ICT-257422

CHANGE

CHANGE: Enabling Innovation in the Internet Architecture through Flexible Flow-Processing Extensions

Specific Targeted Research Project

FP7 ICT Objective 1.1 – The Network of the Future

D6.3 – Summer Boot Camp Report

Due date of deliverable: September 30, 2013

Actual submission date: September 30, 2013

Start date of project	October 1, 2010
Duration	39 months
Lead contractor for this deliverable	Université catholique de Louvain
Version	1.0, September 10, 2013
Confidentiality status	Public

Abstract

This document contains the report on the bootcamp. The bootcamp was held in Louvain-la-Neuve, Belgium, between 15 and 28 July, 2013. We offered, during the first week, a competitive program with 8 technical presentations about software developed within the project by internationally renowned experts. The second week consisted of hands-on projects around the softwares presented during the first week. The complete program is available at `http://www.change-project.eu/events/bootcamp.html`.

Target Audience

The general public.

Impressum

Full project title	CHANGE: Enabling Innovation in the Internet Architecture through Flexible Flow-Processing Extensions
Title of the workpackage	D6.3 – Summer Boot Camp Report
Editor	Olivier Bonaventure, Université catholique de Louvain
Project Co-ordinator	Adam Kapovits, Eurescom
Technical Manager	Felipe Huici, NEC

This project is co-funded by the European Union through the ICT programme under FP7.

Executive Summary

The bootcamp took place during the third and fourth week of July 2013 in Louvain-La-Neuve, Belgium. The number of attendees was 20.

The objective of the first week of this bootcamp was to bring together the different technologies that have been developed by the participants of the CHANGE project and explains the way they work. Experts presented tutorials for their projects.

The objective of the second week was to implement new features for the projects presented during the first week. The attendees were supported by the original developers of the solution. Results and reports of this second week are presented in section 4.

The website for the bootcamp is available at `http://www.change-project.eu/events/bootcamp.html`. It contains further details about the organization of the bootcamp.

List of Authors

Authors	Benjamin Hesmans and Olivier Bonaventure
Participants	Université catholique de Louvain
Work-package	WP6 – Dissemination, Exploitation and Standardization
Security	PUBLIC (PU)
Nature	R
Version	1.0
Total number of pages	20

Contents

1 Introduction

The goal of the CHANGE project is to introduce the concept of flow-processing platforms instantiated at critical points in the network, realize such platforms using commodity hardware - e.g. x86 servers and commodity switching chipsets -, and introduce basic processing primitives to demonstrate the capabilities of the concept. The ultimate vision is to provide innovative end-to-end services to applications through the combination of multiple communicating flow processing platforms . Thus, conventional traffic flows can be processed at varying degrees of granularity, and application-specific virtual network overlays can be constructed, without impacting other network services or traffic. The aim is to do this within an architectural framework that allows application developers and network operators to reason about the emergent end-to-end behavior. To validate the architecture, we will implement and deploy a set of novel and diverse applications and services.

As CHANGE enters its final phase it organized a Bootcamp as part of its impact creation effort, but also to gain feedback and validate its concepts and solutions in wider circles going beyond the project team.

The bootcamp was composed of two types of activities. The first week was devoted to technical presentations and hands-on tutorials on various networking system software and techniques that have been developed within the CHANGE project. This allowed the participants to acquire a broad overview of the key principles that support these software prototypes. During the second week, the students have developed an extension to one of these software prototypes with support from the software developers.

The rest of this report is structured as follows. Section 2 presents the program of the presentations of the first week. Section 4 lists the reports of the attendees' second week work. Section 3 provides the list of the attendees.

2 First week: Program

The bootcamp was composed of two types of activities. The first week was devoted to technical presentations and hands-on tutorials on various networking system software and techniques that have been developed within the CHANGE project. The second week, students were invited to develop an extension to one of the software prototypes with support from the software developers.

First week

- In-network application processing with CHANGE
  - **Date :** 07/15/2013, 14.00-16.00
  - **Speaker :** Costin Raiciu (Universitatea Politehnica din Bucuresti)
  - **Abstract :**

    The original end-to-end Internet architecture has been long praised for its simplicity and openness to innovation. However, the architecture lacks explicit security support, among other things, pushing operators to deploy a myriad of middleboxes in the network to enhance it. Meanwhile, applications running at end-systems are forced to work around these middleboxes by deploying their own in-network application support (e.g., Skype supernodes, STUN and CDN servers).

    Application support is undoubtedly needed in the Internet but currently it's not being provided in any coherent way. In this paper we propose CHANGE, a distributed operat ing system that allows untrusted endpoints to deploy custom in-network processing by specifying what processing they desire to platforms run by operators. The platforms check whether the requested processing is safe by using a mix of static analysis and runtime checking. A set of strict security rules ensure that flow processing does not harm other users.

    The main programming interface offered by CHANGE is Click: users can create flexible, safe processing configurations by joining predefined or new Click elements. To ensure isolation between users, CHANGE runs Click configurations inside ClickOS, a novel virtualized operating system we have developed. A ClickOS virtual machine boots in tens of milliseconds; a commodity server can run hundreds such machines with very high processing performance.

    We have implemented CHANGE and tested it in the wide-area. To showcase its properties, we have implemented and tested a number of use-cases that are difficult to deploy in today's Internet. Our experience shows that Condatis is fast, flexible and secure.

    The talk will conclude with a handson session on client interface programming and hacking and testing out Change configurations on the distributed testbed

- High speed networking in virtual machines
  - **Date :** 07/15/2013, 16.15-18.15

  - **Speaker :** Luigi Rizzo (Università di Pisa)
  - **Abstract :**

    This presentation will discuss performance issues in the handling of network traffic within a virtual machine, covering the entire path from the physical interface of the host to the device driver in the guest.

    We will show how the entire datapath can be optimized, using some of the techniques developed in netmap, plus some improvements in device emulation, so that we can reach bare-metal performance for socket based applications, and over 5 Mpps for netmap-based clients. Practical examples will be given using our QEMU prototype.

- netmap: a framework for high speed software packet forwarding
  - **Date :** 07/16/2013, 9.00-11.00
  - **Speaker :** Luigi Rizzo (Università di Pisa)
  - **Abstract :**

    This presentation will start discussing performance issues in software packet processing and then show how these have been addressed by recent ”OS bypass” techniques.

    We will then proceed with a detailed discussion of the netmap architecture, describing both the user’s view and the internal implementation.

    Finally, we will show how the netmap API has been used to build a software switch, VALE, which among other things enables developing high speed packet processing applications without the need for physical interfaces.

- Taming the network with tiny middleboxes
  - **Date :** 07/16/2013, 14.00-17.00
  - **Speaker :** Mohamed Ahmed (NEC), Felipe Huici (NEC)
  - **Abstract :** This talk will present ClickOS, a tiny Xen VM running the Click modular router, for building in-network processing appliances (middleboxes). We will discuss the architecture, and performance and design trade-offs in ClickOS and show its suitability in realising software defined middleboxes.

    During the handson part, participants will develop a ping responder. Building upon the advantages of fast booting VMs, participants write code to boot ClickOS VM in response to individual ping messages. This will mainly involve small amount of click configuration and writing packet sniffing, vm creation routines.

- Writing modules for middleboxes using FlowOS

- **Date :** 07/17/2013, 9.00-12.00
- **Speaker :** Mehdi Bezahaf (Lancaster University), Abdul Alim (Lancaster University)
- **Abstract :**

  Middleboxes are heavily used in the Internet to process the network traffic for a specific purpose (Firewall, NAT, Traffic matching..etc). As there is no open standards, these proprietary boxes are expensive and difficult to upgrade. During this workshop, you will be able to use a programmable platform for middleboxes called FlowOS to run on commodity hardware. It provides an elegant programming model for writing flow processing software, which hides the complexities of low-level packet processing, process synchronisation, and inter-process communication. At the end of the bootcamp, you will be able to write down some modules as HTTP traffic filter, where you have to drop all the packets containing certain Internet traffic content, or deactivating for example all the media content in each webpage.

- Checking middlebox processing with SymNet
  - **Date :** 07/18/2013, 9.00-12.00
  - **Speaker :** Matei Popovici (Universitatea Politehnica din Bucuresti) / Radu Stoenescu (Universitatea Politehnica din Bucuresti)
  - **Abstract :** Today's networks deploy many stateful procesing boxes ranging from NATs to firewalls and application optimizers: these boxes operate on flows of packets, rather than individual packets. As more and more middleboxes are deployed, understanding their composition is becoming increasingly difficult. Static checking of network configurations is a promising approach to help understand whether a network is configured properly, but existing tools (such as Anteater and Header Space Analysis) are limited as they only support stateless processing.

    We propose to use symbolic execution—a technique prevalent in compilers—to check network properties more general than basic reachability. The key idea is to track the possible values for specified fields in the packet as it travels through a network. Each middlebox or router will impose constraints on certain fields of the packet via forwarding actions, packet modifications and filtering. The symbolic information makes it possible to answer many questions related to flows rather than packets.

    We have implemented this technique in a tool we call SymNet and evaluated it. We find that symbolic analysis is a cheap and effective way of analyzing complex stateful middlebox behaviors.

    This presentation will conclude with a handson workshop with the SymNet software checking Click configurations and CISCO middlebox configurations, as well as larger networks.

- Detecting middlebox interference with tracebox

- **Date :** 07/18/2013, 14.00-16.00
- **Speaker :** Gregory Detal (Université catholique de Louvain)
- **Abstract :**

  Middleboxes such as firewalls, NAT, proxies, or Deep Packet Inspection play an increasingly important role in various types of IP networks, including enterprise and cellular networks. Recent studies have shed the light on their impact on real traffic and the complexity of managing them. Network operators and researchers have few tools to understand the impact of those boxes on any path. tracebox is an extension to the widely used traceroute tool, that is capable of detecting various types of middlebox interference over almost any path. tracebox sends IP packets containing TCP segments with different TTL values and analyses the packet encapsulated in the returned ICMP message. Recent routers quote, in the ICMP message, the entire IP packet that they received. This enables tracebox to detect any modification performed by upstream middleboxes. Furthermore, tracebox can often pinpoint the network hop where the middlebox interference occurs. We explain the principles behind tracebox and some measurement results obtained. During the handson session, the participants will use tracebox to explore the interference caused by various types of middleboxes.

- Decoupling TCP from IP with Multipath TCP
  - **Date :** 07/19/2013, 9.00-12.00
  - **Speaker :** Olivier Bonaventure (Université catholique de Louvain)
  - **Abstract :**

    The Transmission Control Protocol (TCP) is used by the vast majority of applications to transport data reliably across the Internet. TCP was designed in the 1970s and has slowly evolved since then. Today's networks are multipath: mobile devices have multiple wireless interfaces, datacenters have many redundant paths between servers, and multihoming has become the norm for large server farms. Meanwhile, TCP is essentially a single-path protocol: when a TCP connection is established, the connection is bound to the IP addresses of the two communicating hosts and these cannot change. Multipath TCP (MPTCP) is a major modification to TCP that allows multiple paths to be used simultaneously by a transport connection. We will first dsicuss the various use cases for Multipath TCP including smartphones, datacenters and IPv4/IPv6 coexistence. The design of Multipath TCP has been heavily influenced by the presence of middleboxes in the Internet. We will first discuss the hurdles that they impose on the evolution of transport protocols. Then we will discuss the main design decisions that have shaped Multipath TCP and briefly describe some of the main mechanisms included in the protocol. The talk will end with a discussion of several research challenges that remain open for Multipath TCP.

- Hands-on Multipath TCP
  - **Date :** 07/19/2013, 14.00-16.00
  - **Speaker :** Christoph Paasch (Université catholique de Louvain)
  - **Abstract :**

    Multipath TCP changes many of the assumptions that we have about the transport layer in the Internet. During this handson session, participants will use a Multipath-TCP enabled mininet virtual machine to explore various parameters that influence the performance of Multipath TCP in real world scenarios.

3 Attendees

First name	Name	Institution	country
Emery Kouassi	Assogba	Université Catholique de Louvain	Belgium
Martin	Becke	TDR	
Romain	Capron	Université Catholique de Louvain	Belgium
Matthieu	Coudron	Université Pierre et Marie CURIE	France
Eduard	Creciun	Université Catholique de Louvain	Belgium
Niccolo	De Caro	Vrije Universiteit Brussel	Belgium
Thomas	Dreibholz	Simula Research Lab	Norway
Simone	Ferlin	Simula research	Norway
Carlos	Guimarães	Inst. Telecom. Aveiro	Portugal
Fizza	Hussain	NUST-SEECS	Pakistan
Wang-Bong	Lee	Electronics and Telecommunications Research Institute	Korea
Tamás	Lévai	BME	Hungary
Balázs	Németh	BME	Hungary
Sang-Kil	Park	Electronics and Telecommunications Research Institute	Korea
Vito domenico	Piserchia	dreamlab tech.	Deutschland
Arnaud	Schils	Université Catholique de Louvain	Belgium
Sivasothy	Shanmugalingam	Orange Labs	France
Takis	Tzevelecos	Université de Namur	Belgium
Dario	Valocchi	Università degli Studi di Perugia	Italy
Niels	van Adrichem	TU Delft	Netherlands

4 Second week: Attendees Reports

Niccolo' De Caro (Vrije Universiteit Belgium)

Being a PhD student in Telecommunications Engineering, attending the CHANGE bootcamp was of course of great interest for my studies and my career, since the topics of network virtualization and Software-Defined Networking have gained great momentum in the last few years. Moreover, I had the chance to grab enlightening ideas on this research areas, and to reason over how to apply them to my research field, which concerns Sensor Networks.

During the second week, I have joined the working group related to Multipath TCP. I have been provided with a preliminary implementation of Multipath TCP for Mininet, which is a network emulator using lightweight virtualization to run a collection of hosts, switches, routers and links on a single Linux kernel. The implementation was written in Python language, making use of the Python API of Mininet. After getting acquainted with the basic Multipath TCP concepts and the Mininet tool and Python API, I have set up a set of tests to investigate the behaviour of the protocol in different scenarios, by setting different values of the link quality parameters (i.e. bandwidth, delay and losses). I have also tested the implementation by setting a different network topology.

Carlos Guimarães (Inst. Telecom. Aveir)

The work performed during the second week of the CHANGE Bootcamp had as main purpose to understand how ClickOS could be used to implement middleboxes, by implementing and testing new elements.

Since I was not aware of the capabilities of ClickOS, neither how to program elements for Click Modular Router, during the first day I studied the framework, trying to run and to understand some of the provided examples.

In the following two days a simple monitor element was developed for Click and tested in ClickOS, allowing me to understand the basics of how to implement elements for Click. The monitor element implemented allows the definition of different flows based on several rules, which can be configured dynamically by an external entity. Then, for each incoming packet, the packet is matched with the existing and configured flows and some statistics are stored for each flow (such as, received packets and bytes, packet rate and byte rate). Since the main goal of the development of this element was to understand ClickOS, the flow identification was limited to the IP source and IP destination fields.

On the last two days of the bootcamp, the work was focused on adding support for Content-Centric Networking behavior on ClickOS, by defining new Click elements. Since CCN packets may not rely on the IP protocol to be transported, different mechanisms for forwarding, routing and management of the CCN packets are required. Thus, in an initial step, it was necessary to understand how this mechanisms could be implemented in new Click elements and then organized, taking into account the ClickOS environment, in order to provide the behavior of a CCN router. An initial implementation of the elements was started during

the bootcamp but, due to the complexity of the task, it was impossible to finish it during the time provided. Still, I intend to continue this work outside the CHANGE bootcamp and, when a stable implementation is reached, to release the elements implementations as open-source software.

Takis Tzevelecos (Université de Namur)

During this last week of the bootcamp, I have had the opportunity to explore the concept of MultiPath TCP, having previously studied MPTCP during a few months of stage for my final thesis.

In particular I used Mininet, which creates a realistic virtual network as the basis for the implementation of a script that simulates a network topology consisting of two entities linked by a dual channel (MPTCP implementation). The system provides as a result the maximum speed received after a ”iperf”.

Tamás Lévai and Balázs Németh (Budapest University of Technology and Economics, Hungary)

We were working on an unfinished ad-blocking FlowOS processing module (PM). It was supposed to remove the ad looking script elements from the HTTP flow. FlowOS is still in experimental phase and the module was just an old draft, full of undefined functions and misleading lines. What we have done: made a testbed, which consists of two virtual machines (VM) connected to the same network. One of them runs a simple web server, which hosts a web page with advertisements. The other virtual machine is the client. Our ad-blocker middlebox runs on the client machine’s kernel, under the supervision of the FlowOS.

With this test scenario we have found an issue in FlowOS, which is related to FlowOS’s TCP reverse path detection. As a workaround we made a three VM setup: [webserver] [middlebox with FlowOS] [client]. Besides this we expanded the initial ad-blocker PM with some command line interface options, for example help functionality. We also had to add a function the FlowOS API, which provides a high level handling of the HTTP flow. During the first test of our ad-blocker PM we found other problem: it deleted only the first ad. In the last days we were trying to repair this bug by rewriting our PM with kind help of the FlowOS designers. On the whole we understood the main mechanics of FlowOS and had some practice in kernel module debugging.

Wangbong Lee and Sangkil Park (ETRI)

FlowOS Experiments (2013.7.22 - 24)

I installed the FlowOS image to test FlowOS functions such as flow processing and processing modules (checksum, count) on my vmware machine. Developer of FlowOS commented that FlowOS implemented basic processing modules until now. Thus, FlowOS needs to improve to apply the packet application in terms of scalability and performance. I am trying to integrate to DPI processing modules as a pm (processing module). However, this experiment is not completed yet. I have to dig deeper the FlowOS to integrate the other functionality, and to improve its performance.

netmap Experiments (2013.7.24 - 26)

For testing and integrating some application, netmap is installed on Linux (ContOS5.4, 2.6.32-358.14.1.e16.x86_64). Some errors are happened. Makefile is changed such as removing some drivers. I am trying to modify tcpreplay using netmap. Tcpreplay(Open source software) replays network traffic and provides reliable and repeatible means for testing a variety of network devices. Fast packet transmission in tcpreplay is important in Linux machine based traffic generator.

Matthieu Coudron (Universty P.Curie UPMC-Paris6)

My intent when coming to the bootcamp was to write a generic subflow management module more powerful than the sysctl ndiffports currently available in the MPTCP implementation. When explaining this idea to Christoph Paash, he detailed a few performance improvements that could be made such as buffering NETDEV events or handling local addresses globally instead of the current per socket approach. I saw these propositions as a chance to better understand the code with the help of Mr Paash and to increase the performance of MPTCP.

It looks like that ”ifup” and ”ifdown” scripts send opposite NETDEV events. They are currently treated straightaway. I started introducing an eventqueue to handle these events in batches. All NETDEV_UP and NETDEV_DOWN events are now buffered into the queue and prior to their insertion we make sure there is no opposite (or similar) event in the queue in which case we remove the event found (respectively current event).

When an IP address would be added or removed, the kernel would loop through all sockets in order to reset addresses. Treating the previously described events in batches should limit the number of operations needed to update the sockets. I intend to profit from these changes to try to insert a better subflow management module. I still have to find the correct method so that it doesn’t impact performances.

I could not achieve all the previously mentioned points during the bootcamp but I will certainly keep working on these tasks as I strongly believe in MPTCP potential.

Romain Capron, Creciun Eduard and Schils Arnaud (Université catholique de Louvain)

“The packetdrill scripting tool enables quick, precise tests for entire TCP/UDP/IPv4/IPv6 network stacks, from the system call layer down to the NIC hardware. packetdrill currently works on Linux, FreeBSD, OpenBSD, and NetBSD. It can test network stack behavior over physical NICs on a LAN, or on a single machine using a tun virtual network device.”

Packetdrill is available at : https://code.google.com/p/packetdrill/

“The Multipath TCP (MPTCP) adds the capability of simultaneously using multiple paths to a regular TCP session.”

Multipath TCP is available at : http://mptcp.info.ucl.ac.be/

Packetdrill is already able to add the following option in regular TCP packets :

1. MP_CAPABLE

We can thus use all features of Packetdrill to test whether or not we can open a Multipath TCP connection with the kernel or a server.
To be fully compatible with Multipath TCP, Packetdrill should also be able to add the following options in regular TCP packets :

1. MP_JOIN
2. DSS
3. MP_FASTCLOSE
4. ADD_ADDR
5. REMOVE_ADDR
6. MP_FAIL

These options are being implemented and the code is available :
https://bitbucket.org/swax/packetdrill-mptcp

Niels L. M. van Adrichem (TU Delft)

Software-Defined Networking (SDN) paradigms concern recent proposals to separate computer networks data and control plane. Currently, forwarding devices (such a switches and routers) perform both roles, meaning they implement functions to forward packets efficiently (the data plane) as well as deploy routing discovery mechanisms that decide where data has to be sent to (the control plane). One of the disadvantages is that changes in routing discovery mechanisms and ACLs need to be deployed at all nodes, increasing the complexity of upgrading. Neither does any node have a complete overview of network state and usage, obstructing the use of QoS algorithms that need a full overview on topology state and usage. In SDN, one or more controllers are placed in the network that monitor network state, and perform path selection based on application needs and resource availability. OpenFlow, in particular, implements a protocol that standardizes communication between the controller and switches.
Due to administrative boundaries, a controller is often bound to a network of a single administrative instance or institution, such as datacenter networks and intra-networks connecting different locations of a single organization. For resilience, both types of networks often have redundant paths which can be exploited more efficient by using MPTCP. However, both types of networks, especially fat tree topologies used in datacenters, know many paths which are not link-dependent (meaning that they share one or more bottleneck-forming links). During this week we have investigated the behavior of MPTCP in multipath networks exploiting both link-dependent and link-independent subflow paths, calculated by a custom module for the POX OpenFlow controller.
Our experiments are run in a topology simulated by Mininet, which is controlled by a custom module implemented in the POX OpenFlow controller. We have used the following topology containing 3 paths

from host 1 to host 2 of which 2 paths are link-independent and a third one shares a bottleneck with both other paths. To delete variance introduced by the simulation using all CPU resources and to maintain link dependency, all links between switches are rate-limited at 5 Mbps.

Sivasothy SHANMUGALINGAM (Orange Labs)

The CHANGE bootcamp introduced many important networking and middleware components. One is the NetMap API. The netmap achieves line rate packet processing on a commodity hardware. Inside the netmap module, there is an arrangement to link the host network driver. At this point, netmap is limited to few NICs such as Intel Gbit, RealTalk and Nvidia. In this bootcamp, I explored to design a netmap component for Broadcom Netlink 10G NIC. Therefore, Netmap can be used ubiquitously.
I designed the interface between NetMap and Broadcom Netlink NIC and will continue to work after the bootcamp also.

Simone Ferlin (Simula research)

MPTCP can be used to experiment on a multi-homed environment such as the measurement nodes of the NorNet testbed (http://nornet-testbed.no/). The focus in our research is to increase connectivity robustness: Having stable and uninterrupted Internetconnectivity is becoming increasingly important. There are several critical applications that depend on the availability of the network or ISP (Internet Service Provider). As an example, connectivity problems are rather random and could be caused by hardware failure or, in more extreme scenarios, natural disasters. In order to increase the robustness of Internet connectivity, we investigated the potential of multi-homed and/or multi-linked systems. Some answers we are looking for in the NorNet project are, how multi-path transport, e.g., with Multi-path TCP(MPTCP) or Concurrent Multi-path Transfer for SCTP(CMT-SCTP), can efficiently and fairly make use of multi-homing? Can end-system multi-link operation increase mobile broadband connection robustness? Does MPTCP answer all the questions regarding data scheduling through different network paths or ISP's in a mobile broadband scenario? How can different ISPs be /evaluated/at the end-system? In order to be able to properly investigate these questions, we are conducting research and collecting data from real mobile broadband network providers in Norway. The NorNet project is building up a multi-homed and programmable testbed be used for measurements and experimental networking research. NorNet has two main components: NorNet Core and NorNet Edge. When fully operational, NorNet Core will consist of more than ten programmable multi-homed sites, whereas, the NorNet Edge will consist of several hundreds of smaller measurement nodesconnected to all mobile broadband providers in Norway.

Dario Valocchi (Universit'a degli Studi di Perugia)

For the CHANGE bootcamp software project, I chose to work on the Netmap framework. After compiling and installing Netmap on a Linux system, I decided to evaluate the performance of the framework in different scenarios. In the first scenario the Netmap Bridge has been used. Fig. 4.1 shows the involved entities. The Netmap Bridge is an application provided by the Netmap Framework, able to bridge two

interfaces, without copying packets from the memory space of a driver to the memory space of another one. The application uses a set of primitives able to swap the buffers of the Netmap rings between two "netmapped" drivers that share the same user space memory. To generate the UDP packets used in the test, the Netmap PktGen has been used. This simple packet generator exploits the Netmap capabilities to achieve high throughput packet generation. On N3 runs the same application, but in receive mode. In the second scenario the Netmap Bridge capability to bridge to the Linux stack has been used in order to introduce Layer 3 routing in the test. Fig. 4.2 shows the protocol entities involved in the test. A packet incoming from a "netmapped" interface is sent to the classic network stack of the Linux Kernel, via the first Bridge, and than routed by the kernel IP layer and sent, via the second bridge, to the right outgoing interface. In order to evaluate performances in this scenario has been necessary to change the packet generator. Packets generated by the Netmap PktGen were always dropped by the Linux Kernel after being processed by the PREROUTING chain of the mangle table of Iptables. For this reason has been decided to use the well-known Iperf application. The third and last scenario uses the standard Linux network configuration, without using Netmap. Fig. 3 show the configuration of the third scenario. Also in this case Iperf has been used to evaluate performances. Table 4.1 shows the collected results for the configurations explained above. One can notice that using Netmap with the classic Linux network stack is equal, or even worst, than not using Netmap at all, even considering that IP routing is executed all in kernel space, therefore without Figure **??**. Testing scenario using the classic Linux network stack any time expensive copy between kernel space and user space. After collecting these results has been decided to create a new application, inspired by the Netmap Bridge, able to exploit the buffer swapping paradigm and the zero-copy approach of Netmap in order to provide basic Layer 3 features. The application will be able to manage a variable number of network interfaces. Each interface will run a "netmapped" driver and the application will manage all the traffic using exclusively the Netmap framework. It will use the system routing table and a state-of-art software implementation of a longest prefix matching algorithm. The goal of this application is to point out whether the bottleneck at layer 3 is due to the algorithm that decides the right interface for forwarding or to the memory management and to find out if any improvement is possible making use of the Netmap framework.

Scenario	Throughput (kBps)
1	7013,4
2	1276
3	127

Table 4.1: Throughput in kByte per second for the different scenarios

Vito Domenico Piserchia (Dreamlab Technologies AG)

The CHANGE Bootcamp was extremely useful to meet the other participants of the project and hear from them the latest results of their research. All of this happened during the first week of the bootcamp.

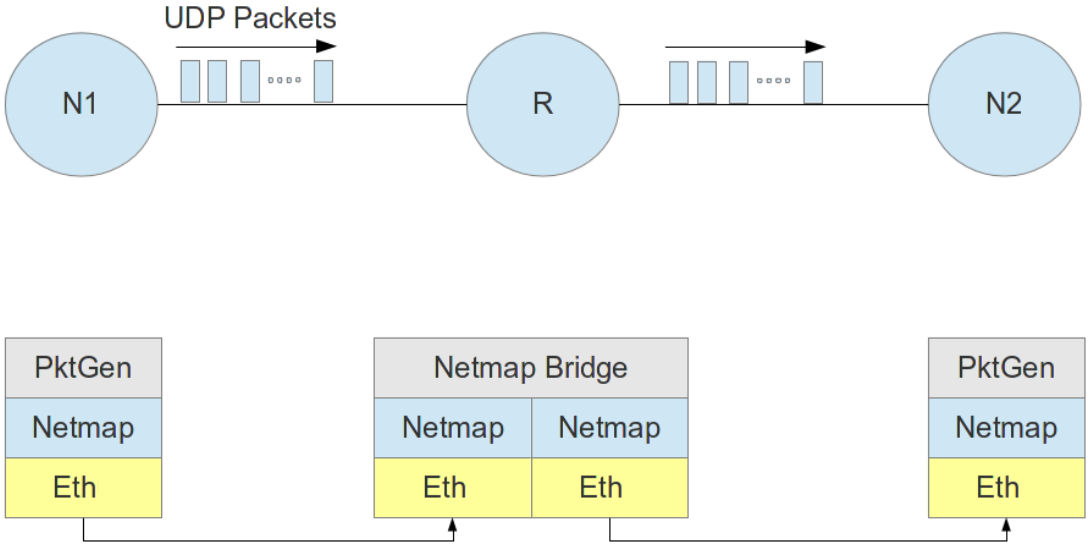

Figure 4.1: Testing scenario using Netmap Bridge

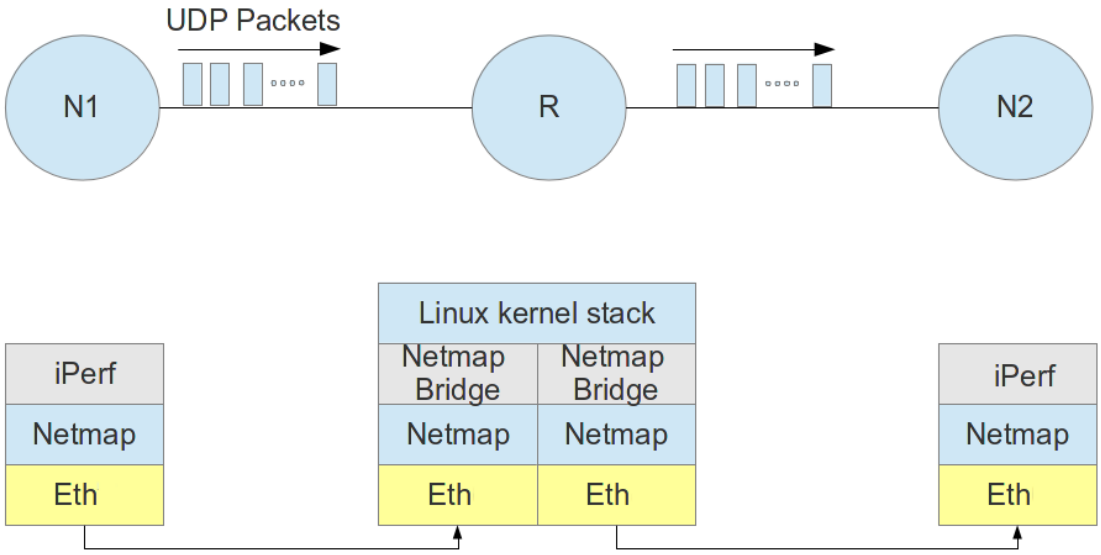

Figure 4.2: Testing scenario using netmap bridge with Linux kernel stack

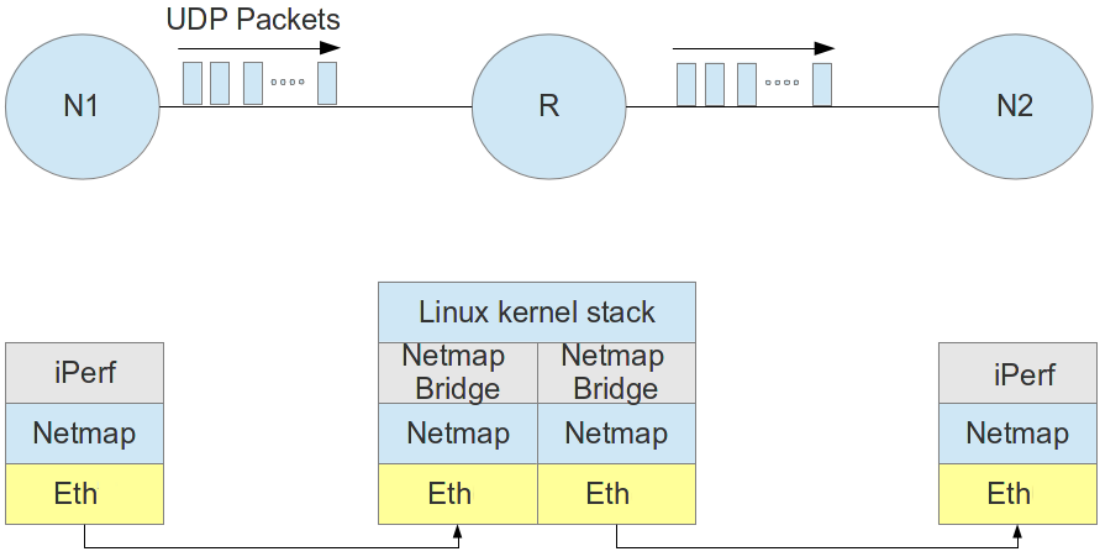

Figure 4.3: Testing scenario using the classic Linux network stack

I dedicated the second week to put my hands on two of the CHANGE projects, namely the NETMAP fast packet IO processing engine and the ClickOS virtual environment.

In the first two days of this week, I tried to compile and run the example tools coming with the NETMAP software distribution. I found several compilation problems basically related to the changes to some NIC driver interfaces. After solving these issues I was able to run the examples.

The remaining days of the week were dedicated to make experiments with the ClickOS, and getting the needed confidence with the framework and the main concepts behind it. So, first I tried to get some experience with the examples provided by the NEC researchers and doing this way I was able to build a first implementation of a Click processing module, that I called FlowPinner (FP).

Without going to much into the implementation details, this FP is able to calculate a hash for each packet coming through the system and in this way distribute packets to the different output ports of the module. Also, using a symmetric hashing function, the FP is able to balance the traffic load as all the packets belonging to the direct and reverse flows are sent to the same output port.