



OFELIA

ICT-258365

Deliverable 9.2

EXOTIC evaluation plan and methodology

Editor:	<i>Stefano Salsano (CNIT/University of Rome Tor Vergata)</i>
Work Package (leader)	WP9 (Stefano Salsano, <i>CNIT/University of Rome Tor Vergata</i>)
Deliverable nature:	Report (R)
Dissemination level: (Confidentiality)	Public (PU)
Contractual delivery date:	31/12/12
Actual delivery date:	22/03/2013
Version:	1.0
Total number of pages:	19
Keywords:	OFELIA, FP7, Information Centric Networking, OpenFlow

Disclaimer

This document contains material, which is the copyright of certain OFELIA consortium parties, and may not be reproduced or copied without permission.

In case of Public (PU):

All OFELIA consortium parties have agreed to full publication of this document.

In case of Restricted to Programme (PP):

All OFELIA consortium parties have agreed to make this document available on request to other framework programme participants.

In case of Restricted to Group (RE):

All OFELIA consortium parties have agreed to full publication of this document. However this document is written for being used by <organisation / other project / company etc.> as <a contribution to standardisation / material for consideration in product development etc.>.

In case of Consortium confidential (CO):

The information contained in this document is the proprietary confidential information of the OFELIA consortium and may not be disclosed except in accordance with the consortium agreement.

The commercial use of any information contained in this document may require a license from the proprietor of that information.

Neither the OFELIA consortium as a whole, nor a certain party of the OFELIA consortium warrant that the information contained in this document is capable of use, nor that use of the information is free from risk, and accepts no liability for loss or damage suffered by any person using this information.

Imprint

[Project title]	<i>OpenFlow in Europe – Linking Infrastructure and Applications</i>
[short title]	<i>OFELIA</i>
[Number and title of work package]	<i>WP9 – Support of Content centric networking functionality</i>
[Document title]	<i>D9.2 - EXOTIC evaluation plan and methodology</i>
[Editor]	<i>Stefano Salsano (CNIT/University of Rome Tor Vergata)</i>
[Work package leader]	<i>Stefano Salsano (CNIT/University of Rome Tor Vergata)</i>
[Task leader]	<i>Stefano Salsano (CNIT/University of Rome Tor Vergata)</i>
[PM (estimated)]	<i>1</i>
[PM (consumed)]	<i>1</i>

Copyright notice

© 2010-2013 Participants in project OFELIA

Optionally list of organizations jointly holding the Copyright on this document

Executive summary

OFELIA WP9 is concerned with the design, implementation and validation of a solution for supporting Information Centric Networking using Software Defined Networking. In particular, the solution is being implemented and 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 our previous deliverable (D9.1), we described two proposed solutions: 1) a long term solution based on SDN concepts without taking into account specific limitations of SDN standards and equipment and 2) a short term solution based on OpenFlow 1.0 equipment, taking into account the features that are currently available on the OFELIA testbed.

In this deliverable, we describe the operations used to setup and run the first experiment of our short term solution.

We also describe the goals of a larger scale demonstration, the planned topology, and the performance metrics that will be used to evaluate the results. The results of this demonstration and performance evaluation activities will be included in the next and final deliverable D9.3.

List of authors

Organisation/Company	Authors
CNIT	Stefano Salsano, Luca Veltri, Stefano Brogi, Andrea Araldo
Lancaster Univ.	Matthew Broadbent

Table of Contents

1	Introduction	9
2	Methodology for performance evaluation	11
3	Offloading content server: the first experiment	12
3.1	Experiment results	13
4	More complex topologies and ICN traffic engineering	15
4.1.1	Sub sub section	15
5	References.....	17
6	Appendix: detailed setup instructions for the experiment.....	18

List of figures and/or list of tables

Figure 1- ICN testbed in OFELIA.....	10
Figure 2 – Methodology for evaluation of performance metrics.....	11
Figure 3 – Management web front end GUI.....	12
Figure 3 - ICN client load.....	13
Figure 4 - ICN server load	13
Figure 5 - Cache server load	14
Figure 6 – Number of cached items	14
Figure 7- Topology for ICN traffic engineering experiment	15

Abbreviations

CONET	COntent NETwork
ICN	Information Centric Networks
NDN	Named Data Networking
OF	OpenFlow

1 Introduction

In our deliverable D9.1 [1], we proposed and discussed solutions to support ICN using SDN concepts. We focused on an ICN framework called CONET, which grounds its roots in the CCN/NDN architecture. We addressed the problem in two complementary ways. First we discussed a general and “long term” solution based on SDN concepts. This “long term” solution does not take into account specific limitations of current SDN standards and equipment. Then we focused on a “short term” solution 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. Therefore, we designed the experiment to use only those features that are currently available.

We refer the reader to the deliverable D9.1 [1] for all the details of the solution. We also assume a background knowledge in the principles of ICN and SDN. As a result, we will only give a short introduction, which describes a demo experiment setup. This is shown in Figure 1.

In the experiment we have a “Content Client” making requests to download a set of “contents” using an ICN approach. This means that the ICN application in the Content Client will make requests to download a certain content identified by its “content name”. According to the ICN paradigm, these requests will be served by the origin Content Server, or by a node in the path, which may have cached the content beforehand. The application running on the Content Client is based on the CCNx implementation **Errore. L'origine riferimento non è stata trovata.** In our experiment, the requests are initially served by the Content Server. While the content data is provided back to the Content Client, the OpenFlow controller can instruct the OpenFlow switch to forward a copy of the packets to the Cache Server. The Cache Server will inform the Controller when the “chunks” of the contents becomes available in its cache. The Controller can instruct the OpenFlow Switch to direct further requests for cached content to the Cache Server, rather than the Content Server. For the above procedure to work correctly over an OpenFlow 1.0 testbed, the packets of the different content chunks are identified by a “tag” that is carried in the source and destination port fields of UDP packets. Therefore a mapping functionality from ICN content name to the 4 bytes tag is implemented at the border of the SDN network, in our case in the Content Client itself. Moreover, regular UDP packets and UDP packets carrying ICN requests/ICN content are identified using a different set of IP addresses.

Under the control of the management server, the OpenFlow controller can activate “Tag based forwarding”. This means routing ICN packets based on their tag. Alternatively, it may simply use default layer 2 forwarding based on MAC address learning.

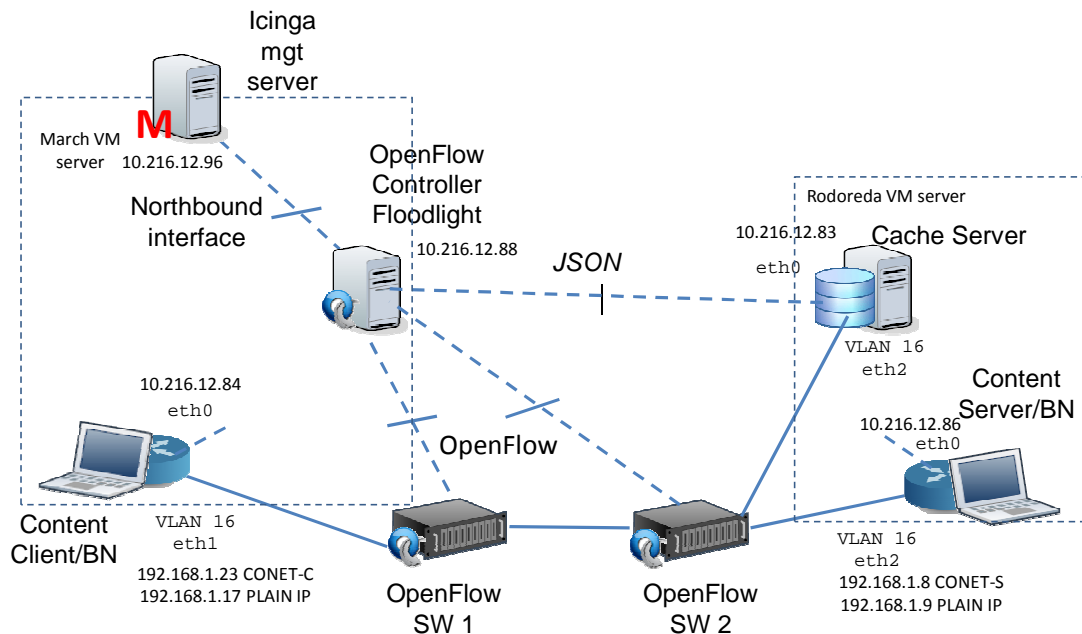


Figure 1- ICN testbed in OFELIA

The management server shown in Figure 1 is in control of the experiment, offers a GUI to the experimenter, and collects the results by monitoring the interfaces.

The main results that can be obtained from this simple experiment are: i) verification of the functionality of the system and ii) measurement of the load shift from the Content Server interface toward the Cache Server interface when the ICN mechanism is activated. Further details are given in section 3.

We aim to evaluate the performance of the proposed solution in more complex scenarios, as described in section 4. The performance metrics of interest are: i) the distribution of load on network links and on content servers / content cache elements; ii) the improvement in user perceived performance, such as download rate and response time, for downloading content.

2 Methodology for performance evaluation

The overall methodology for the evaluation of performance of the proposed solution starts with the generation of synthetic ICN traffic using a set of ICN client and server applications. This synthetic ICN traffic will be processed by the ICN/SDN network. From this, we will gather performance metrics. These are used to evaluate different strategies for controlling resources. This process is graphically represented in Figure 2.

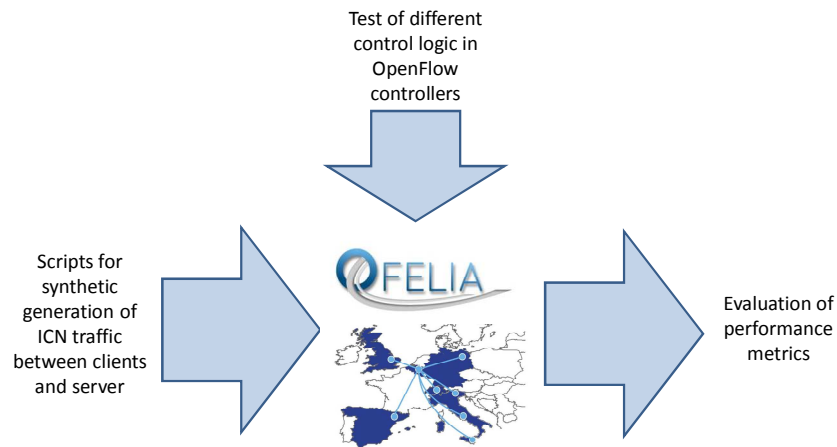


Figure 2 – Methodology for evaluation of performance metrics

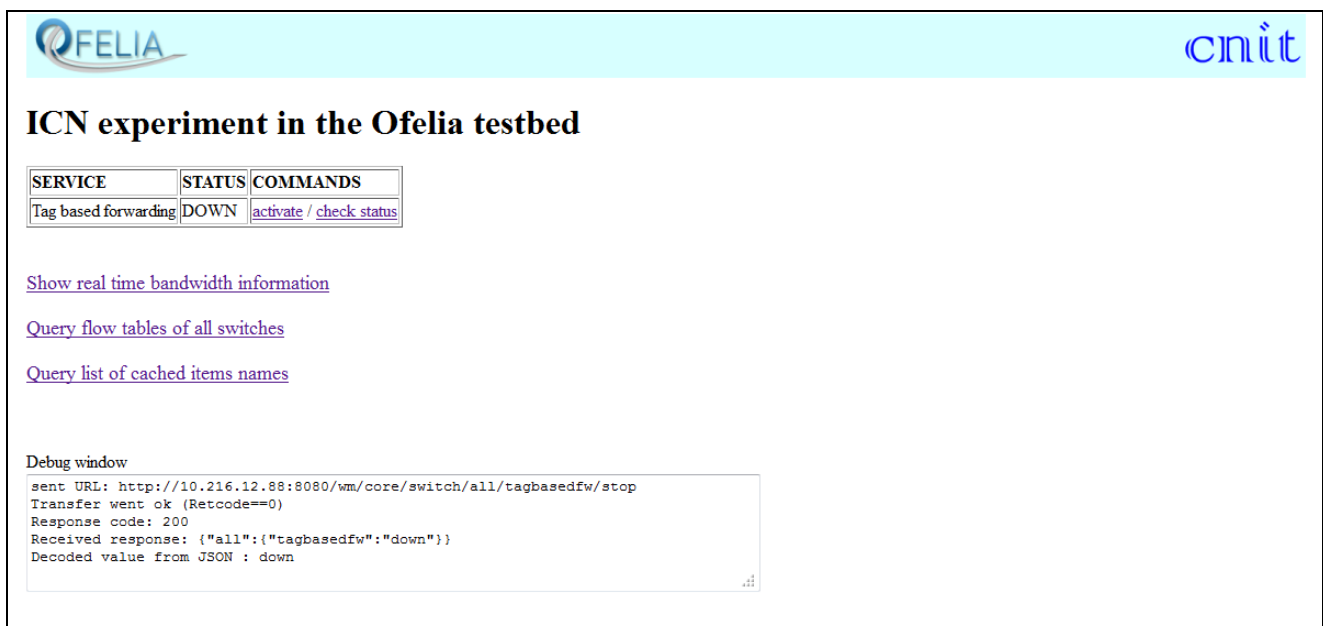
In order to evaluate our OpenFlow controller control, we will take as reference a solution that does not use in-network caching and only uses shortest path routing between ICN client and ICN servers. Then we will introduce the in-network caching capability provided by Cache servers, but without changing the routing between ICN clients and ICN servers. The penultimate step is to influence the routing for traffic engineering purposes. Finally, we will influence the placement of cached objects in cache servers.

3 Offloading content server: the first experiment

The purpose of the first experiment is to show the offloading of traffic from the interface between OpenFlow SW 1 and the Content Server, and on to the interface between OpenFlow SW 1 and the Cache Server. The performance parameter is the traffic rate [kb/s] at IP level measured over the interfaces (in both incoming and outgoing direction). We used SNMP to collect the interface statistics from SNMP agents running on Content Server, Cache Server and Content Client. The SNMP manager runs on the management server and collects the results. The gathered interface statistics are processed using the MRTG tool [2], and are stored using the RRD tool [3]. The MRTG tool provides a convenient mechanism to produce graphic representations that can be presented on the web GUI.

We also measured the number of Cached items stored in the Cache Server, as counted by the Controller. In this case, the information is gathered from the Controller over the REST Northbound API [4] by a process running in the management server. The information is stored with the RDD tool and graphically rendered using MRTG, so that a uniform representation is shown to the experimenter.

From a functional point of view, the web front end GUI allows the retrieval and display of “raw” REST messages exchanged with the controller, as well as more elaborate and structured information. In particular, the switch flow tables can be displayed. The main frame of the GUI is shown in Figure 3. It shows the links that “Tag based forwarding” can be activated upon. It also shows the load on interfaces and a list of cached item names reported from the Cache server to the Controller.



The screenshot shows the management web front end GUI. At the top, there is a header with the OFELIA logo on the left and the cnit logo on the right. Below the header, the main content area is titled "ICN experiment in the Ofelia testbed". Underneath this title, there is a table with three columns: SERVICE, STATUS, and COMMANDS. The table contains one row: "Tag based forwarding", "DOWN", and "activate / check status". Below the table, there are three links: "Show real time bandwidth information", "Query flow tables of all switches", and "Query list of cached items names". At the bottom of the page, there is a "Debug window" showing a REST API call and its response. The debug window content is as follows:

```

Debug window
sent URL: http://10.216.12.88:8080/wm/core/switch/all/tagbasedfw/stop
Transfer went ok (Retcode==0)
Response code: 200
Received response: {"all":{"tagbasedfw":"down"}}
Decoded value from JSON : down

```

Figure 3 – Management web front end GUI

The traffic generators are realized using python scripts running on the CN server and ICN client. A script on the ICN server loads a set of content files in the CCNx repository. A script on the client side first inserts the ICN routing information, i.e. it associates the ICN server IP address to the content names (no dynamic dissemination of ICN routing information is performed in this experiment). Then the script starts making

ICN requests by continuously cycling over the list of contents. The routing in the SDN network (from the ICN client to the ICN server) is modified by the controller.

3.1 Experiment results

The experiment is structured as follows. From the management GUI we make sure that “Tag based forwarding” is disabled. Then we run the ICN traffic generator script that will start making requests for a set of contents. When tag based forwarding is disabled, all requests and data will flow between the ICN client and the ICN server. In the figure below, the green solid bars shows the incoming traffic while the blue line shows the outgoing traffic. Looking at Figure 4, we see that the ICN client load is constant with time, with a lower rate of outgoing traffic due to the ICN requests and a higher rate of incoming traffic due to the ICN content being downloaded.

At a given time, the ICN “tag based forwarding” is activated via the management web GUI. This means that the controller will instruct the switch to duplicate ICN data packets flowing from client to server towards the Cache Server. Therefore we observe in Figure 6 a rise of incoming (blue) traffic. When the Cache Server completes the caching of chunks, it informs the Controller. This is show in Figure 7 by the rise of the number of Cached Items counted by the Controller. When the Controller is notified of a Cached content chunk, it instructs the switch to forward the requests for that chunk to the Cache server. In Figure 6, we start to see that outgoing traffic increases while outgoing traffic on the ICN server interface (Figure 5) decreases (as well as the incoming traffic on the ICN server, as the requests are no longer arriving). When all contents have been cached on the Cache Server, no traffic is flowing on the ICN server interface. Finally, the tag based forwarding is deactivated, and the traffic profile switches back to the initial one, flowing only between ICN client and ICN server.

ICN client load

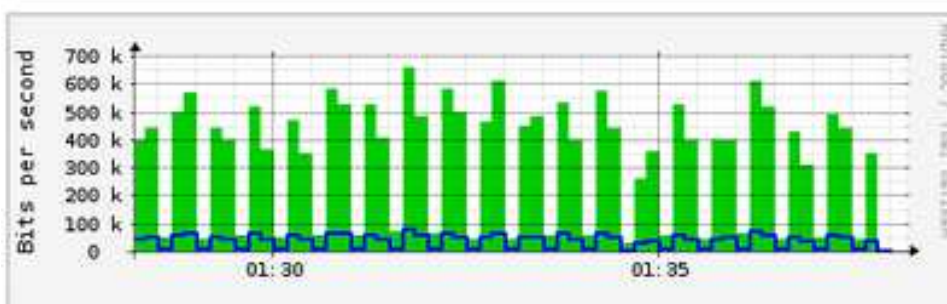


Figure 4 - ICN client load

ICN server load

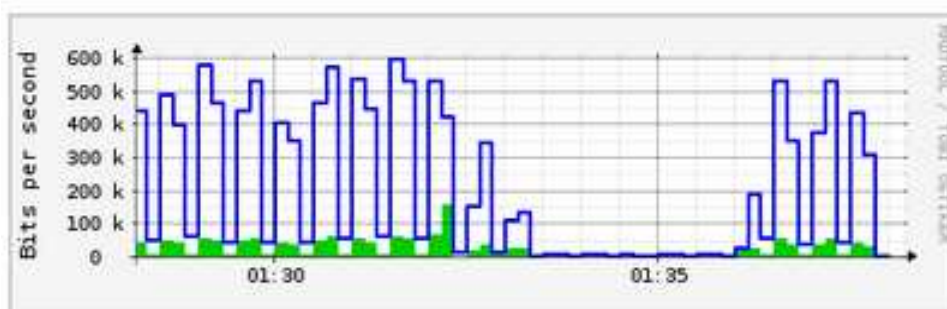


Figure 5 - ICN server load

Cache server load

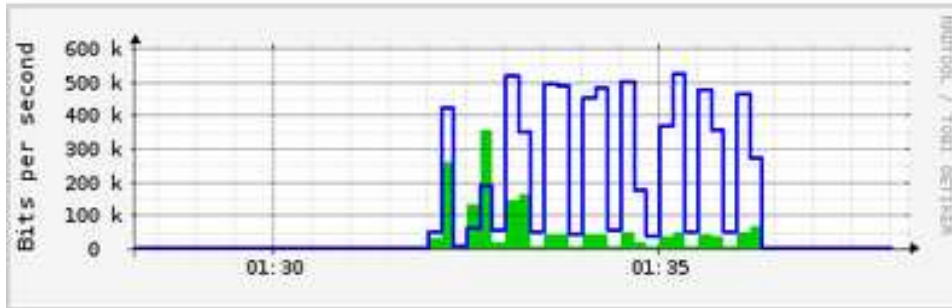


Figure 6 - Cache server load

Number of cached items

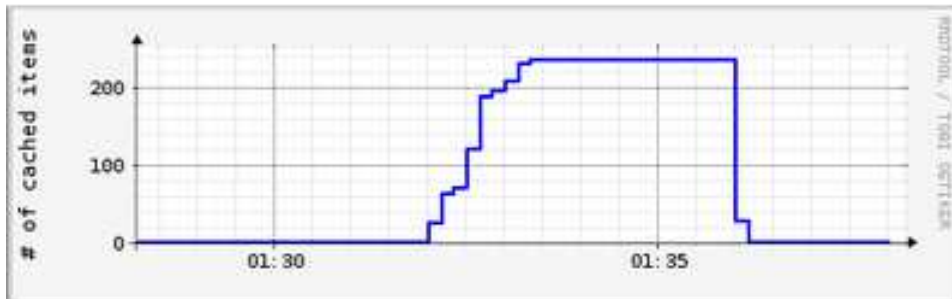


Figure 7 – Number of cached items

4 More complex topologies and ICN traffic engineering

After developing the basic experiment described in section 3, we will develop the system with a more complex topology. This topology will show that the controller can make routing decision based on traffic engineering goals and perform load balancing across cache servers. The target topology for the experiment is shown in Figure 8. This relatively large topology is built using inter-island federation of OpenFlow resources, and is made possible by the recent versions of the OFELIA Control Framework. In Figure 8 we have 5 federated islands: Trento (1), Barcelona (2), Catania (3), Zurich (4) and Berlin (5). The overall topology used in the experiment is made of 21 OpenFlow switches and 10 VM hosting servers, connected by 60 links (2 more switches and 5 links that are not used are marked with a cross in Figure 8. The topology is meshed and there are several redundant paths.

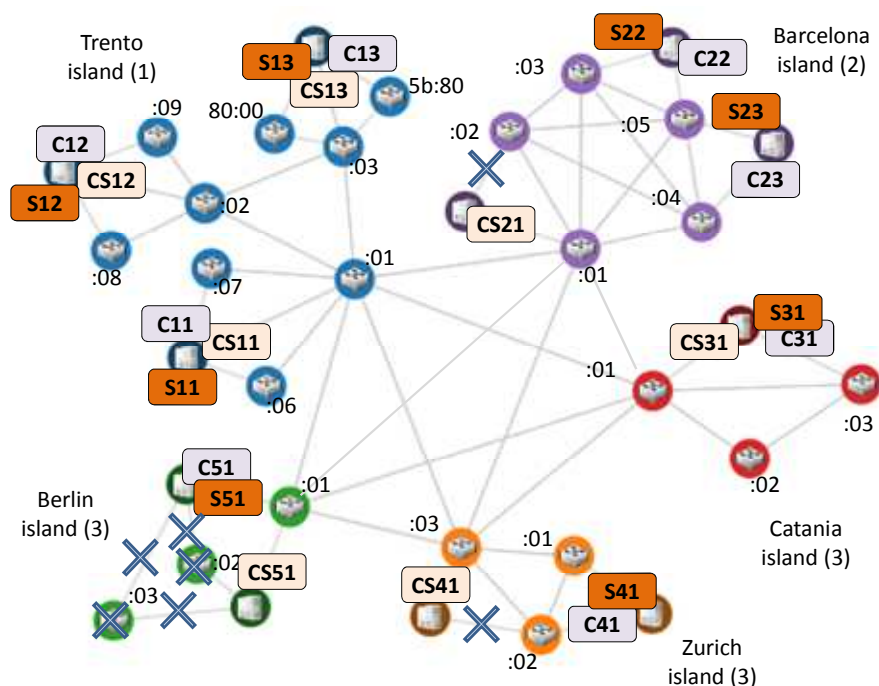


Figure 8- Topology for ICN traffic engineering experiment

On the VM hosting server we will run 8 ICN clients, 8 ICN servers and 6 Cache Servers. These elements are represented respectively with C_{xy} , S_{xy} , CS_{xy} , where x is the island and y an index of the given element within the island. The same VM hosting server can run multiple elements at the same time. The Cache Server elements are always connected to the switches through a separate physical port with respect to Client and Servers. The Clients and Servers are connected through different ports where possible, but in some cases they share the same physical port. In Figure 8 it is possible to see on which port the elements are connected.

4.1 Performance metrics

In order to evaluate the proposed ICN traffic engineering mechanism we will consider: i) the load on the links; ii) the load on the servers (ICN servers and cache servers); iii) the user perceived performance, i.e. download rate/download latency.

5 Summary

In this document we have first provided a short introduction on the ICN /SDN integration solutions proposed in our previous deliverable D9.1. Then we have identified our general methodology for experimentation and performance evaluation. We have provided a description of the first experiments on a simple topology and of their results. Finally we have identified our plans for experimenting with more complex topologies and ICN traffic engineering, defining the performance metrics of our interest. The results of the experiments will be provided in our next deliverable D9.3

6 References

- [1] S. Salsano (editor) “EXOTIC final architecture and design”, Deliverable D9.1, Project FP7 258365 “OFELIA”
- [2] “Tobi Oetiker's MRTG - The Multi Router Traffic Grapher” <http://oss.oetiker.ch/rrdtool/>
- [3] “About RRD” <http://oss.oetiker.ch/rrdtool/>
- [4] “Floodlight REST API”, <http://www.openflowhub.org/display/floodlightcontroller/Floodlight+REST+API>

7 Appendix: detailed setup instructions for the experiment

```
#####
##### Addresses for the demo #####
#####
+-----+
|      host      | control IP | experim. IP | exp. VLAN |
| icn_client     | 10.216.12.84 | 192.168.1.23 | eth2.16   |
| icn_server     | 10.216.12.86 | 192.168.1.8  | eth1.16   |
| new_conet_controller | 10.216.12.88 |              |           |
| management    | 10.216.12.94 |              |           |
| node_one (cache server) | 10.216.12.83 |              |           |
+-----+

#####
##### controller setup #####
#####
- ssh to 10.216.12.88
- cd /ofelia/users/lucaveltri/conetcontroller_bin/
- su
- start.sh

if needed stop.sh, restart.sh are available

check if controller is running
* ps ax | grep java
look at controller output log in in real time
* # tail -f /ofelia/users/lucaveltri/conetcontroller_bin/log/conetcontroller.log

#####
##### management server setup #####
#####

su
check if apache is running
#ps ax | grep apache
if apache is running, stop it
#apachectl -k stop

start xampp
#/opt/lampp/lampp start
check that lampp is running
#ps ax | grep lampp

run icinga
# /opt/icinga/bin/icinga -d /opt/icinga/etc/icinga.cfg
check that icinga is running
#ps aux | grep icinga

check if mrtg is running
ps aux | grep mrtg

in order to kill mrtg
kill -9 `cat /home/mrtg/cfg/mrtg.pid`

in any case, before starting mrtg, check that there are no lock files
#cat /home/mrtg/cfg/mrtg.pid
if present, delete it
#rm /home/mrtg/cfg/mrtg.pid

check also this folder
#ls /var/lock/mrtg
if there is any file, delete it
#rm /var/lock/mrtg/*

run mrtg
#/usr/bin/mrtg --user=nobody --group=nogroup /home/mrtg/cfg/mrtg.cfg --logging
/home/mrtg/logs/mrtg.log
check that mrtg is running
#ps aux | grep mrtg

if mrtg does not start it may go in "out of memory", this is logged in the dmesg
```

```

in this case, try to delete all RRD files like
#rm /opt/lampp/htdocs/mrtg/icnserver_/*.rrd
#rm /opt/lampp/htdocs/mrtg/controller_/*.rrd
#rm /opt/lampp/htdocs/mrtg/icnclient_/*.rrd
#rm /opt/lampp/htdocs/mrtg/cacheserver_/*.rrd

the cached item RRD file needs to be manually created
#/ofelia/users/stefanosalsano/myscripts/myrrd.pl -i 10 -create
/opt/lampp/htdocs/mrtg/controller_/cacheditems.rrd
#chown nobody:nogroup /opt/lampp/htdocs/mrtg/controller_/cacheditems.rrd
#chmod 777 /opt/lampp/htdocs/mrtg/controller_/cacheditems.rrd

once icinga and mrtg have been started, the web GUI of icinga is accessible at
http://10.216.12.94/icinga/

#####
##### cache setup #####
#####
- ssh to 10.216.12.83 as andreaaraldo
or
-su andreaaraldo
after ssh to to 10.216.12.83 as any other user (but not root)

- cd /ofelia/users/matteocancellieri/
- update ccnx code:
  - cd /ofelia/users/matteocancellieri/ccnx062base-cp-raw
  - svn update
  - Open /ofelia/users/matteocancellieri/ccnx062base-cp-raw/include/conet/conet.h and check
that "#define IS_SERVER" and "#define CONET_IFNAME "eth2.16"

- cd /ofelia/users/matteocancellieri/ccnx062base-cp-raw/cache_server
- sudo sh make_all.sh
- launch cache server
  sudo sh go_listener.sh
  (if you want to change some parameters, nano go_listener.sh)

#####
##### icn-server setup #####
#####
- ssh to 10.216.12.86
- su
- cd /ofelia/users/matteocancellieri/ccnx062base-cp-raw/
- svn update
- Open /ofelia/users/matteocancellieri/ccnx062base-cp-raw/csrc/include/conet/conet.h and check that
"#define IS_SERVER" and "#define CONET_IFNAME "eth2.16"
- sh /ofelia/users/matteocancellieri/ccnx062base-cp-raw/mymake
- launch the ccnd daemon
  export CCND_CAP=10
  /ofelia/users/matteocancellieri/ccnx062base-cp-raw/csrc/ccnd/ccnd 0 0

or you can run /ofelia/users/matteocancellieri/goserver

- open another terminal
- cd /ofelia/users/matteocancellieri/
- singleserver_6contenuti

#####
##### icn-client setup #####
#####
- ssh to 10.216.12.84
- su
- cd /ofelia/users/matteocancellieri/ccnx062base-cp-raw/
- svn update
- check if /ofelia/users/matteocancellieri/ccnx062base-cp-raw/include/conet/conet.h is set properly
- sh /ofelia/users/matteocancellieri/ccnx062base-cp-raw/mymake

script for requesting 200 chunks cyclically (it also runs the ccnd daemon)

- cd /ofelia/users/matteocancellieri
- python start_client_loop.py

```