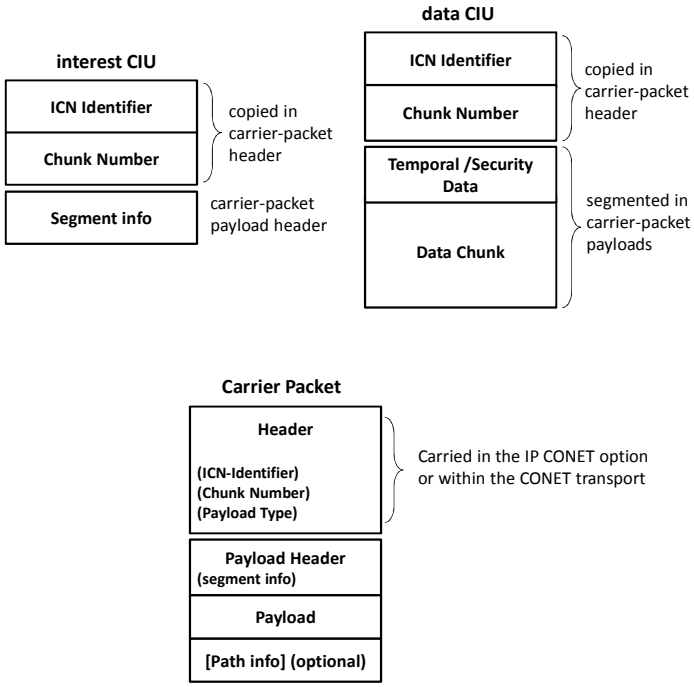


**Figure 9- More on CONET CIUs and CPs**

We have defined and experimented with different ways of transporting name-based information in extended IP packets. One proposal is to introduce CONET IP options into standard IPv4 and IPv6 headers. The second proposal is to include them in "transport layer" protocol headers: particularly within the headers used by the proposed CONET transport protocol.

The first proposal (#1 in Figure 10) is "cleaner" from the point of view of protocol layering. Considering that we are extending the networking layer with content oriented functionality, it seems reasonable to extend the headers of the IP protocol. In [8] we investigated (with practical experiments on PlanetLab) how an unrecognized option is handled by current routers in the Internet. The result was that, in the main, it was possible to add unrecognized options and achieve end-to-end connectivity. The second proposal (#2 in Figure 10) calls for a cross-layer processing of packets in ICN-aware nodes. We assume that Border Routers are able to perform line-speed operations also while looking at higher layer information. This is in line with current "middle-box" operations (like NAT and Firewalls). It is also fully aligned with the SDN/OpenFlow architecture, in which flow table entries can match on protocol headers from layer 2 up to layer 4 in a cross-layer fashion. The third packet format (#3) is a temporary solution that has been used in the experiment with OpenFlow v1.0 equipment.

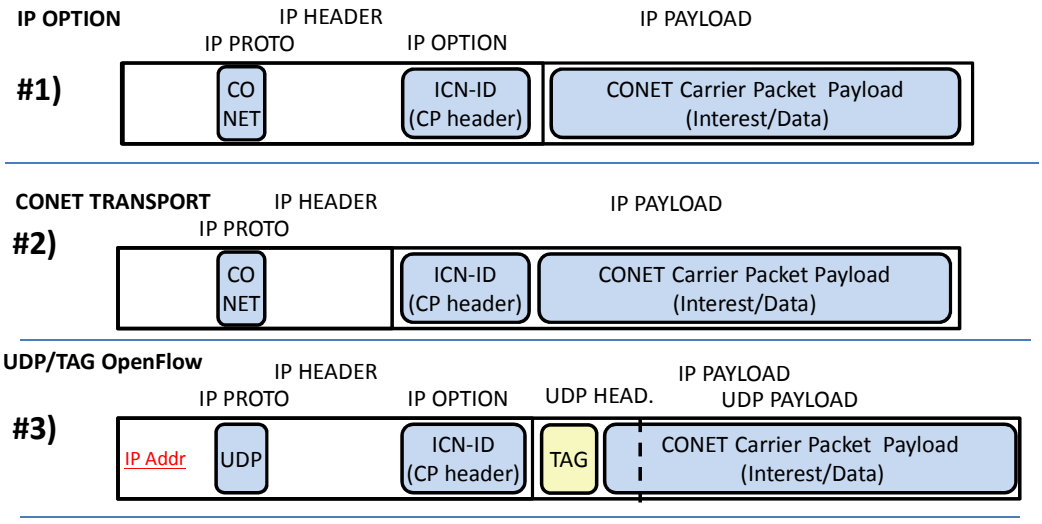


**Figure 10- Different choices for CONET packet format**

We have detailed the proposed IPv4 and IPv6 CONET options in an Internet draft ([12]). The CONET IP Options have the same content for both IPv4 and IPv6. In IPv6, the CONET Option would be transported in the Hop-by-Hop Options header, which is to be analyzed by all routers in the path. The CONET IP Options include the content name, called ICN-ID, which can be of variable size or fixed size, according to the selected naming approach. The CONET IP Options include a "Diffserv and Type" field. This one-byte length field is used to differentiate the quality of service operations that can apply to the content. It also identifies the content type. If the second packet format is used (#2 in Figure 10), the previously described information is transported at the beginning of the IP payload.

The CONET IP Options (see Figure 11) have the same content for IPv4 and IPv6, with an initial byte difference for the coding of option rules. In IPv6, the CONET Option will be transported in the Hop-by-Hop Options header, which is analyzed by all routers in the path. The CONET IP Options include the content name, called ICN-ID, which can be of variable size or fixed size according to the selected naming approach. A namespace ID allows the support of different types of naming schemas. A Chunk Sequence Number (CSN) of variable length can be optionally present. This allows the chunk number information to be read by the ICN layer in a "standard" way. Alternatively, we can define a naming schema that includes the Chunk number, so that this information is transported in the ICN-ID.

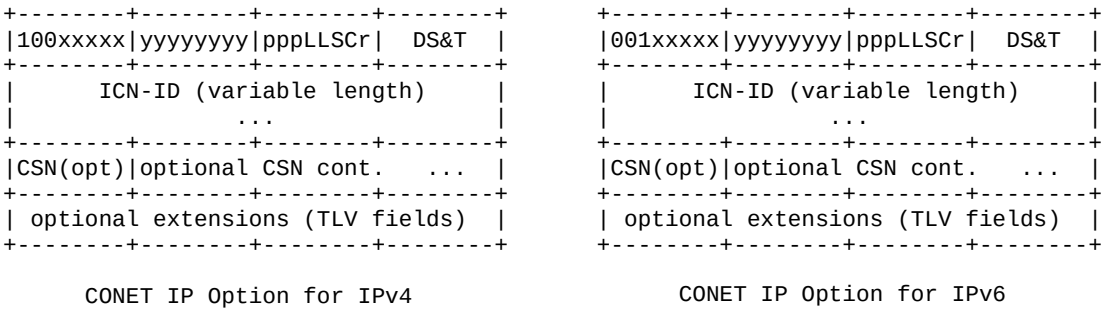


**Figure 11- CONET IP Options for IPv4 and IPv6**

 

## 4.1 Packet matching extensions

The enhanced OpenFlow protocol introduced in section 3 is built using extensions to the matching capability of OpenFlow based ICN nodes. The OpenFlow 1.2 specification has introduced flexible matching for new fields and protocol types. These can be successfully parsed and matched with proper flow table entries. Albeit, this does not enable OpenFlow 1.2 (or the current 1.3 version) to parse a generic unknown field. Rather, it grants the possibility to extend the OpenFlow protocol, in a modular way, such that it is aware of new header fields.

Furthermore, we analyze the extensions required to match the packet formats above described (Figure 10).

Let us start from packet format #1, an OpenFlow switch needs to match ICN fields inserted as IPv4 or IPv6 options header fields. This is currently not possible in OpenFlow 1.3 specifications.

In order to facilitate this functionality, we would need to add new Flow Match Fields, such as "oxm_ofb_match_fields" (see section A.2.3.7 in [18]):

OFPXMT_OFB_ IPV4_OPTION = xx, /* IPv4 CONET option. */

OFPXMT_OFB_ IPV6_OPTION = yy, /* IPv6 CONET option. */

The codes *xx* and *yy* would be subject to standardization. As a set of IP Options can be present in an IP header, the matching rule will specify a value for the CONET IP option. Therefore, the match is verified if there is one IP option in the IP header of type "CONET IP option".

The Match fields details (as reported in Table 11 of [18]) are:

| Field | OFPXMT_OFB_IPV4_OPTION |
|---|---|
| Bits | 8 |
| Mask | No |
| Pre-requisite | ETH TYPE=0x0800 |
| Description | Allows to match a generic IP Option |

Then we need to match the ICN ID field within the CONET IP option. This goes beyond the current OpenFlow match capabilities, as the ICN-ID field is a variable length field. Its length can be in the order of tens of bytes, for example we can (rather arbitrarily) fix a maximum length of 128 bytes. The capability to match a variable length field of such dimension could be problematic to implement at hardware level.

The new match field to be introduced is OFPXMT_OFB_IP_OPTION_ICN_ID, the match field details are:

| Field | OFPXMT_OFB_IP_OPTION_ICN_ID |
|---|---|
| Bits | Variable, up to 128 bytes. |
| Mask | No |
| Pre-requisite | OFPXMT_OFB_IPV4_OPTION = xx /* IPv4 CONET option. */ |
| Description | Allows to match the ICN-ID contained in the IPv4 CONET option |

For this type of match field, the OpenFlow flexible matching would need to be enhanced. Current capabilities only allow for a fixed length match field. In the OXM (OpenFlow Extensible Match) TLV's header (section A.2.3.2 in [18]), oxm_length is a constant for any given match field (described by oxm_class and oxm_field). In our case, the oxm_length is used to actually reflect the length of the field to be matched.

Let us now move to packet format #2. An OpenFlow switch needs to match ICN fields inserted in the transport layer header of the newly defined CONET protocol. Again, we need to extend OpenFlow 1.3 specifications.

For matching IP packets using CONET as a transport protocol, we rely on the existing match field in the "Protocol" byte of the IP header.

OFPXMT_OFB_ IP_PROTO = CONET_PROTO_CODE /* CONET transport protocol. */

We add a new Flow Match Field, such as "oxm_ofb_match_fields" (see section A.2.3.7 in [18]):

OFPXMT_OFB_CONET_ICN_ID

| Field | OFPXMT_OFB_CONET_ICN_ID |
|---|---|
| Bits | Variable, up to 128 bytes. |
| Mask | No |
| Pre-requisite | OFPXMT_OFB_ IP_PROTO = CONET_PROTO_CODE /* CONET transport protocol. */ |
| Description | Allows to match the ICN-ID contained in the CONET protocol header |

Notably in this case we have the problem of matching the variable length ICN-ID and we need to assume that the oxm_length is used to actually reflect the length of the field to be matched.

Finally, we consider packet format #3. In this case we do not need to enhance the flow matching capabilities of OpenFlow. We can rely on OpenFlow 1.0 to match the UDP source and destination ports that carry the "TAG".

 

# 5 Design of OpenFlow protocol extensions

The current approach in the OpenFlow protocol is to have binary encoded messages. Each message is assigned a code, listed in the ofp_type enumeration (section A.1 of [18]). A code is reserved for "Experimenter" messages, and can be used for the experiment/design phase. In the long term, adding new messages to the OpenFlow protocol is possible through the proposition of new message codes to be added to the ofp_type enumeration. We think that this extension mechanism can be simplified and propose to introduce a new "generic" message in the OpenFlow protocol, which can convey operations and parameters encoded with a JSON syntax. In this way we can encode new extension messages using a JSON syntax within standard binary OpenFlow messages. This leads to an interface that can be more readily extended.

In case further implementations want to favor message length rather than readability, an equivalent binary format can be defined for each extension message.

In our experiment, we implemented this using the "Experimenter" message. As a more stable solution, we propose the addition of a similar symmetric OpenFlow message entitled: "ExtensionMessage".

The specific messages that we propose to support the CONET/ICN operations are described hereafter.

### 5.1.1 Cached content notifications

These messages are sent from a node that has cached content, towards the controller. The address of the cache node is implicitly derived from the OpenFlow connection over which the message is sent (see the "Connection setup" section).

type: "CACHE-INFO"

command: "stored" | "refreshed" | "deleted"

fields: "tag", "icnID", "csn", "dataPath", "destIpv4Addr"

notes:

"tag" is the tag that can be associated to the content;

"icnID" and "csn" is the ICN-ID and CSN (chunk sequence number) of the content;

"dataPath" is the data-path of the switch for this operation;

"destIpv4Addr" is the next-hop IPv4 address for this operation; if not specified, the tag has been considered valid for all known next-hop nodes (or icn-servers).

The command field specifies if the content has been stored, or refreshed or deleted. Default value for the command is "stored".

Example message

```
{“type”:“CACHE-INFO”,“command”:“stored”,“tag”:“869728095”,“icnID”:“/example/1k3”,“csn”:“0"}
```

### 5.1.2 *Content publication*

A Serving Node sends a REGISTER message to advertise that is able to provide content under the specified name-prefix. Similar messages can also be propagated among IC-ASes over the iC interfaces (note that the iC interface is out of the scope for this document).

type: “REGISTER”

fields: “contentName”, “address”, “flag”, “distance”

notes:

“contentName” is the name-prefix served by the publisher

“address” is the publisher's IP address

“origin” indicates the origin of the message, either “intra” for messages coming from within the AS or “inter” for inter-autonomous system messages

“distance” indicates the hop-count in term of traversed ASes. This would be zero for register message received from a Serving node of the same IC-AS. It takes the form of a BGP's AS path when publication messages have been propagated between different IC-ASes.

Example message:

```
{“type”:“REGISTER”,“contentName”:“/avg/",“address”:“132.227.62.121”,“origin”: “intra”,
“distance”:“0”}
```

### 5.1.3 *Name to next hop messages*

**Request message**

A node issues a “nexthop request” message each time it does not know the next-hop towards the content

type: “NEXTHOP-REQ”

fields: “contentName”, “address”

notes:

“contentName” is the prefix-name to be resolved into a “next hop”

“address” is the requester's IP address

 

Example message:

```
{“type”: “NEXTHOP-REQ”, “contentName”:“/kakaku.com/”,“address”: “192.42.43.23”}
```

**Response (or Indication)**

A controller provides the mapping of a name (or a name prefix) into a next hop or into a set of possible next hops, assuming in this last case that the ICN node will take care of deciding which next-hop is better suited. The controller can also provide a list of such mappings. It can be sent in a response to a request. Alternatively, it can be sent proactively by the controller, towards a node.

type: “NAME-TO-NEXTHOP”

commands: “set” | “remove”

fields: “prefixName”, “distance”, “nexthops”, “mapArray”

notes:

“prefixName” is the prefix-name mapped into the nexthop.

“distance”: it contains the distance (or the AS-path, as mentioned above).

“nexthops”: list of next-hop values (“nexthop”), optionally associated with an origin scope (“origin”). The origin scope restricts the scope of IP addresses of ICN clients that can use the provided next-hop.

“mapArray” is used when an array of name to next hops mappings is sent.

The command field specifies if the mapping has to be set or removed. Default value for the command is “set”, therefore the command can be omitted if the message is used to set a mapping. The set command is also used to modify an existing entry.

Example message 1:

```
{“type”:“ NAME-TO-NEXTHOP”,
“prefixName”:“/kakaku.com/”, “distance”:“3”,
“nexthops”: [{“nexthop”:“83.230.127.122”}]
}
```

Example message 2:

```
{“type”:“ NAME-TO-NEXTHOP”,
“prefixName”:“/kakaku.com/”, “distance”:“3”,
“nexthops”: [{“origin”:“132.227.62.121/32”, “nexthop”:“83.230.127.122”},
{“origin”:“192.42.43.23/32”, “nexthop”:“83.230.127.122”},
{“origin”:“83.230.127.122/32”, “nexthop”:“130.37.193.143”},
{“origin”:“139.165.21.211/32”, “nexthop”:“192.42.43.23”}]
}
```

Example message 3:

```
{“type”:“ NAME-TO-NEXTHOP”, “mapArray”: [
{“prefixName”:“/kakaku.com/”, “distance”:“3”,
“nexthops”: [{“origin”:“132.227.62.121/32”, “nexthop”:“83.230.127.122”}]
},
{“prefixName”:“/kakaku2.com/”, “distance”:“3”,
“nexthops”: [{“origin”:“132.227.62.121/32”, “nexthop”:“83.230.127.122”}]
}]
}
```

In the example message 1, the next hop mapping for a name is given, offering only one next hop (no origin scope is given). In the example message 2, the next hop mapping for a name is given, offering four next hops, associated with their origin scope. In the example message 3 the next hop mappings for two names are given (the “mapArray” field is used), offering one next hop for each name.

### 5.1.4 *Connection setup*

This message is used by a Cache node to setup a connection towards a controller. With this message the Cache node sends its identification information to the controller.

type: “CACHE-CONNECT”

optional fields: “macAddr”, “ipv4Addr”, “dataPath”

notes:

“macAddr” and “ipv4Addr” are the mac and IPv4 addresses of the cache-server.

“dataPath” is the data-path of the switch to which the cache-server should be bound with (optional)

Example message:

```
{“type”:“CACHE-CONNECT”, “macAddr”: “01:02:03:04:05:06”, “ipv4Addr”: “132.227.62.121”}
```

### 5.1.5 *Name-to-tag mapping*

**Request message**

A node issues a “name-to-tag mapping” request in order to ask for the mapping of a content item’s name into a tag.

type: “TAG-REQUEST”

fields: “contentName”

notes:

“contentName” is the content name to be mapped into a tag

“address” is the requester's IP address

 

Example message:

```
{“type”: “TAG-REQUEST”, “contentName”:“/kakaku.com/myContent”}
```

**Response (or Indication)**

A controller provides the mapping of a name into a tag (or a list of names into tags). It can be sent in response to a request. Alternately, it can be sent proactively by the controller, towards a node.

type: “NAME-TO-TAG”

fields: “contentName”, “tag”, “mapArray”

notes:

“contentName” is the content name that is mapped into a tag.

“tag”: the tag for the content name.

“mapArray” is used when an array of name to tag mappings is sent.

Example message 1:

```
{“type”:“NAME-TO-TAG”, “contentName”:“/kakaku.com/myContent”, “tag”:” “869728095”}
```

Example message 2:

```
{“type”:“NAME-TO-TAG”, “mapArray”:[{“contentName”:“/kakaku.com/myContent”, “tag”:” “869728095”},
{“contentName”:“/kakaku.com/myContent2”, “tag”:” “869727890”}]}
```

In the example message 1, the name to tag mapping for a name is given. In the example message 2 the name to tag mappings for two names are given (the “mapArray” field is used).

### 5.1.6 *Key distribution messages*

The ICN controller configures the ICN nodes with the public keys of the contents. This operation can be performed by the controller by sending KEY-MOD messages. A KEY-MOD message specifies the operation to be applied (“set” or “remove”), the key owner (“contentName”), key validity (“issueDate” and “expirationDate”) and the key value (key).

type: “KEY-MOD”

commands: “set” | “remove”

fields: “contentName”, “command”, “key”, “issueDate”, “expirationDate”

notes:

the command specifies the operation that has to be executed with this content-key pair; possible values are “set” and “remove”

“contentName”: is the content name mapped to the given key.

“key”: contains the key value, encoded in base-64.

“issueDate”: is the starting date of the key validity (not valid before of this date).

“expirationDate”: is the ending date of the key validity (not valid after of this date).

Example message:

```
{“type”: “KEY-MOD”, “command”:”set”, “contentName”:“/kakaku.com/myContent”,
“key”:”afcx567vc9v9qpoi5kj46ks2n6oen4655623jkjn23yr6..97hk==”}
```

 

# 6 Experimentation in the OFELIA testbed

We consider a scenario in which an OpenFlow based network (i.e. the OFELIA testbed) is a part of a larger ICN network (Figure 12). Within this OpenFlow based network, the delivery of content will exploit the specific features of this domain. Whilst, outside the OpenFlow network, non-OpenFlow ICN processing will be used. A node at the edge of the OpenFlow network will be denoted as "Edge Node". Such a scenario is generalized and includes the case in which the OpenFlow domain corresponds to the entire network.

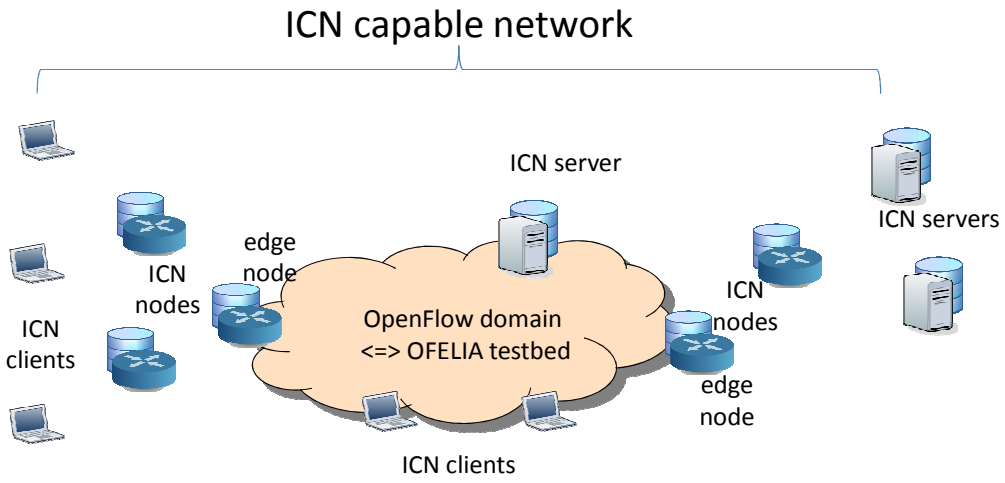


**Figure 12- An ICN network including an OpenFlow domain**

The compatibility between the OF domain and the external world is assured in the Edge node by an Inter Working Element (IWE) which translates regular ICN operations to OpenFlow based ICN operations and vice-versa. This is represented in Figure 13, which also provides further details of the operations in the OpenFlow domain.

Our main assumption is the use of OpenFlow 1.0 switching equipment (i.e. off-the-shelf switches that cannot be enhanced). As we want to add in-network caching in this configuration, we utilized external "Cache Servers" that are able to store and return content upon request. In the future, the caching functionality should be integrated into the switch. This would facilitate the storage of content without needing an external storage server. Therefore we use OpenFlow v1.0 for the communication between our controller and the OpenFlow v1.0 switching equipment in the OFELIA testbed. On the other hand, the communication between our controller and the external "Cache Server" does not need to be OpenFlow v1.0 compliant.

 

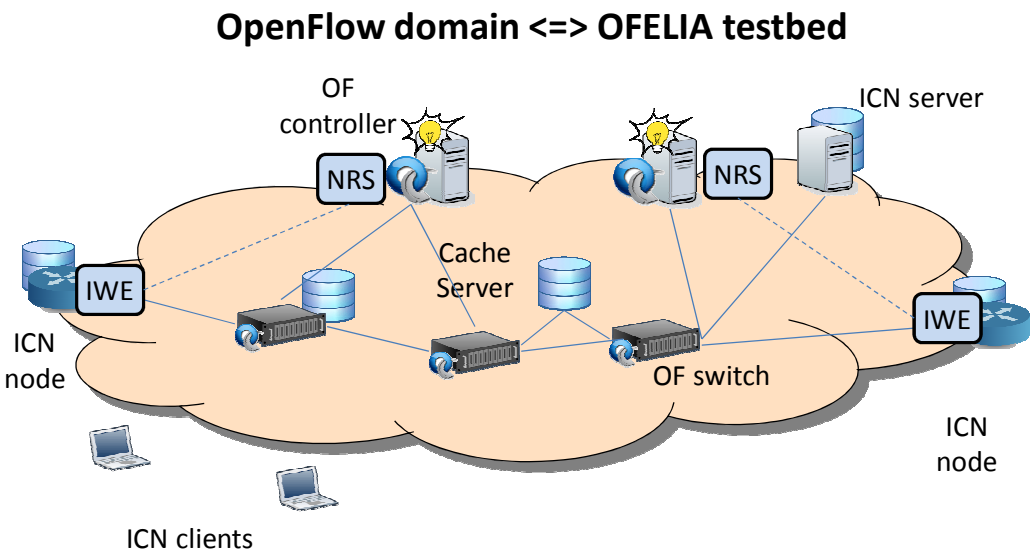


**Figure 13- Details of the OpenFlow 1.0 solution to support ICN**

Let us now describe the proposed operations. An Interest packet is received by an edge node (sitting at the border of the OF domain) and the content has not yet been cached by any node in the OF domain. If the content requested by the Interest has originated from an ICN server internal to the OF domain, the Interest packet will be forwarded to the ICN server in the OF domain. If the content has originated from a server outside of the ICN domain, the interest packet will be forwarded towards the outgoing edge node on the path to the target ICN server. If the content has been already cached within an "in-network" Cache server, the Interest packet has to be forwarded towards it.

In order to keep compatibility with existing hardware, we use Cache servers external to the OF switches. When the data packets flow from the Content server towards the receiving host, a traversed OF switch will duplicate the data packets in such a way that one copy is sent towards the destination (i.e. the ICN client that requested it), and the second copy to the Cache Server.

The above-described operations need to be mapped into the capability provided by OpenFlow switches and controllers. In our assumptions, the OpenFlow domain can be a single IP subnet, operating at layer 3. Alternatively, it may be a set of IP subnets, interconnected by OF capable routers. In both cases, the incoming edge node will resolve the ICN name into the IP address of the content server or of the outgoing edge node, using ICN routing information. Then the incoming edge node will send the Interest packet using the IP address. Likewise, the content server or the outgoing edge node will send the content Data packets using the incoming edge node as the destination IP address. In both cases, the interest and data packets will carry the content name (ICN-ID) in the IP option. In order to support an ICN approach, the internal OF nodes should be capable of reading this content name and behave accordingly: for interest packets they should check if the requested content is already in an in-network cache, and for data packets they should cache the content (if not already cached). Using OpenFlow 1.0, this cannot be accomplished using only the content name stored in the IP option. Therefore, we decided to map the content name in a tag that can be processed by OpenFlow 1.0 equipment and to base the ICN operation on such tag. The mapping from content name to tag will be performed by ICN border nodes with the help of a NRS node, which is logically centralized for a single OF domain.

 

The NRS node will ensure that the mapping from name to tag is unequivocal across the OF domain, therefore Interest requests for the same content arriving at different edge nodes will be mapped into the same tag. The tag is then added to the interest and data, but the content name in the IP option is not removed. This means that the outgoing ICN edge node do not have to query an NRS node to reverse the mapping and get the content name from the tag: it can simply strip out the tag and forward the interest. The outgoing ICN edge node will also cache the content-name- to-tag association, so that when the content data packet returns, it can add the tag and forward the tagged packet in the OF domain.

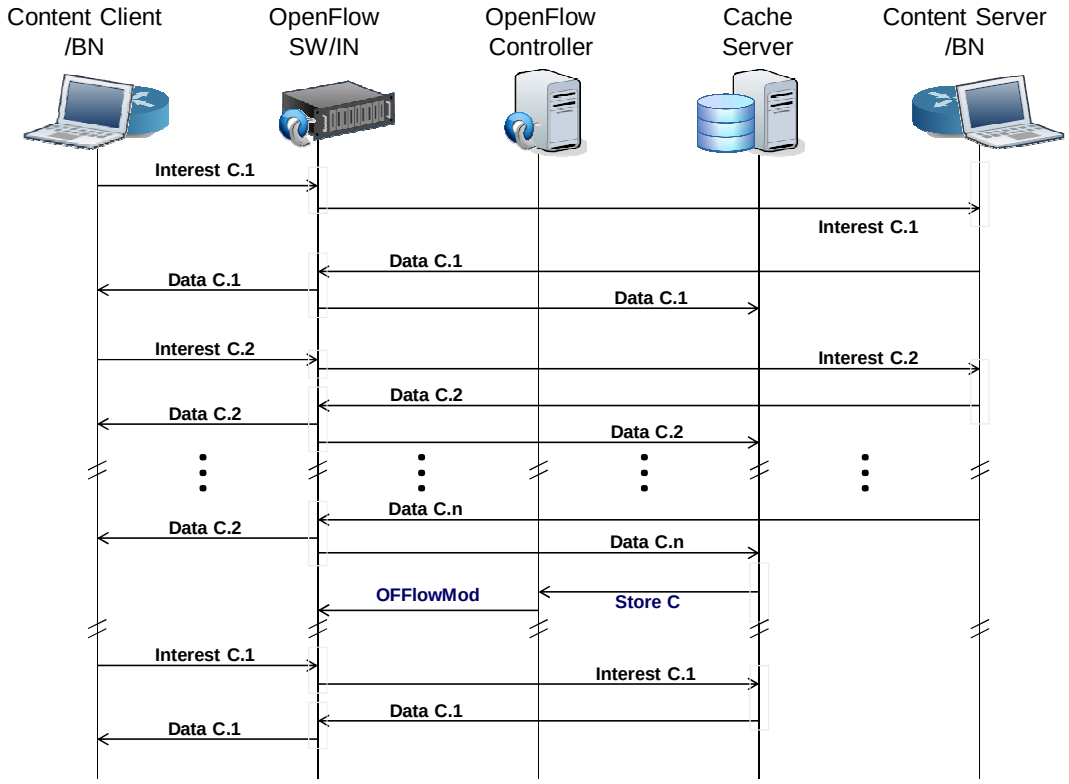


**Figure 14- Sequence of operations**

Figure 14 shows the sequence of operations in the OF domain. Note that the first interest packets for a given chunk of content flow toward the ICN server (or to the Border node if the content server is outside the OF domain). As the data packets return, they are duplicated towards the cache server. When the cache server has cached a full chunk, it notifies the controller, which adds the flow information to the switch. Further requests (i.e. interest packets) for the same chunk of content will be directed to the cache server.

In [15], we discussed different options to carry the tag in the IP packets so that it could be handled by the OF equipment. The solution chosen was to carry the tag as the first four bytes of the IP payload, as for option 3) of Figure 10; note that the first four bytes correspond to the source and destination port header fields in the case of UDP and TCP transport protocol.

The edge node receives IP packets with CONET as its IP payload. It then inserts a 4-byte tag and sets the protocol type to 17. This identifies CONET packets within the OpenFlow-based domain. The IP protocol type 17 corresponds to UDP and has been chosen as to ensure that the CONET-based ICN scenario works correctly with OpenFlow v1.0. This is because OpenFlow v1.0 only allows the inspection of transport protocol header case of UDP and TCP. In the case of protocol 17, the tag can be obtained as the source and destination port of a fictitious UDP header.

In order to distinguish actual UDP packets from CONET packets, we reserve an IP address at ICN nodes for CONET use. In this way, the combination of IP destination addresses and UDP ports (containing the tag) unambiguously identifies content. Moreover, at the OF switch level, we want to distinguish interest packets from the data packets. Therefore, we use different IP addresses for clients (and BNs) that need to receive CONET data and for servers that need to receive CONET interests. An example of this addressing schema is shown in Figure 15, which reports the layout of the deployment testbed. The OF switches identify the packets with IP address 192.168.1.8 as CONET interests and the packets with IP address 192.168.1.23 as CONET data. We note that the use of these special addresses is only due to the limitations of the OpenFlow 1.0 API and existing switches. The clean solution would have been to mark the CONET packet with a specific IP protocol and not with UDP packets. This allows the OF switch to differentiate between interest and packets using information contained in the IP payload. Unfortunately, as already stated, under OpenFlow 1.0 it is only possible to inspect UDP/TCP transport ports and not IP payload content belonging to an arbitrary IP protocol.

The rules for packet forwarding that the controller needs to install in the switch are the following:

RULES

(1) IF IP proto is CONET_PROTO and the destination address belongs to an ICN Client (data receiver)

THEN IF source MAC address is the Cache Server

THEN (1.1) forward the packet (only) to the destination MAC address (i.e. the ICN Client)

ELSE (1.2) forward to both the destination MAC address (i.e. the ICN client) and to the Cache Server

(2) IF IP proto is CONET_PROTO and the destination address belongs to an ICN Server (interest receiver)

THEN IF the tag correspond to a content stored in the Cache Server

THEN (2.1) forward the packet (only) to the Cache Server

ELSE (2.2) forward the packet (only) to the destination address (ICN Server)

In this case, CONET_PROTO is 17.

The first rule (1.1) will avoid the Cache Server receiving duplicate content.

Note that the Cache Server can be associated one-to-one with an OpenFlow switch (and in this case a direct link between the two nodes can be used), or they can be “remotely” located and a one-cache-server-to-many-OF switch relation can be used.

Rules 2.2 are dynamically set and removed according to control message received by the Cache-Server.

Figure 15 shows the first simple configuration implemented over the OFELIA testbed, involving the client, the ICN server, the cache server, the OF Controller realized using Floodlight and two OF switches. This configuration implements option 1) in Figure 5, as the CCNx applications interact using the CCNx API with our CONET module in the ICN client and server. The client and server also implement the IWE in order to send CONET packets, including the tag that is used to map the content name.

 

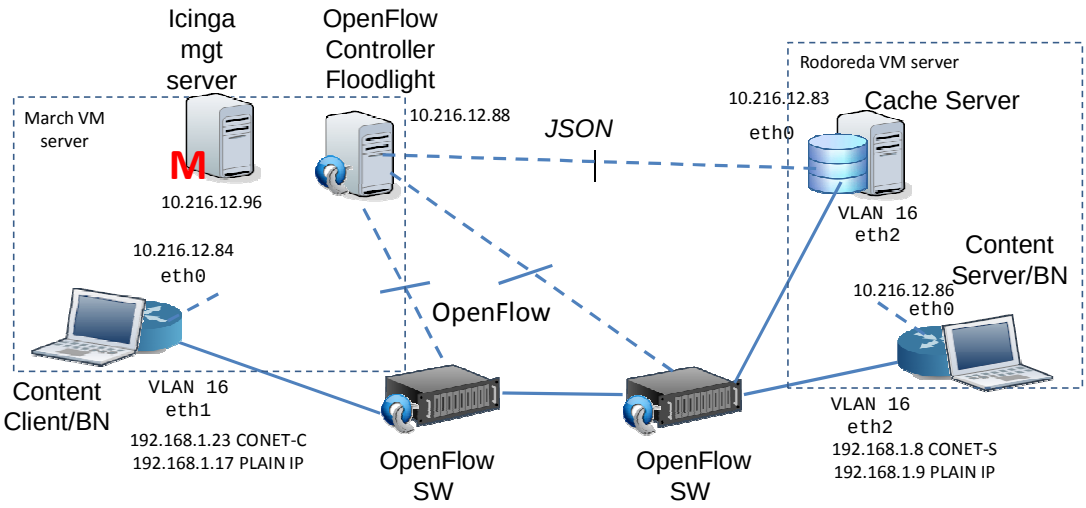


**Figure 15- ICN testbed in OFELIA**

A subset of results from the testbed is reported in Figure 16. It shows the throughput over the network interfaces of the Serving Node ("ICN server"), Cache Server, End-Node ("ICN client") and the number of cached items in the Cache Server. A request generator running on the End-Node is continuously requesting a set of 6 content elements. The Serving Node initially provides the content, as the ICN caching is disabled. When the caching is enabled, the number of cached items becomes 6 and when the client requests such contents, the Cache Server provides them (while the load on the Serving Node becomes zero).

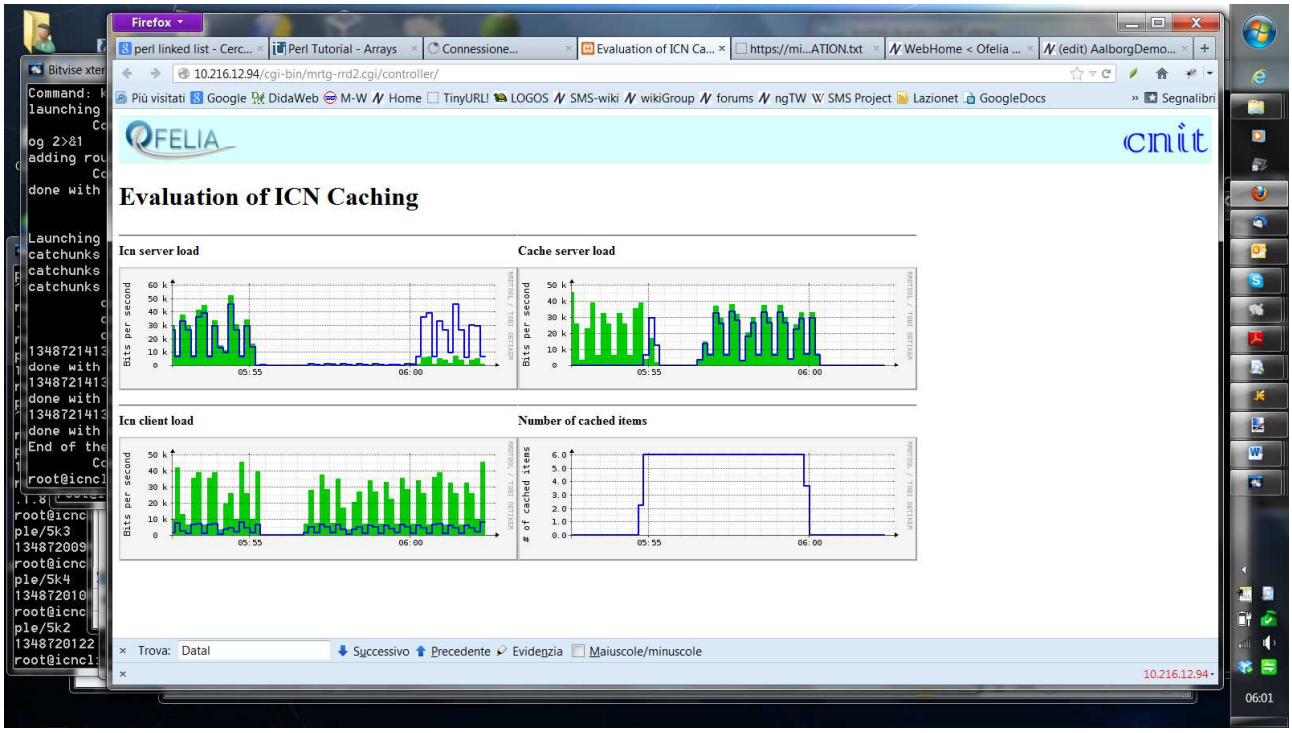


**Figure 16- CONET experiment in OFELIA OpenFlow 1.0 testbed**

## 6.1 The "north-bound" iM interface used in our experiment

The north-bound iM interface is used by the management entities to interact with the controller. It is a REST based HTTP interface, that extends the REST API [32] provided by the Floodlight controller. In our experiment we have used this interface to instruct the controller following the inputs of the web GUI and to gather information from the controller to be reported in the web GUI.

Let us consider the new messages that we have added:

The token **CONTROLLER** replaces the IP address and port of the REST based http interface of the Floodlight controller (default port is 8080).

The following message is used to get the number of Cached items.

```
http://CONTROLLER/wm/core/cache-server/CACHE-SERVER-MAC-ADDRESS/cacheditems/json
```

Example answer:

```
{"CACHE-SERVER-MAC-ADDRESS":{"cacheditems":1}}
```

The following messages are used start/stop the tag based forwarding operations for all switches under the control of a controller, a single switch, or to query the status of either:

```
http://CONTROLLER/wm/core/switch/all/tagbasedfw/start
http://CONTROLLER/wm/core/switch/all/tagbasedfw/stop
http://CONTROLLER/wm/core/switch/all/tagbasedfw/info

http://CONTROLLER/wm/core/switch/SWITCH-DATAPATH/tagbasedfw/start
http://CONTROLLER/wm/core/switch/SWITCH-DATAPATH/tagbasedfw/stop
http://CONTROLLER/wm/core/switch/SWITCH-DATAPATH/tagbasedfw/info
```

Example answer:

```
{"all":{"tagbasedfw":"up"}}
{"all":{"tagbasedfw":"down"}}
```

The following message is used to query the list of cached items in a Cache Server:

```
http://CONTROLLER/wm/core/cache-server/CACHE-SERVER-MAC-ADDRESS/cachedcontents/json
```

We receive back an array of 3-ple like {"icnID":"xxxyyyzzz", "csn":123, "tag":123456789}.

An example of returned JSON message is:

 

{"CACHE-SERVER-MAC-ADDRESS":
[{"icnID":"_example_1k3","csn":2,"tag":869728097},{"icnID":"/example_1k3","csn":1,"tag":869728096},
{"icnID":"/example_1k3","csn":0,"tag":869728095},{"icnID":"/abc/def/ghi/jkl","csn":123,"tag":123456
789}]}

# 7 References


[1] T. Koponen, M. Chawla, B.G. Chun, et al.: “A data-oriented (and beyond) network architecture”, ACM SIGCOMM 2007

[2] V. Jacobson, D. K. Smetters, J. D. Thornton et al., ”Networking named content”, ACM CoNEXT 2009

[3] PURSUIT project, http://www.fp7-pursuit.eu

[4] SAIL project, http://www.sail-project.eu/

[5] CONVERGENCE project, http://www.ict-convergence.eu

[6] COMET project http://www.comet-project.org

[7] CCNx project web site: www.ccnx.org

[8] A. Detti, N. Blefari-Melazzi, S. Salsano, and M. Pomposini. “CONET: A Content-Centric Inter-Networking Architecture”, Proc. of ACM Sigcomm – ICN 2011. August 2011

[9] A. Detti, M. Pomposini, N. Blefari-Melazzi, S. Salsano, “Supporting the Web with an Information Centric Network that Routes by Name”, Elsevier Computer Networks, vol. 56, Issue 17, p. 3705–3722

[10] L. Veltri, G. Morabito, S. Salsano, N. Blefari-Melazzi, A. Detti, “Supporting Information-Centric Functionality in Software Defined Networks”, SDN’12: Workshop on Software Defined Networks, co-located with IEEE ICC, June 10-15 2012, Ottawa, Canada

[11] N. Blefari-Melazzi, A. Detti, G. Mazza, G. Morabito, S. Salsano, L. Veltri, “An OpenFlow-based Testbed for Information Centric Networking”, Future Network & Mobile Summit 2012, 4-6 July 2012, Berlin, Germany.

[12] A. Detti, S. Salsano, N. Blefari-Melazzi, “IPv4 and IPv6 Options to support Information Centric Networking”, Internet Draft, draft-detti-conet-ip-option-04, Work in progress, November 2012

[13] S. Salsano, M. Cancellieri, A.Detti, “Receiver driven transport protocol for CONET ICN”, draft-salsano-rditp-00, Work in progress, November 2011

[14] N. Blefari Melazzi, M. Cancellieri, A. Detti, M. Pomposini, S. Salsano: “The CONET solution for Information Centric Networking”, Tech. report, available at http://netgroup.uniroma2.it/CONET

[15] G. Mazza, G. Morabito, S. Salsano, “Supporting COntent NETworking in OpenFlow”, Tech. report, available at http://netgroup.uniroma2.it/CONET

[16] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, “OpenFlow: Enabling Innovation in Campus Networks”. White paper. March 2008 (available at: http://www.openflow.org).

[17] http://www.openflow.org/

[18] “OpenFlow Switch Specification”, Version 1.3.0 (Wire Protocol 0x04) June 25, 2012, Open Networking Foundation

[19] OFELIA project: http://www.fp7-ofelia.eu

[20] A. Köpsel and H. Woesner, “OFELIA – Pan-European Test Facility for OpenFlow Experimentation”, Lecture Notes in Computer Science. Vol. 6994/2011. 2011

[21] Enterprise GENI (eGENI) project: http://groups.geni.net/geni/wiki/EnterpriseGeni

[22] M. Caesar, D. Caldwell, N. Feamster, J. Rexford, A. Shaikh, and J. van der Merwe, “Design and implementation of a routing control platform”, in NSDI’05, 2005

[23] M. Gritter, D. Cheriton, “An Architecture for Content Routing Support in the Internet”, Proc. Usenix USITS, March 2001

[24] A. Narayanan, Ed., S. Previdi, B. Field, “BGP advertisements for content URIs”, draft-narayanan-icnrg-bgp-uri-00, Work in progress, July 28, 2012

[25] S. Salsano, A. Detti, M. Cancellieri, M. Pomposini, N. Blefari-Melazzi, “Transport-layer issues in Information Centric Networks”, ACM SIGCOMM Workshop on Information-Centric Networking (ICN-2012), Helsinki, Finland, August 2012


 

[26] S. Salsano, M. Cancellieri, A.Detti, “ICTP - Information Centric Transport Protocol for CONET ICN”, draft-salsano-ictp-01, Work in progress, November 2012

[27] V. Jacobson, D. K. Smetters, N. Briggs, M. F. Plass, P. Stewart, J. D. Thornton, R. Braynard, “VoCCN: voice over content-centric networks”, Proceedings of the 2009 Workshop on Re-architecting the Internet (ReArch 2009).

[28] Diego Perino, Matteo Varvello, "A reality check for content centric networking", ACM SIGCOMM Workshop on Information Centric Networking (ICN), Toronto, Canada, August 2011

[29] B. Pfaff, et al., “OpenFlow Specification”, Version 1.1, February 28, 2011, http://www.openflow.org/documents/openflow-spec-v1.1.0.pdf

[30] K Katsaros, G. Xylomenos, G. C. Polyzos: “MultiCache: An overlay architecture for information-centric networking”, Computer Networks, Elsevier, Volume 55, Issue 4, 10 March 2011, Pages 936-947

[31] D. Smetters, V. Jacobson: “Securing Network Content”, PARC technical report, October 2009

[32]“Floodlight REST API”, http://www.openflowhub.org/display/floodlightcontroller/Floodlight+REST+API