31-03-2015

Open Call Deliverable OCB-DS2.1
Final Report on Experimentation & Results (AUTOFLOW)

Open Call Deliverable OCB-DS2.1

Grant Agreement No.:	605243
Activity:	NA1
Task Item:	10
Nature of Deliverable:	R (Report)
Dissemination Level:	PU (Public)
Lead Partner:	UPRC
Document Code:	GN3PLUS14-1285-43
Authors:	Panagiotis Demestichas, Evaggelia Tzifa, Kostas Tsagkaris, Marios Logothetis, Andreas Georgakopoulos, Aimilia Bantouna, Dimitris Karvounas, George Poulios, Vassilis Foteinos, Dimitris Kelaidonis, Aristotelis Margaris, Kostas Petsas

The research leading to these results has received funding from the European Community's Seventh Framework Programme (FP7 2007–2013) under Grant Agreement No. 605243 (GN3plus).

Abstract
This deliverable presents the final integrated experimentation framework of AUTOFLOW and examines the 5 selected experimentation scenarios: i) A framework for autonomic management of SDN/OpenFlow networks, ii) Governance of Autonomic Control Loops, iii) Operator-driven traffic engineering in SDN/OpenFlow networks, iv) Load prediction in SDN/OpenFlow networks, and v) Autonomic traffic engineering with load prediction in SDN/OpenFlow networks. The conducted research resulted in a complete, validated solution that combines ANM and SDN for the operator-friendly and efficient control and management of networks.

Document Revision History

Version	Date	Description of change	Person
1	16-03-15	First draft issued	UPRC
2	31-03-15	Pre-final version	UPRC
3	08-04-15	Final version	UPRC

Table of Contents

Table of Figures

Table of Tables

Executive Summary

This deliverable is the main outcome of the conducted studies in WP2 “Experimentation & Evaluation”. This work package was responsible for defining the evaluation process, executing the selected scenarios, and extracting the experimental results. In this document, the final integrated experimentation framework is presented, as well as the 5 selected experimentation scenarios and results:

i. A framework for autonomic management of SDN/OpenFlow networks

ii. Governance of Autonomic Control Loops

iii. Operator-driven traffic engineering in SDN/OpenFlow networks

iv. Load prediction in SDN/OpenFlow networks

v. Autonomic traffic engineering with load prediction in SDN/OpenFlow networks

The conducted research resulted in a complete, validated solution that combines ANM and SDN for the operator-friendly and efficient control and management of networks.

1 Introduction

This deliverable is the main outcome of the conducted studies in WP2 “Experimentation & Evaluation”. This work package was responsible for defining the evaluation process, executing the selected scenarios, and extracting the experimental results. This part of the work started from the execution of the experiments for the selected scenarios based on the configuration and specification of the components defined in WP1 “Preparation, Development, Integration”. Afterwards, there was an initial validation of the results. A detailed analysis allowed us to come up with useful recommendations as well as improvements for the framework. Furthermore, additional extensions were identified for fine-tuning of the integrated components and experiments (provided as a feedback to WP1), as well as for improvements of the Autonomic Network Management (ANM) components. Finally, a working implementation of the experiments was developed and used as an input for the demonstration activities of WP3 “Demonstration & Dissemination”.

The following list summarizes the objectives of WP2:

- To execute the experiments for the proposed scenarios and extract results.
- To perform an analysis of the results and report on the findings of this analysis.
- To validate the results derived from the AUTOFLOW experiments and come up with useful recommendations.
- To utilize the validation results from experiments for identifying potential future features, experiments as well as improvements and their prioritization.
- To provide feedback to Task 1.2 of WP1 for experimentation improvements.
- To collect and provide feedback to GN3plus main holders regarding the usability and support offered by the GÉANT OpenFlow-enabled facility and tools.
- In parallel, to set-up a showcase of the experiments that will comprise a demonstrable form of the experimentation with ANM functionalities and interfaces.

The present document presents the research findings and results derived from the AUTOFLOW experimentation. The final integrated experimentation framework is described in Section 2. Sections 3-7 discuss the 5 examined experimentation scenarios:

i. A framework for autonomic management of SDN/OpenFlow networks

ii. Governance of Autonomic Control Loops

iii. Operator-driven traffic engineering in SDN/OpenFlow networks

iv. Load prediction in SDN/OpenFlow networks

v. Autonomic traffic engineering with load prediction in SDN/OpenFlow networks

Finally, Section 8 concludes this deliverable and presents the future plans of AUTOFLOW.

2 Experimentation Framework

The main objective of AUTOFLOW is to exploit both ANM and Software Defined Networking (SDN) in order to provide a realistic solution to the complexity of network management. For this purpose, we designed a framework comprising of these two emerging technologies (Figure 2-1). At the top lies the ANM Core, which includes a set of blocks that provide fundamental functionalities (Governance, Coordination and Knowledge) for the efficient operation of autonomic mechanisms. The latter are deployed as SDN applications on top of an SDN controller and use the controller's northbound Application Programming Interface (API) to interact with the network. A detailed description of this framework and its components can be found in milestone M1.1 "Detailed experiment description and specifications" [1]. Milestones M1.2 "Prefinal description of Software" [2] and M1.3 "Final description of Software & Manual" [3] provide guidelines for accessing the developed software.

2.1 ANM Core

The main role of the ANM core is characterized by the following objectives: to enable a seamless integration and expandability ("plug & play" and "unplug & play") as well as to ensure a trustworthy interworking of autonomic mechanisms within an operator's management ecosystem. In particular, the Governance block aims at giving a human operator a mechanism for controlling the network from a high level business point of view, without the need of having a deep technical knowledge of the network. Information regarding autonomic mechanisms, monitoring parameters, and alerts are sent to the operator, while operator's policies are translated into specific parameters, relevant to the operation of the mechanisms, and sent to them through this block. The Knowledge block manipulates information and knowledge to be exploited from the other blocks and mechanisms. The role of the Coordination block is to ensure the proper sequence in triggering of mechanisms and the conditions/constraints under which they will be invoked taking into account operator and service requirements.

2.2 Autonomic mechanisms

The ANM core is responsible for the efficient operation and coexistence of multiple autonomic mechanisms (Autonomic Control Loops - ACLs). However, in our studies, we focus on two specific mechanisms, an autonomic Traffic Engineering (TE) mechanism (Core-TE) that addresses the problem of policy-based TE in

SDN/OpenFlow networks and a Load Prediction (LP) mechanism that addresses the problem of congestion prediction.

Core-TE is based on a heuristic algorithm, which is designed as an autonomic MAPE-K control loop [4]. Therefore, 4 main blocks (functions) can be identified, namely Monitor, Analyze, Plan and Execute. In the Monitor phase, the algorithm retrieves measurements from the network (through the controller's northbound API) and derives the status of the links (e.g. utilization level). In the Analyze phase, it checks the operator's desired policy (e.g. energy efficiency) and identifies the characteristics of the paths that will be selected for the accommodation of traffic in the network (e.g. multipath degree, disjointness level). Afterwards, the optimal paths are selected in the Plan phase. Finally, there is the enforcement of the decision to the network, the establishment of the appropriate paths by sending the necessary commands to the network elements (e.g. switches) through the controller (Execution phase). More information about this algorithm can be found in [5].

The LP mechanism addresses the problem of congestion prediction in SDN/OpenFlow networks and targets at offering the system the capability to proactively handle such situations. Towards this direction, the LP combines network data that can be monitored and reports how close a core link is to a potential congestion. This information can be delivered through the ANM core and in particular, through the knowledge (KNOW) ANM core block, to the Core-TE so as to proactively treat potential congestions of the links and avoid them. The main merit of the mechanism is that it targets at discovering potential congestion proactively, i.e. before they reach the user or at least at their very beginning. This enables the network operator or the system per se to also handle it proactively, minimizing the consequences that reach the user, e.g. deterioration of the network performance/ application and thus deterioration of the Quality of Service (QoS) and/ or the Quality of Experience (QoE).

2.3 Controller

Many SDN controllers exist today including POX [6], Ryu [7], Trema [8], Floodlight [9], and OpenDaylight [10]. The authors in [11] provide a feature-based comparison of these topmost five controllers. In our case, there was a need for a controller that would offer an easy-to-use northbound API, remotely accessible through a REST (Representational State Transfer) API, as HTTP (Hypertext Transfer Protocol) REST interfaces would facilitate the integration with the rest of the framework. Therefore, we selected the Floodlight controller. In addition, monitoring statistics regarding the network's status that could be retrieved from the Floodlight controller were also sufficient for the needs of the autonomic TE mechanism (Core-TE). Moreover, a Java client-side library for Floodlight's northbound API that abstracts out all HTTP/JSON (JavaScript Object Notation) details was implemented. This library, besides being reusable during any Java (Floodlight) SDN application development, served the adaptation of the ANM core/mechanisms to the controller, simplifying development and integration. Finally, utility methods were developed that were both generic and useful.

2.4 Infrastructure

The GÉANT OpenFlow facility that was exploited in our experiments is co-located in five GÉANT Points of Presence (PoPs), in Vienna, Zagreb, London, Amsterdam and Frankfurt. The facility's management and control

plane elements/software are hosted in the Frankfurt PoP. This includes the GÉANT OpenFlow Control Framework and the FlowVisor software [12]. FlowVisor is a network virtualization application, which can be considered as a proxy protocol to sit logically between the multiple controllers and the OpenFlow switches in order to allow multiple controllers to share the same network infrastructure without interfering with each other. The main purpose of FlowVisor is to provide virtualization in OpenFlow networks; therefore it does not provide many traffic engineering mechanisms. Moreover, in the GÉANT facility, computing resources are offered as Virtual Machines (VMs). Network resources are offered utilizing software-based OpenFlow switches based on Open vSwitch [13] and network links that interconnect the OpenFlow software switches. The VMs can be used as traffic producers/consumers.

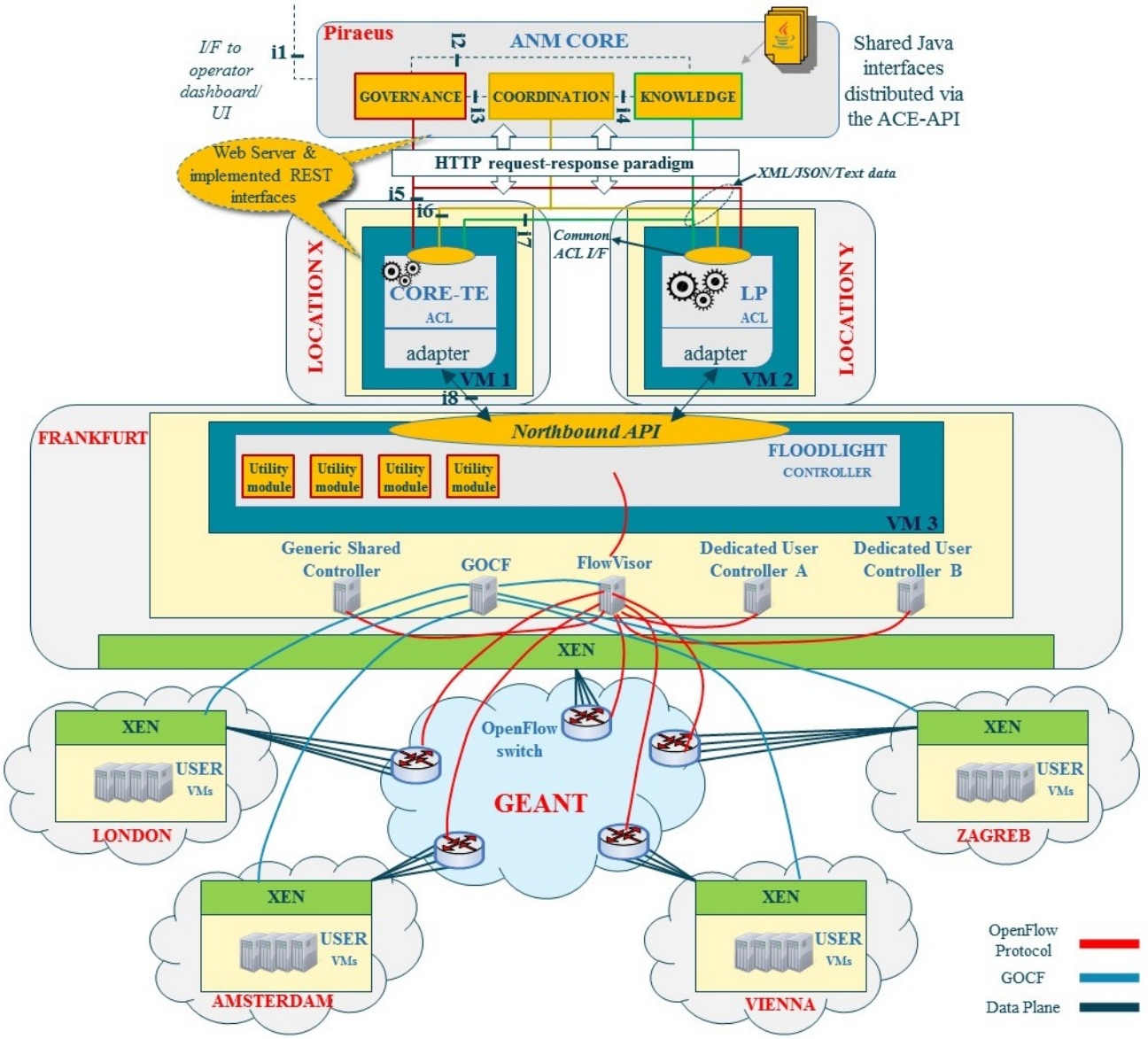


Figure 2-1: Experimentation framework for autonomic software-defined networks

3 Experiment #1: A framework for autonomic management of SDN/OpenFlow networks

3.1 Introduction

One of the main objectives of AUTOFLOW was to study and shed light on the relationship between ANM and OpenFlow/SDN and to make a position statement with respect to their synergy and interworking for the management and control of future networks. During this project we realized that the real keywords for AUTOFLOW are integration, synergy and efficient interplay. ANM and SDN technologies have appeared to address the operational challenges of today's networks by providing simplified management and control, decreasing the burden of human presence and enhancing the ability of network elements to self-govern their behavior in an autonomic manner. In general, an interest around the investigation and actual linking between ANM and SDN architectures/approaches is apparent. In AUTOFLOW, we believe that SDN/OpenFlow can be seen as an enabler for simplifying the introduction of autonomics into networks and network management, i.e. it can be used to make it possible to inject and track more autonomic functions/loops into the networks, purpose-wise and freely, but at the same time in a well-defined manner. In addition, ANM technologies can be used to provide a management platform for managing SDN and software network applications featured with autonomics and/or for providing guidelines for developing such autonomic applications. Conclusively, AUTOFLOW aimed at designing a framework able to support programmability, network virtualization and autonomic network functions (SDN applications) for the management and control of future networks.

Network virtualization has been promoted recently as a key solution to the "well-known" ossification problem of the existing Internet architectures and is proposed as an integral part of Next Generation Networks [14]. The key advantages that network virtualization provide are diversity, flexibility, reliability, security and increased manageability. Virtualized networks are highly dynamic, as links and nodes could be reconfigured extremely quickly. Virtual routers could migrate on demand, based on resource availability, in order to create an energy efficiency plan and save energy from the physical locations [15]. The GÉANT facility supports virtualization technologies, allowing multiple alternatives in the deployment of the architectural components. Framework components, SDN controller, network elements, nodes or VMs may also move logically. In addition the ANM components and the controller are hosted in separate virtual environments, in order to meet the requested performance levels.

Other concepts that present autonomic solutions integrated with the use of OpenFlow and/or other SDN approaches can be found in literature. In particular, authors in [16] claim that with the use of an autonomic

system, manual operation and time-consuming configuration changes will be performed automatically and valuable operators' time will be saved, enabling them to perform efficiently high-level network functions, such as network design, planning and optimization. In line with the autonomics, SDN philosophy promotes the reduction of network management complexity, the programmability of the data plane, the centralization of the control plane and empowers programming flexibility [17]. In [18], the authors explain the role of SDN and present the capabilities of network operating systems to gain greater governance of the control plane within a given network. By using this SDN approach and the specifications provided by OpenFlow, they show how QoS is managed and defined by a centralized network controller. Also, they claim that SDN and OpenFlow could adopt other traditional networking techniques as well, like load balancing. Further to that, in [19] an autonomic QoS policy enforcement framework based on SDN (PolicyCop) is presented. PolicyCop provides an interface for specifying QoS policies and exploits the SDN controller to enforce them. It also takes advantage of the control applications to monitor the compliance of the policies and autonomically adjusts the control plane rules to the changing traffic conditions. Another autonomic network management model is CogMan, which is based on a cognitive model [16]. Authors argue that the cognitive control loop of CogMan provides reactive, deliberative and reflective loops for managing systems based on an analysis of the current status. In order to validate the proposed architecture and the control loop, it is applied in SDN networks. Procera [20] is a network control framework that helps operators to express event-driven network policies that react to various types of events using a high-level functional programming language. Procera effectively serves as glue between high-level event-driven network policies and low-level network configuration. Finally, authors in [21] propose PayLess – a monitoring framework for SDN. PayLess provides a flexible RESTful API for flow statistics collection at different aggregation levels. It uses an adaptive statistics collection algorithm that delivers highly accurate information in real-time without incurring significant network overhead.

Obviously, the interest around the investigation and actual linking between ANM and SDN architectures/approaches is apparent. Yet, it is still in its infancy, while a consistent justification/specification and positioning that can assist in reaping the benefits of such interplay is not adequate in the existing literature and industrial/research studies. Let alone, novel experiments of existing ANM solutions, in real OpenFlow-based facilities are still missing from the research/academic experimental field, making essential and of paramount importance the proposed AUTOFLOW experimentation studies. Therefore, this section is dedicated to the first part of the AUTOFLOW experimentation, where the main target is to demystify and fully clarify the relationship between SDN and a novel ANM solution by experimenting over the real testbed of GÉANT. In addition, apart from the actual integration, the performance of the developed ANM/SDN framework is also examined.

3.2 Actors, Setup, Topology

The GÉANT OpenFlow testbed is a facility designed to support software-defined networking experiments and prototyping. This is located in five GÉANT PoPs [22] in the cities below:

- London
- Frankfurt
- Vienna

- Zagreb
- Amsterdam

The connection with the GÉANT OpenFlow-enabled facility is achieved with the use of VMs that are hosted on the servers offered by the facility. Computing resources are offered as VMs upon dedicated physical servers using Xen [23] hypervisor-based virtualization.

General Server Specification	
Number of CPUs	≥ 2
Number of cores per CPU	≥ 4
CPU Cache Size	≥ 6 MBytes
CPU Frequency	≥ 2.60 GHz
Memory Size	≥ 16 GBytes
RAID Controller	RAID 1/5 with SAS HDDs &≥ 4 HDDs support
HDDs	≥2 x SAS 146 GB
Network Interfaces	12 Gigabit Ethernet

Table 3-1: General server specification

Two general-purpose servers (Table 3-1) are installed in each of the five GÉANT PoPs. Each of the servers that are located in PoPs are either the host of a software-based OpenFlow switch (Open vSwitch) on top of a native Linux Debian distribution, or the host of a Xen hypervisor for the instantiation of multiple VMs that can be allocated to user slices.

The data plane topology of the GÉANT OpenFlow facility is configured as a full mesh graph, so that every OpenFlow switch has direct connectivity with all of the other OpenFlow switches through direct connection with the routers of each facility and further on through L2MPLS VPN circuits.

The network virtualization technology used by the GÉANT OpenFlow Facility is called “network slicing technique”. This network technique dynamically allocates one or a range of VLAN IDs to each user slice. The network is partitioned per Open vSwitch interface (physical or logical) and user traffic is distinguished by the VLAN ID. VLANs can be involved in experimentation within the slice when a set of VLAN IDs is allocated to a slice. Thus any of the experimenters can use its own range of VLAN IDs for routing purposes on top of the

OpenFlow topology. Network resources are offered by utilizing software-based OpenFlow switches based on Open vSwitch (OvS) and network links of 1Gb that interconnect the OpenFlow software switches. The switches are logically separated for each research group with the use of VLAN technology. Moreover the hosts of the software-based OpenFlow Switches are hosted in the pre-installed servers in each PoP. These hosts can be reachable through the Internet, as they use public IP addresses:

- London: 62.40.110.137
- Frankfurt: 62.40.110.71
- Vienna: 62.40.110.110.3
- Zagreb: 62.40.110.101
- Amsterdam: 62.40.110.35

In addition, the VLAN that we are currently using in the GÉANT OpenFlow facility has the 256 ID and the private "WAN" network that the switches are connected to is the 192.168.1.0 /24. The IP addresses of the hosts that are connected to the relevant OpenFlow switch are:

- London: 192.168.1.3
- Frankfurt: 192.168.1.2
- Vienna: 192.168.1.4
- Zagreb: 192.168.1.5
- Amsterdam: 192.168.1.1

The AUTOFLOW experimentation framework includes the ANM prototype that lies above the GÉANT OpenFlow-enabled facility. The ANM part consists of the three core blocks:

- Governance
- Coordination
- Knowledge

and the two ACLs :

- Core Traffic Engineering (Core-TE) ACL
- Load Prediction (LP) ACL

In this case, the ANM mechanisms are considered as SDN applications, where the complete control of the programmable network is rendered feasible with the use of the controller. The controller that is used for the experiments is the Floodlight controller. The integration between the ACLs and the controller's northbound (NB) interface has been taking place through HTTP REST.

Both the controller and the ANM components can be hosted and operate in any environment, physical or virtual. The use of virtualization for both the controller and the ANM components promotes the manageability of network resources and meets the requirements of an efficient SDN operation. Furthermore, the ANM prototype can operate regardless the location with the simple use of an Internet connection. That means that it can operate in a personal laptop, as an instance in a physical or virtual server or even as a "cloud" network service for SDN, LAN or WAN environments.

As already mentioned, the integration shall take place based on the appropriate coding/modification of the adapters used by the involved ACLs to interact with the network elements. The technological heterogeneity among networking equipment dictates the use of specific adapters per ACL. In our case, with the five software OpenFlow network elements (Open vSwitches), the generic shared controller, which can be deployed at the GÉANT facility, is a suitable OpenFlow adapter that allows for the interaction of the ANM core / ACLs components with the Open vSwitches.

Since the experiments are taking place in a real environment, network characteristics may vary between the switches. Therefore metrics such as packet loss, delay and jitter variation are relevant to the traffic that is carried through the physical interface of each network element. The measurement of the metrics could be performed directly from the controller. The operator could retrieve info such as:

- Topology details (MAC addresses, IP addresses, link status per switch/host)
- Number of packets, bytes, flows, dropped packets, errors per switch
- Slice details (VLAN ID, Action ID)
- Policy details (e.g. utilization levels per network element, time)

Further to that and in regards to the needs of this project, our research team has developed "Metric Monitor", a graphical tool that extends the controller by providing numerous metrics from the chosen path(s) in real time. Moreover, it provides a graphical interface of the established paths of the topology. Some of the metrics that can be monitored by the operator through the Metric Monitor tool are:

- Transmitted packets / bits per second
- Received packets / bits per second
- Collisions
- Transmit drops / errors

- Received drops / errors
- Received CRC / frame errors

In this scenario, the integration of the ANM and SDN is demonstrated, indicating that it is a realistic solution. More specifically, this experiment validates the beneficial synergy and interworking of ANM with SDN for better and more efficient control and operation of future networks.

In the previous section, Figure 2-1 depicted our solution for the mapping of ANM core / ACLs components in the GÉANT OpenFlow-enabled facility. The required tools for interfacing with the OpenFlow facility are implemented with special consideration for reusability. In particular, the API that the controller exposes is going to be used in order to achieve the interaction with the OpenFlow network elements below. Therefore, the network's status and any monitoring parameters in general are collected when needed from the ACLs, with a simple call of the API exposed by the controller. Additionally, in order to push down the configuration to the proper software routers/switches, the API is exploited so as to configure the suitable fields of the flow tables, as it is being defined (OpenFlow protocol [24]), in order to realize the network operations (e.g. switching, multipath, etc.).

Therefore, the ANM core / ACLs components are going to interact with the OpenFlow controller in order to monitor and/or configure the elements of the core network. Particularly, the Core-TE ACL is responsible for the configuration of the elements and forwards the updated flow tables to the routers through proper interaction with the controller. On the other hand, the LP ACL monitors the traffic at the core network and learns its characteristics over time. The ANM core / ACLs components can be considered in this case as SDN applications, where the complete control of the programmable network is rendered feasible with the use of the controller. Other scenarios and experimentations will follow to demonstrate and evaluate the use and the operation of the ACLs.

3.3 Case 1: ANM/SDN Integration

3.3.1 Storyline

Let's assume a topology similar to that of Figure 2-1. The operator may activate the framework's components directly from a virtual server that is located at the Data Centre of the University of Piraeus. For security reasons each of the operators has a unique username and password in order to gain access to the server. The server has already an Internet connection in order to obtain network connectivity with the SDN controller that is located in Frankfurt.

Meanwhile, for the purposes of this scenario the host that is located in Frankfurt sends traffic to the host that is located in London. The sender creates UDP flows of randomized size and duration. Traffic is generated using Iperf [25]. Due to the data exchange the end-to-end active paths appear in the main section of operator's dashboard. In this manner, the operator monitors the status of the network and is allowed to enforce a high-level policy to the network's operation, while the QoS is guaranteed. We should mention that the hosts may be seen as clients that have a contract for specific quality thresholds. The operator thus is able to maintain the

control of the network while being granted an opportunity to observe the status of the exchanged data in real time.

3.3.2 Execution & Results

The architectural components presented before are deployed, instantiating an OpenFlow/SDN based ANM prototype that simplifies the management and control of network and services. The self-x attributes provided by the ANM ecosystem are combined with the programmability of network devices offered from SDN, facilitating advanced customizability of network control and forwarding behaviors.

As we indicated earlier, the network infrastructure is controlled by an identical VM server that is hosted in Frankfurt and is running the OpenFlow controller. Hence, every change in the network related to the creation, modification and deletion of flows is realized through this controller on the OpenFlow-enabled switches. In addition, the ANM components should have connectivity with the controller, in order to provide the needed information that will lead in the enforcement, monitoring and control of a high-level policy. Especially, for this experiment the ANM components are hosted in different locations. Initially they are located in a server of the Data Centre of the University of Piraeus and afterwards in a personal laptop. Both solutions have been tested and evaluated.

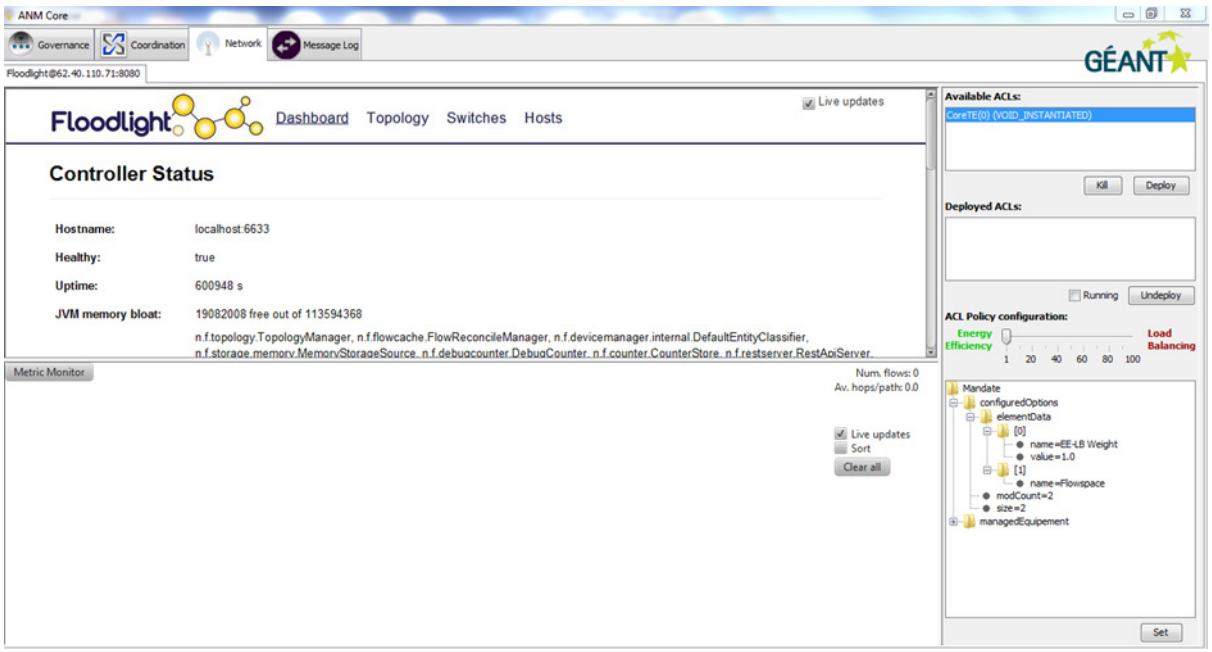


Figure 3-1: Operator's dashboard

For the purpose of the evaluation, we start the ANM core component first. As soon as there is connectivity with the controller, the Floodlight Web interface appears in the upper left pane of the developed dashboard (Figure 3-1). From left to right, the tabs "Dashboard", "Topology", "Switches" and "Hosts" present information relevant to the interconnections of the switches, Layer 2 & 3 details for the switches, packets/byte/flows per switch and

Layer 2 & 3 information for each host. Due to the scenario's needs, we exploit the learning switch method [26]. Thus, the switch examines each packet and learns the source-port mapping. Thereafter, the source MAC address is associated with the port. If the destination of the packet is already associated with some port, the packet will be sent to the specific given port, or else it will be flooded to all ports of the switch.

The ACLs are displayed in the right up pane of the dashboard and are ready for deployment ("Deploy" option). When deployed, the ACL remains at an operational state until the operator decides to deactivate it ("Un-deploy" option).

For this scenario we generate UDP traffic from Frankfurt to London. UDP flows of randomized size and duration are generated into the GÉANT facility. When the sender starts the data exchange, the four flows appear in the left down pane (Figure 3-2). The operator's awareness is optimized by the view of the established end-to-end flows in the network. IP addresses, application ports and network elements appear on the screen providing a powerful tool used for monitoring, analysis and troubleshooting.

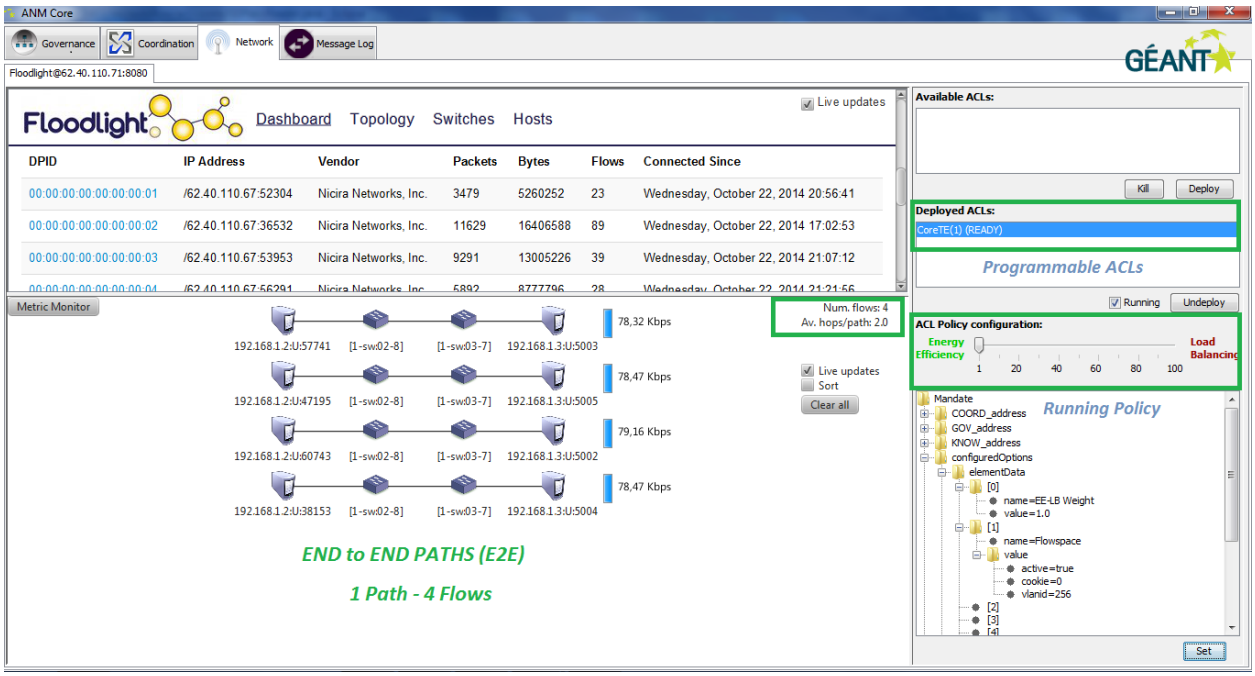


Figure 3-2: View of end-to-end flows in the network

In addition, at the metric monitor (Figure 3-3) we have selected to monitor in real time the total transmitted bits per second and per switches' port.

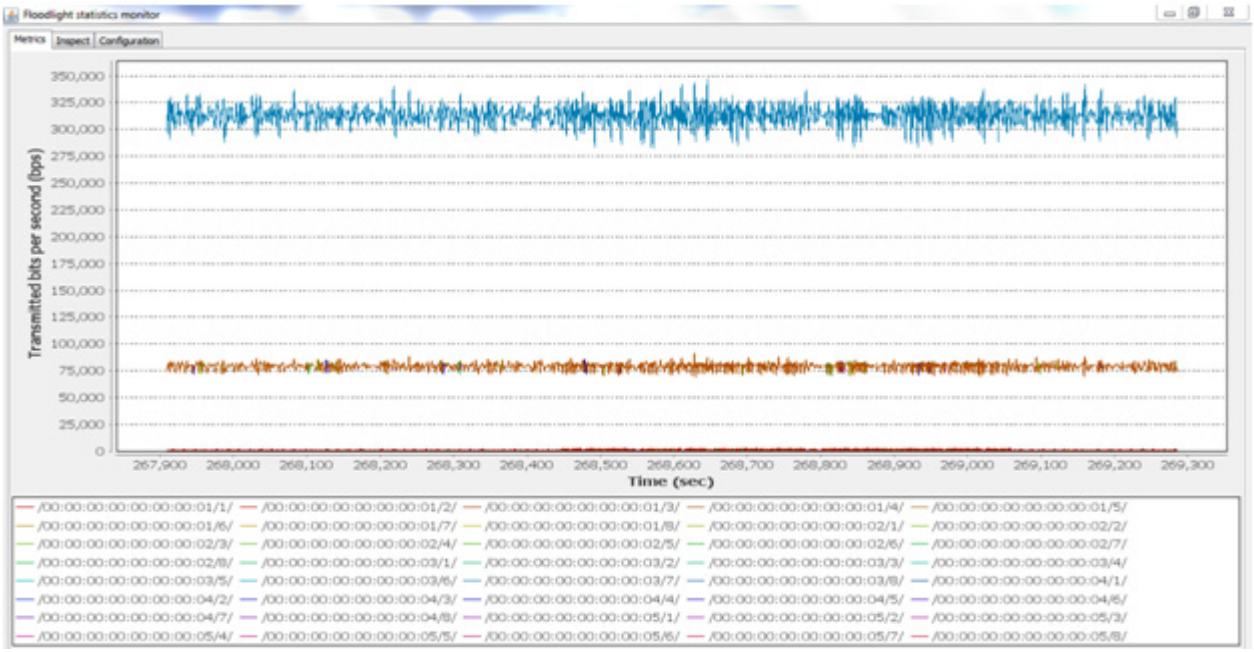


Figure 3-3: Metric monitor

For the purposes of this experiment, we have downloaded the ANM components in a personal laptop. For security reasons, we have authorized specific personnel (AUTOFLOW team) to grant such access. In order to observe if there is any difference in the operation of the framework, we have requested to use a different public IP address for the ANM prototype. The new address that has been assigned to our host is 195.46.9.68.

As soon as the ANM core component runs, the application opens and the OpenFlow controller appears in the left up pane of the dashboard. The framework is operational, while the interplay between the controller and the ANM components leads in a flexible management solution from everywhere, regardless the environment, with the simple use of an Internet connection. In the figure below (Figure 3-4), the output of the tcp-dump command from the host in Frankfurt, where the controller is located, is presented. The SDN controller sends TCP traffic that is destined to 195.46.9.68 public IP address.

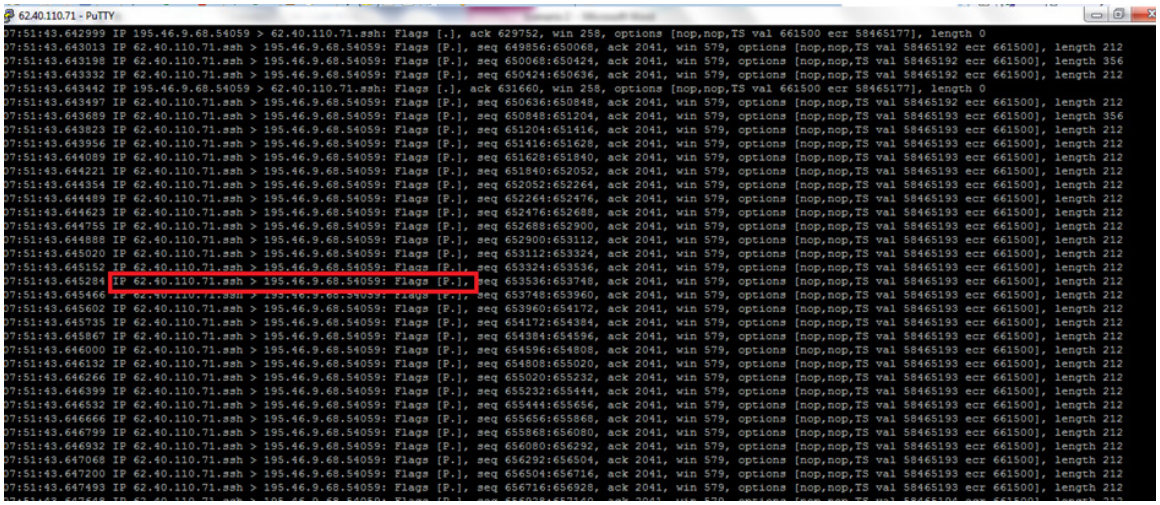


Figure 3-4: Tcpdump – IP 195.46.9.68

Moreover, the traffic generation procedure is repeated. Based on different traffic models, different flows of traffic are generated into the GÉANT facility. The new end-to-end paths appear in the left down pane of the dashboard (Figure 3-5), as expected. In the metric monitor view, the operator may follow the changing conditions in the network and observe any critical situation might occur.

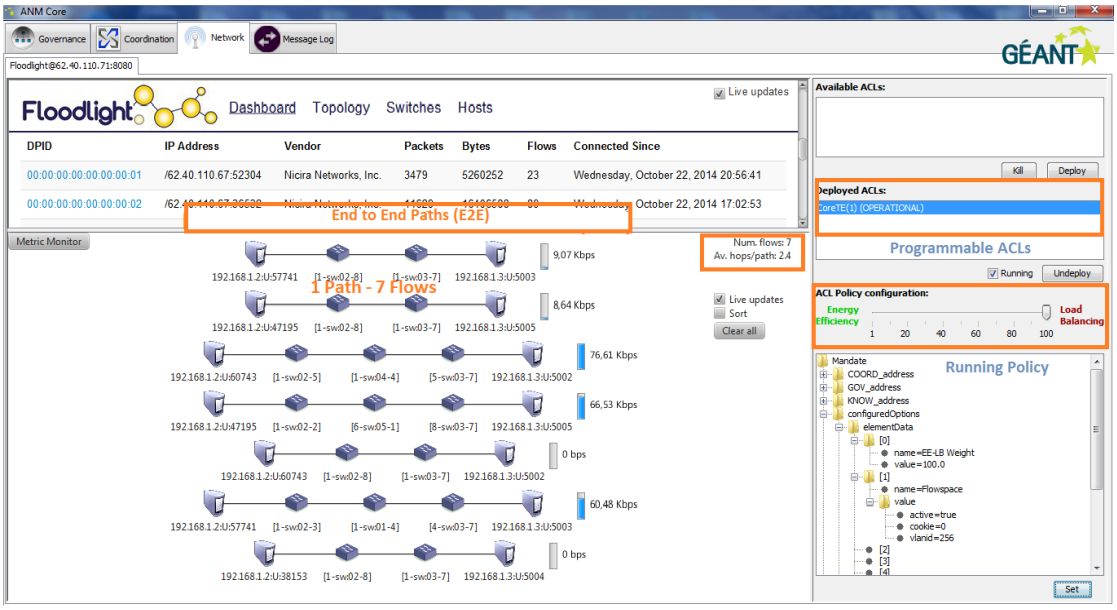


Figure 3-5: View of end-to-end flows in the network

From our point of view the reliability and the availability of the framework is an important requirement for its efficient operation. In general, placing controllers in different locations provides major fault tolerance benefits. In the GÉANT OpenFlow-enabled facility, there was no need to operate with two controllers simultaneously. Nevertheless, the controller failover scenario is extremely easy to setup and deploy. The failover scenario works by having two servers with identical content on them – a primary server and a secondary server. Both primary and secondary servers have the relevant software (Floodlight files). In our case, the primary server is located in Frankfurt, while the backup could be every other host that is located in the facility. From the ANM framework part, the operator monitors the primary server and detects whether there is a problem. The second controller can be operational by running a script that will enable the controller functions to be operational in a few seconds from another location. This change will enforce the secondary controller to act as primary. Once the primary server is functioning again, the same steps are applied in order to get back to normal.

3.4 Case 2: Performance

3.4.1 Storyline

One of the fundamental challenges of SDN is how to handle high-performance packet processing flows in an efficient and reliable manner, while achieving high-level goals. Several studies have been performed in this area that take under consideration the two key elements, performance and programmability/flexibility. In these experiments, we follow the main aspects described in the previous case (Section 3.3.1) and we intend to show how the operation of the ACLs supported by the framework, could affect the performance of the controller or the ANM core. Performance refers specifically to the consumption of the resources of the integrated framework.

3.4.2 Execution & Results

For the purposes of these experiments, we generate different volumes of data based on the traffic matrix that is exploited in [27]. In the next figures, we present experimental results for different traffic demand intensities in the network and six policies (different policy index, Figure 3-5 right middle pane “ACL Policy configuration”) varying from energy efficiency to load balancing. As different policies are selected, new paths are decided and enforced to the network (updating the flow tables at the switches) through the controller. For these experiments, traffic was distributed between the following PoPs:

- Amsterdam
- Zagreb
- London
- Vienna

Average Demand Intensity (Mbps)	Number of Paths
80	39
100	43
120	43
140	48
160	39

Table 3-2: Number of paths per average demand intensity

Table 3-2 shows the number of utilized paths per average demand intensity. At that point, we examine the resources' consumption of our framework. In Figure 3-6 and Figure 3-7 the variations of the CPU values (%) of the controller with and without the use of the ACLs is depicted. As expected, the consumption of the resources did not show any major discrepancy. The differences in the CPU performance are between 2.5% and 3.6%, while for the memory are near 3%.

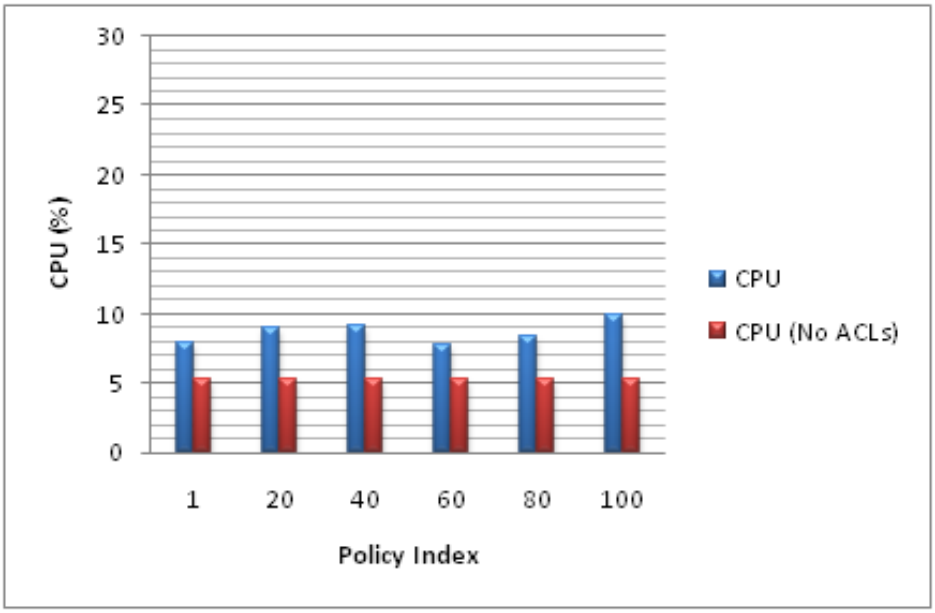


Figure 3-6: CPU consumption for the controller

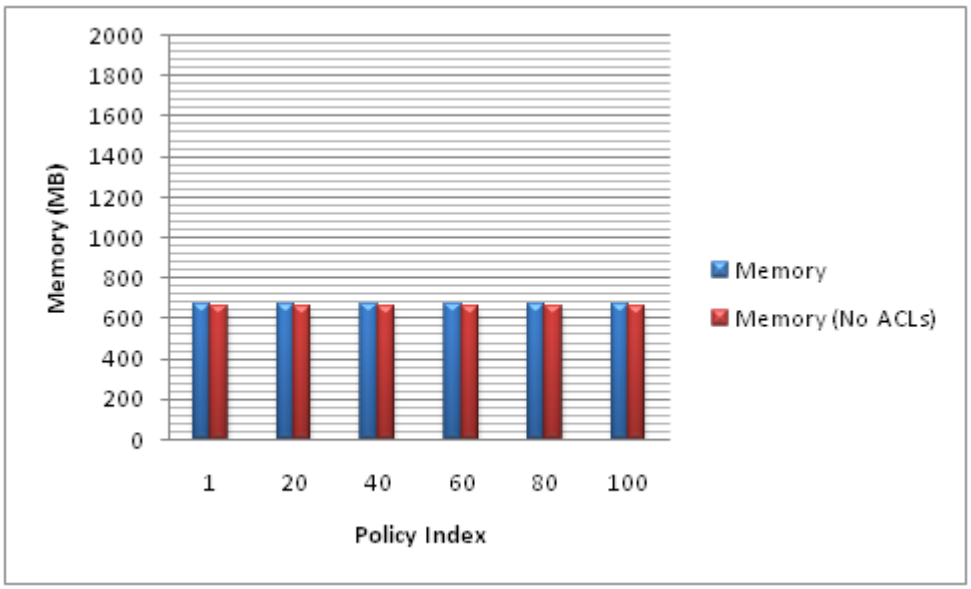


Figure 3-7: Memory usage for the controller

In addition, as presented in Figure 3-8 and Figure 3-9, the selection of different high-level policies did not affect the ANM core's resources.

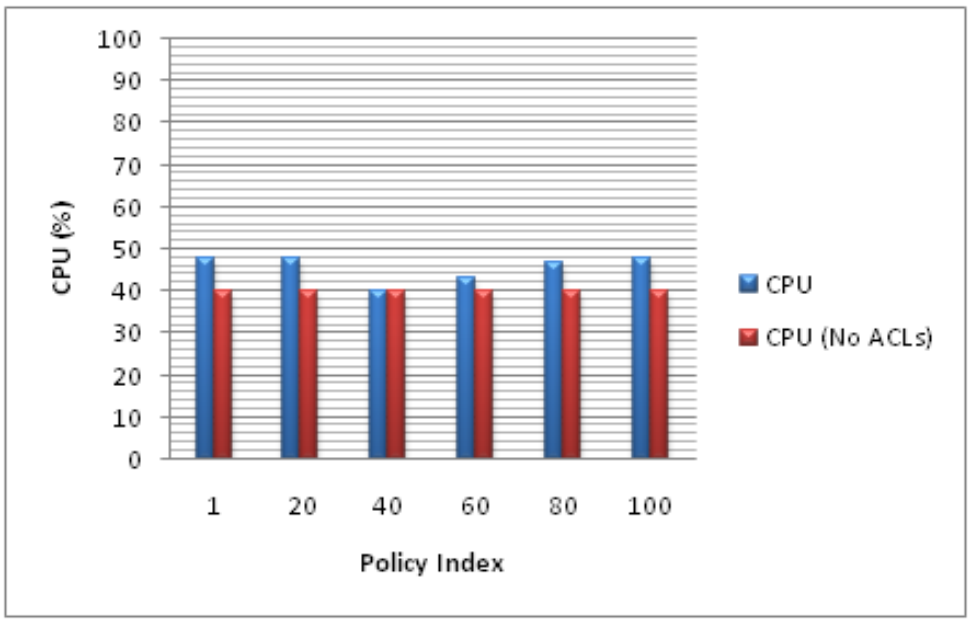


Figure 3-8: CPU consumption for the ANM core

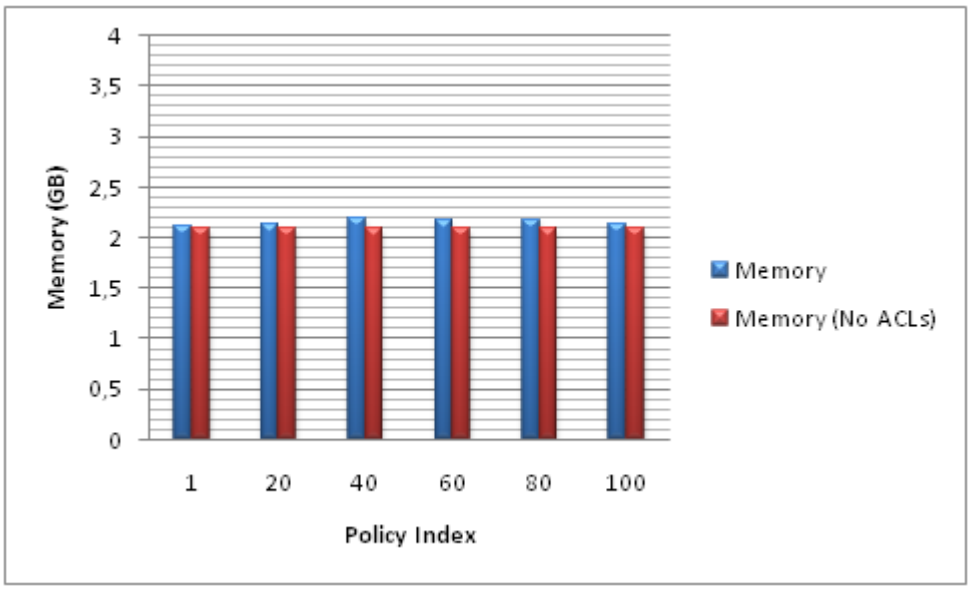


Figure 3-9: Memory usage for the ANM core

3.5 Conclusions

In this section, the first experiments with the AUTOFLOW experimentation framework and its components were presented. The conducted studies have shown that the integration of ANM and SDN allows operators to manage their networks efficiently, (un)deploying ACLs in a “plug and play” way, monitoring network elements and enforcing decisions in run-time. Particularly, we presented the interaction between the ANM core / ACLs components, the OpenFlow controller and the network elements of the GÉANT facility. Through this interworking, we managed to monitor and/or configure the elements of the core network. We should highlight that the framework allows the operator to act proactively, as he is able to monitor the changing conditions in the network in real-time (e.g. through the Metric Monitor tool). Furthermore, the operator may choose the desired high-level policy (e.g. energy efficiency, load balancing) and steer network’s operation respectively, while guaranteeing the requested services to the clients. Moreover, the framework operates efficiently even when the operator is at a remote site. Performance objectives such as CPU and memory consumption of both the controller and the ANM core remain at low levels and do not report any significant discrepancies.

The flexibility that our experimentation framework offers meets the challenges of network function generalization, performance, isolation, re-configurability and manageability of SDN that authors present in [28]. AUTOFLOW creates the conditions on top of the GÉANT SDN/OpenFlow facility for facilitating subsequent development, deployment, testing and experimentation of various ANM solutions. The framework encapsulates autonomic functions (closed control loops/algorithms) that can be embedded into legacy and future networking systems and services in a “plug and play”/”unplug and play” way, promoting the network's programmability. The required tools used for the interfacing of ANM/ACL are implemented with special consideration for reusability. Therefore the ANM Core / ACLs components have an interaction with the OpenFlow Controller in terms of

"Monitoring-Configuring" and vice-versa, in order to complete the network operations (e.g. switching, multipath, etc.) and achieve specific service goals. Therefore, the network's behavior and any parameters in general are highly managed and controlled from the operator's dashboard.

The differentiation of AUTOFLOW among the other frameworks is that it offers extended network capabilities such as autonomic network functions (e.g. Traffic Engineering, Load Prediction), network isolation and monitoring in real time for SDN/OpenFlow infrastructures. Moreover, due to the usability and the flexibility that AUTOFLOW offers, it could provide *any* autonomic network function as a service from *any* location to *any* user regardless of its location. For once more authors in [28] point out the challenges of the management and monitoring capabilities of SDN. They claim that although SDN monitoring tools can be powerful in small and medium sized network topologies, yet debugging, troubleshooting, monitoring and enforcing security compliance are very difficult tasks. OpenFlow makes this even harder due to the poor choice of the monitoring location and the statistics that should be output for the OpenFlow SDN session. In AUTOFLOW, we emphasize on the monitoring process and we strongly believe that it improves the diagnosis of network's performance by enhancing the visibility of the traffic characteristics. AUTOFLOW's monitoring tools (e.g. Metric Monitor) are able to capture the network's information and behavior, facilitating operators' management decisions.

Furthermore, it provides a proactive capability that leverages the flexibility and programmability of SDN to create elastic network monitoring that will lead in advanced configuration and troubleshooting. Thus, these tools can support service awareness by interactively mapping the flows onto high-level policies/services (energy efficiency/load balancing) and monitoring the status of these flows. Additionally, they supervise the topology discovery either of the entire network or specific slices like the slice that was assigned to us by GÉANT. In particular, although network configuration remains a difficult task for future networks, we argue that AUTOFLOW is able to enforce various high-level policies and respond to a wide range of network events in a reliable, efficient and autonomic way.

In summary, in the future networks the location of the network elements and SDN applications should be distributed physically and virtually. The user should observe, feel and expect the quality of delivery of the requested service to be reliable, efficient and flexible. Networks should be more scalable, cunningly and autonomic. AUTOFLOW contributes to this vision of future communications and services. It is an evolutionary step, paving the way towards the management of future networks and services with the use of autonomic network functions.

4 Experiment #2: Governance of Autonomic Control Loops

4.1 Introduction

On the road to the future networks and services, service providers are trying to remodel their infrastructures with high volumes of resources in terms of computing and communication. Meanwhile, the operation and the management of these infrastructures become progressively more complex, hard to manage and less cost-effective. Current operational and management systems are designed to cover different needs and it seems that they cannot fulfill the requirements of the next generation environments. The understanding of the operational and management challenges, enforce us to bring together SDN and autonomics and design a framework that will simplify the end-to-end service availability, delivery and management for the operator. Based on this approach, one of the main characteristics of AUTOFLOW is the governance of ACLs within the operator's ecosystem. Governance is described as a way for managing autonomous behaviors instead of relying on the stovepipe type of traditional network management.

Moreover, governance is tightly connected with the enforcement of policies. Typically, business level policies are defined in the highest level, that is, they express business objectives. These policies are propagated to the network where they are being transformed into lower level policies, until they finally reach the node(s) on which they are enforced in terms of low level, configuration commands. In addition, in AUTOFLOW, the role of governance is identified through the provision of a powerful and evolved interface that will be handled by human operators for targeting their business goals. Further to that, governance validates the complete instantiation and deployment of the ACLs. By having a global view of the network components, governance participates in the overall evaluation of the performance of the provided services. The operators specify their requirements at a top level to the ANM core and these requirements are translated into policies, being applied to the SDN infrastructure in an autonomic manner.

4.2 Actors, Setup, Topology

This scenario includes the measurement of some metrics regarding the availability and the performance of the ACLs. The ANM core was connected with the GÉANT Topology and we focused on the server on which the framework was running.

4.3 Storyline

In this scenario, we examine the framework's performance in terms of available, registered and activated ACLs. Firstly, as soon as the ACL is instantiated, it appears as available. Then, when the operator selects the ACL, its state changes to "registered" and it is ready for deployment. The last part is to activate (deploy) the ACL in order to begin its operation. The experiment was accomplished under the same conditions described earlier at Section 3.3.1.

4.4 Execution/Results

For the purpose of the evaluation, we initially initiate the ANM core. After some seconds the ANM core communicates with the controller and the latter appears in the upper left pane of the operator's dashboard. Then, we start increasing the number of the available ACLs by creating instances of the Core-TE ACL. We start with the instantiation of 2, 4, 10 and then continue with 30 to 40, until we stop at 50 instantiated (available) ACLs. The corresponding increase in the memory consumption is presented in Figure 4-1.

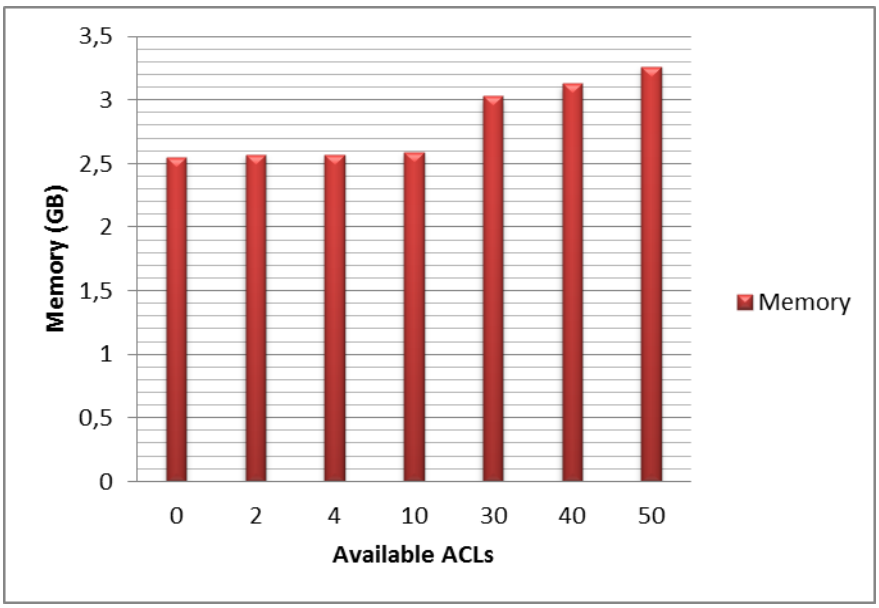


Figure 4-1: Memory usage vs. Available ACLs

Afterwards, we start the registration process. 50 ACLs are simultaneously registered during our experiment. At that point, it isn't possible to proceed to the deployment of any other ACL, so we gradually start "killing" registered ACLs until we are able to deploy one. Table 4-1 summarizes the experimental results, presenting the number of available/registered/activated ACLs and the corresponding memory consumption. No abnormal behavior is observed in the CPU consumption.

Available (Instantiated) ACLs	Registered	Activated	Memory consumption
2 ACLs	**YES**	**YES**	**2.57 MB**
4 ACLs	**YES**	**YES**	**2.57 MB**
10 ACLs	**YES**	**YES**	**2.59 MB**
30 ACLs	**YES**	**YES**	**3.03 GB**
40 ACLs	**YES**	**NO**	**3,13 GB**
50 ACLs	**YES**	**NO**	**3.26 GB**

Table 4-1: ACLs vs. memory consumption

Finally, in the context of this experiment, we examine the needed times for the transition of the ACLs between all possible states. In particular, an ACL starts from the “Available” state, where it is ready to be instantiated. Then, it is “Instantiated” and available to be “Deployed”. Afterwards, the operator may decide to “Un-deploy” it or even “Stop” it. The execution times for these transitions depend mainly on two factors; the first one is related to the ANM framework and the second one is related to each ACL (internal operation for set-up). Regarding the framework, these transitions are almost instantaneous. In general, only a few HTTP requests are required, for the exchange of the appropriate commands. Moreover, the specific ACLs that are examined (Core-TE, LP) do not show substantial effect to the needed times for their transition between the different states.

4.5 Conclusions

In this experiment, we focused on the governance of the ACLs in the context of the developed framework. Memory/CPU consumption in conjunction with the actual numbers and states of the ACLs was measured. In addition, execution times for the transition of the ACLs between different states were examined. As presented in the previous paragraph, no major limitations can be identified. Several ACLs can be successfully deployed and operated in parallel.

5 Experiment #3: Operator-driven traffic engineering in SDN/OpenFlow networks

5.1 Introduction

Internet Traffic Engineering (TE) is defined as the aspect of Internet network engineering dealing with the issue of performance evaluation and performance optimization of operational IP networks [29]. The key TE performance objectives, classified as being either traffic-oriented or resource-oriented, include the minimization of packet loss, the minimization of delay, the maximization of throughput, the enforcement of service level agreements and the efficient management of network resources [30]. TE mechanisms have been widely exploited in the past and current data networks, such as Asynchronous Transfer Mode (ATM) and Internet Protocol/ Multiprotocol Label Switching (IP/MPLS) networks. However, these past and current networking paradigms and their corresponding TE solutions are unfavorable for the next generation networking paradigms and their network management due to the need for: i) real-time reaction, ii) scalability for large amount of traffic, and iii) improved resource utilization for better system performance [31].

To overcome this ever-increasing size and complexity of network management, ANM has been proposed as an emerging solution. ANM simplifies management and control of network and services by minimizing the burden of manual operation. It is based on the distribution of multiple autonomic elements, which are composed of the monitoring, analyzing, planning and executing components, referred to as the MAPE-K model, to enable the self-adaptation to the environment changes. Autonomicity and self-adaptation are key factors for taking fast, online and reliable TE decisions.

Furthermore, SDN is an emerging technology that promises to tackle the complexity of network management and to minimize the operational costs (in the long run). The concept of SDN has been under extensive research, in particular evolving around the well-known OpenFlow protocol [32] and its supporting consortium i.e. Open Networking Foundation (ONF) [33]. SDN provides: i) centralized control of multi-vendor environments including global network information, ii) more granular network control, iii) the programmability without having to handle individual infrastructure elements, and iv) reduced complexity through automation. TE technology is of critical importance to the evolution and success of SDNs. TE mechanisms in SDN can be much more efficiently and intelligently implemented as a centralized TE system compared to the conventional approaches.

Moreover, it is worth mentioning a number of industrial TE tools for SDNs. The objective of [34] (proposed at Bell Labs, Alcatel-Lucent) is to develop an SDN deployment scheme that can adaptively and dynamically

manage traffic in a network to accommodate different traffic patterns. The authors formulate the SDN controller's optimization problem and outline a Fully Polynomial Time Approximation Scheme (FPTAS) to solve the controller's problem. B4 (designed by Google) [35] is a Software-Defined WAN for data centre to data centre connectivity. In this approach, centralized traffic engineering allocates bandwidth among competing services based on application priority, dynamically shifting communication patterns and prevailing failure conditions. B4 has enabled Google to deploy substantial cost-effective WAN bandwidth, running many links at near 100% utilization for extended periods. Software-driven WAN [36] (SWAN, proposed by Microsoft) is a system that boosts the utilization of inter-datacentre networks by centrally controlling when and how much traffic each service sends and frequently re-configuring the network's data plane to match current traffic demand. SWAN achieves high efficiency while meeting policy goals such as preferential treatment for higher-priority services and fairness among similar services.

Considering the state of the art solutions in the areas of autonomics, SDN and TE in SDN, in this experiment, we focus in the experimental validation of the performance of Core-TE ACL. In particular, this experiment aims at validating the efficient enforcement of high-level policies into the network. Such policies may vary from "Energy Savings" (ES) to "Load Balancing" (LB) and fire the execution of different actions. For instance, the fulfillment of the ES goal can be achieved by minimizing the number of active links between the core routers, whereas the LB goal can be achieved by sufficiently balancing the load between the core links. The performance evaluation is based on traffic metrics such as utilization, used port/links, average hops/path, jitter and packet loss.

5.2 Actors, Setup, Topology

The GÉANT OpenFlow facility is used for these experiments and VMs are instantiated in all 5 PoPs (Vienna, Zagreb, London, Amsterdam and Frankfurt). The offered capacity that is assigned to each of the network links that interconnect the OpenFlow switches is exploited at its whole. In our experiments, the VMs are used as traffic producers/consumers. Sources and destinations of traffic are randomly selected and the traffic (UDP traffic) that is generated using Iperf follows the traffic models that are described in [37] and [38].

5.3 Storyline

The main subject of this experiment is the operator that decides to change the network's operation dynamically according to policies and the Core-TE ACL that is responsible for the enforcement of the appropriate routing configurations. Therefore, we investigate how different policies affect network's performance and whether the proposed solution succeeds to steer network's operation dynamically on run-time. Regarding the policies, they are realized by selecting a specific value (Policy *selection*, Figure 5-1). *Selection* can take values from 1 to 100. This is a high level choice, denoting the level of aggregation/diffusion that will characterize the accommodated traffic in the network. Lower values of *selection* result in energy-efficient routing configurations, while as these values increase, the load in the network is balanced accordingly. The objective of the Core-TE mechanism is to translate this policy into a compliant network configuration and to enforce this decision on the network through the northbound API of the Floodlight controller.

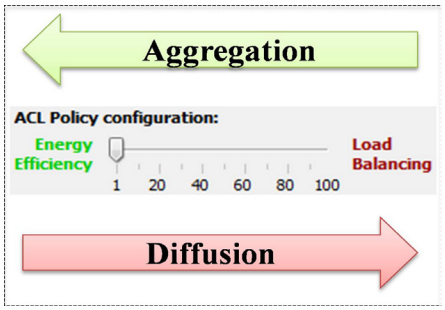


Figure 5-1: Policy selection

5.4 Execution/Results

5.4.1 Case 1

Figure 5-2 depicts the utilization levels (transmitted bits per second) of the ports of the switches in the network during variations in the selected policy. In particular, at the beginning and at the end of the experimentation period, we request for the most energy-efficient routing configurations (*selection* equal to 1), while in the intermediate period the traffic is balanced among several links (diffusion of traffic). Therefore, as depicted in this figure, for the energy efficient configurations, there exist fewer utilized ports with higher utilization levels contrary to the load balancing period, where there exist more utilized ports but with lower utilization levels. Furthermore, we should highlight that the transition between different policies is extremely time-efficient (a few seconds only), validating apart from the performance and the online behavior of the TE mechanism, the efficiency of the designed ANM/SDN framework. Figure 5-2 is derived from the monitoring tool (Metric Monitor) that exploits the northbound API of the controller in order to acquire live measurements from the network.

Figure 5-2: Transmitted bits per second for the two high-level policies (energy efficiency and load balancing), experimenting over the GÉANT OpenFlow facility.

The experimentation process continues with variations in the traffic demand intensity of the generated flows in the network. For validity purposes, several test cases are examined, resulting in different number of paths and volumes of traffic transmitted between the GÉANT hosts. The following table (Table 5-1) presents the total number of paths that were activated between the GÉANT hosts for the relevant demand intensity based on the selected traffic matrix.

Average Demand Intensity	Number of Paths
20	39
40	43
60	43
80	48
100	39
120	48
140	51

Average Demand Intensity	Number of Paths
160	32
180	40
200	35

Table 5-1: Total number of utilized paths

In all cases, the maximum links utilization is minimized for high values of selected policies (load balancing), while the number of the utilized links is increased. This is expected, as more network elements are activated, in order to accommodate the diffused traffic in the network. Figure 5-3 presents one of these test cases, where the maximum utilization is examined for the different demand intensities, as well as for six different selected policies. It is evident that the mechanism reduces the maximum level of utilization in the network even in extreme conditions of high traffic. As depicted, the mechanism succeeds to reduce the levels of maximum links utilization almost between 45-60%.

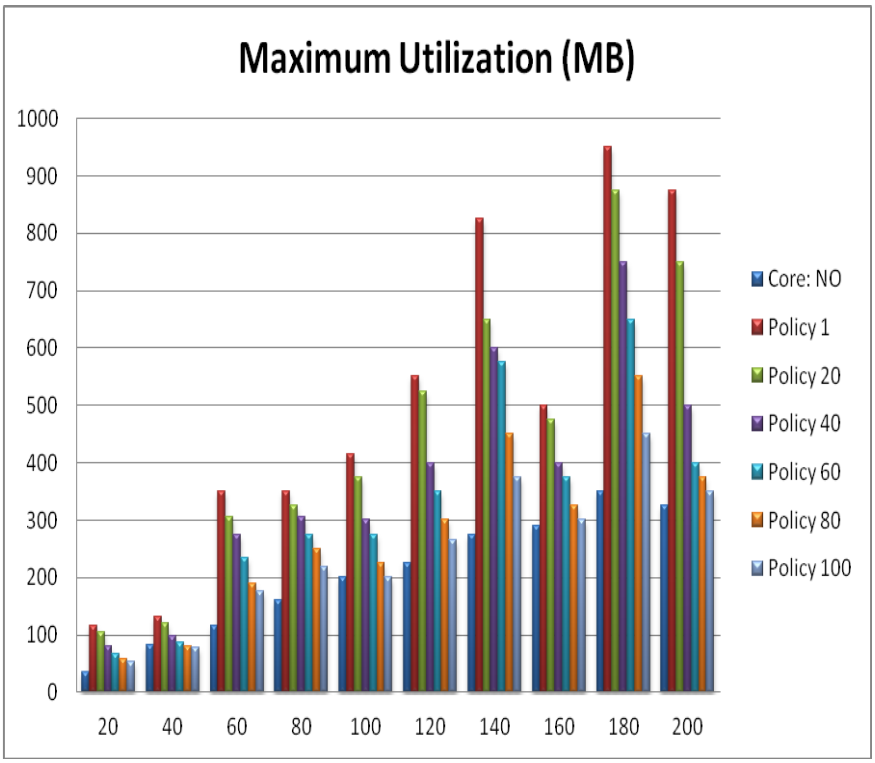


Figure 5-3: Maximum links utilization for different values of selected policies and traffic demand intensity, experimenting over the GÉANT OpenFlow facility.

In our study, we identify packet loss and jitter as important performance indicators of network stability. Indeed, in both traditional and SDN infrastructures, packet loss and jitter values are considered as critical success factors. For this purpose, in our experiments, we examined these metrics too. Figure 5-4 and Figure 5-5 present experimental results regarding packet loss and jitter for different demand intensities, as well as for six different selected policies. Packet loss is a critical issue in network environments. Packet loss generally occurs due to problems at the physical layer caused by congestion on network links and interfaces. The most common scenario for packet loss is when the queues of the intermediate network elements are full of packets ready to be transmitted and there is no additional resources e.g. memory for additional packets to be handled. The elements with full queue have no choice but to drop the packet. Authors in [39], [40] refer to the relationship between packet loss and network congestion. Packet losses in the network due to congestion (or any other reason) can significantly harm data quality during transmission. Therefore, considerable research has been conducted to understand and avoid packet loss in data networks.

In this scenario, we examined the behavior of the network during the transition from the energy efficiency policy that aggregates the traffic to the load balancing policy that diffuses the traffic and results to low level volumes per link. Eventually, we managed to reduce the resulting dropped packets due to our policy. As expected, the TE mechanism managed to reduce the packet loss rate near to 50%, while traffic was diffused due to the different values of the *selection* index (1-100). In traditional TE schemes, when a congested rendezvous point in a path is responsible for frequent packet losses, the congestion sustains as well as the packet loss until new routes for the majority of the traffic are manually established by the administrator. This situation indicates inefficient network management and degraded network performance. We use multipath routing for congestion and packet loss avoidance [41] and we contribute to this area with the exploitation of autonomics.

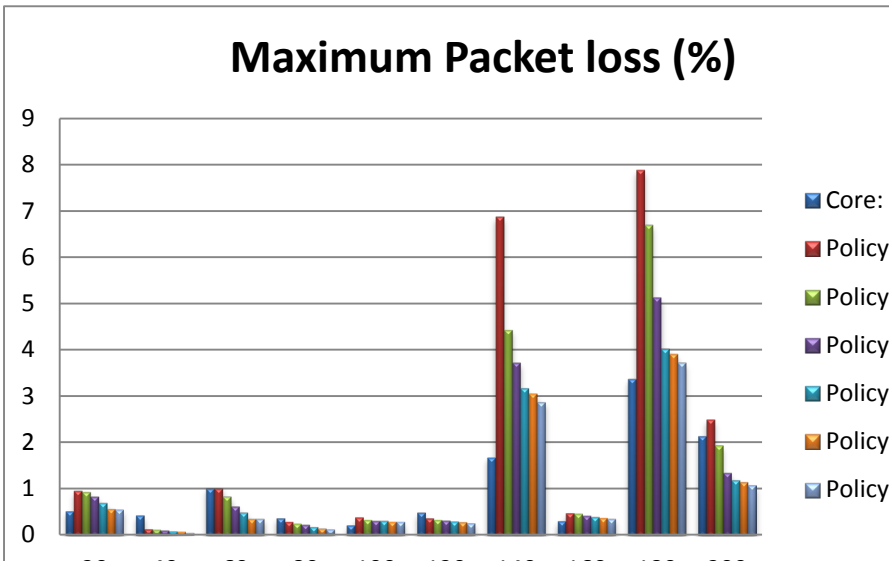


Figure 5-4: Maximum packet loss

Furthermore, we examined another important parameter of the quality transmission of data/applications/services. Jitter is critical to the network operation in maintaining consistent data rates. Hence, all vendors in market use jitter buffers on their network devices to smooth out changes in arrival times of data

packets that contain real-time sensitive data e.g. voice, and video. In [41], jitter buffers are only able to compensate for small changes in latency of arriving packets. When the arrival time of subsequent packets increases beyond a specific threshold, a jitter buffer underrun occurs. During jitter buffer exhaustion, there are no packets in the buffer to process for a specific stream or flow. In addition, when traffic manipulation is performed through unequal paths with different latency values, jitter increases. In [42] researchers argue that for a multipath environment with equal paths the value of jitter for different throughput circumstances is roughly the same, but it will increase when the speed of the paths are different. Jitter increases due to the use of higher number of paths from the sender to the receiver. In Figure 5-5, the maximum jitter rate in comparison with the different demand intensities is depicted.

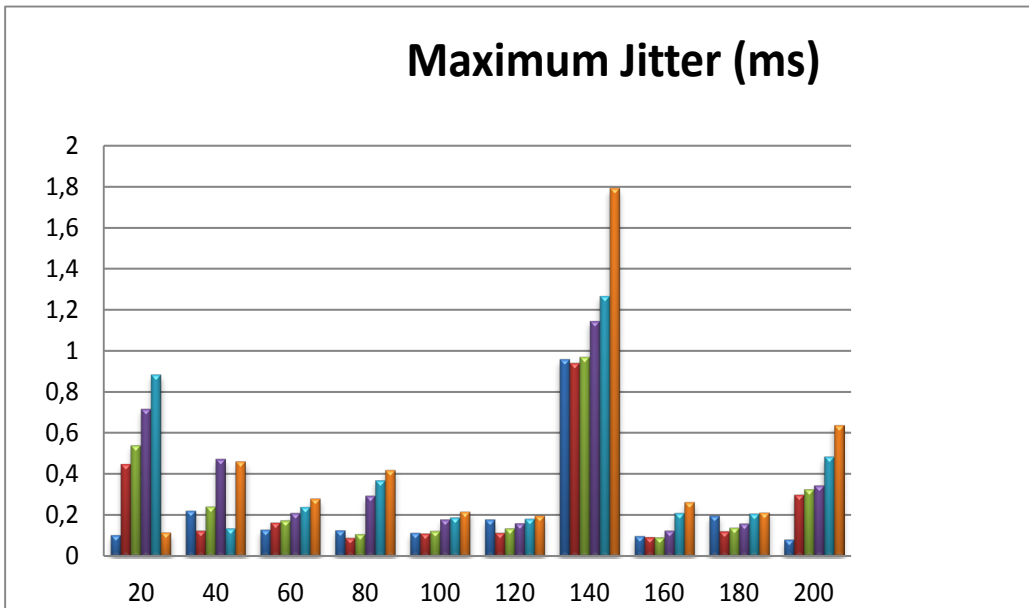


Figure 5-5: Maximum jitter

From the above figure we have not extracted a specific result, as it seems that jitter's behavior is independent of the demand intensities that were used. Thus in the figure below (Figure 5-6) we isolate and compare jitter per specific demand intensity for the six mentioned policies. We present that during our policy transitions - from Load Balancing to Energy Efficiency - there is a notable reduction on jitter's value almost up to 50%. Finally, Figure 5-7 reports the average number of hops per path during the data exchange for each policy.

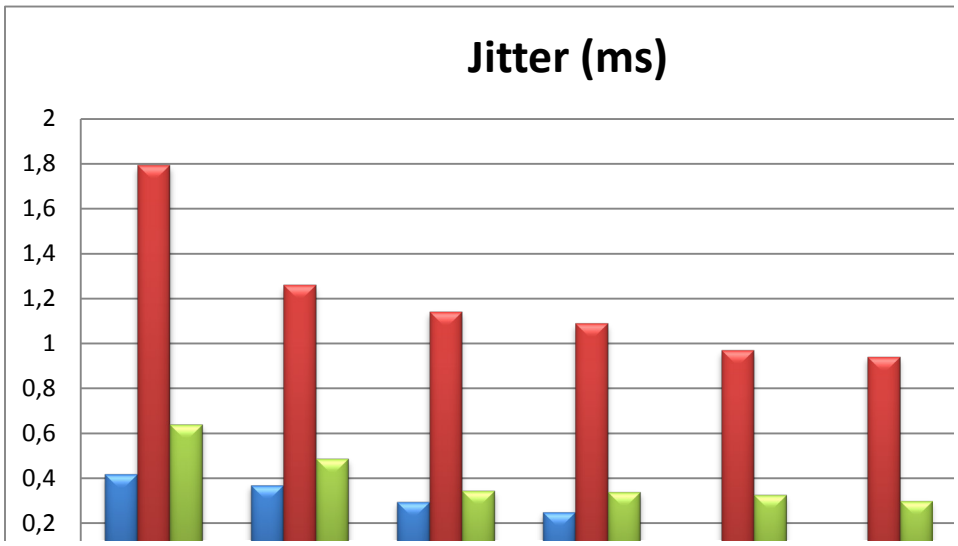


Figure 5-6: Jitter per average demand intensity (ADI)

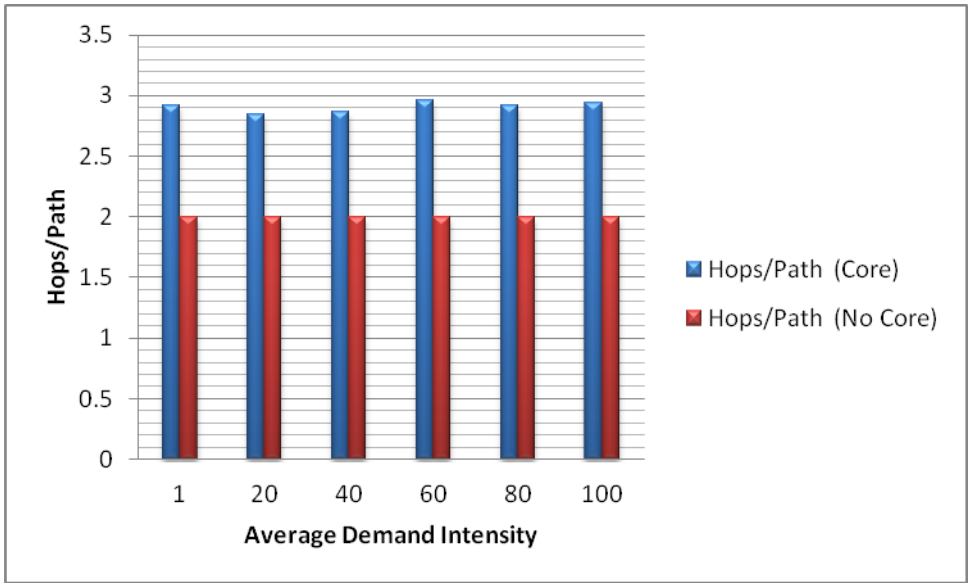


Figure 5-7: Average number of hops per path

5.4.2 Case 2

In this experiment, we evaluate the performance of the Core-TE ACL under variations in the traffic requests during a day (low/high traffic hours). Figure 5-8 depicts the maximum number of packets per port in the network, which is a metric that corresponds to the network's maximum utilization level of links. In this case, we used a

different traffic matrix based on distinct demand intensities during daytime. The effect of the two high-level policies and the corresponding different routing configurations in the network's consumed power can be indirectly evaluated through the number of the activated ports. The energy efficient routing configurations allowed the reduction of the consumed power as half of the ports were actually utilized in comparison to the load balancing routing configurations. Contrary, the maximum number of packets per port in the network, which is a metric that corresponds to the network's maximum utilization level of links, was significantly reduced due to the use of the load balancing configuration.

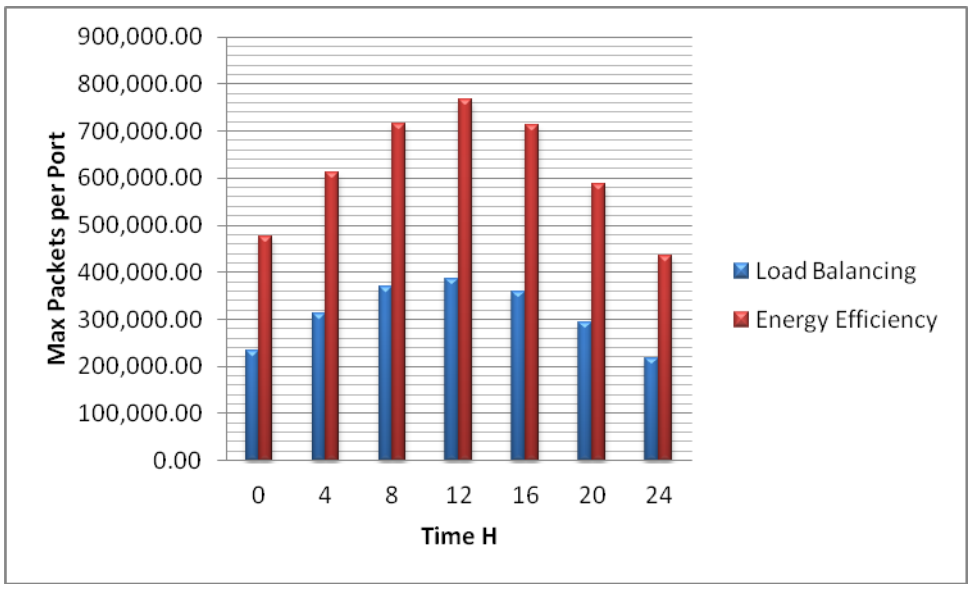


Figure 5-8: Maximum number of packets per port (#) for different times during day

Apart from utilization of links and power consumption, two other performance metrics were also examined in our experiments. Figure 5-9 presents the maximum jitter variation for the various flows that were traversing the network. As can be observed, the energy efficient configurations simplified routing of traffic and reduced this value. Furthermore, Figure 5-10 depicts the packet losses for the same experiments. The load balancing routing configurations facilitated the decrease of the percentage of packet losses in the network by distributing the load to multiple paths and minimizing the probability of network congestion. During multipath traffic sharing jitter increases, but not to an extent that it might affect the quality of data exchange.

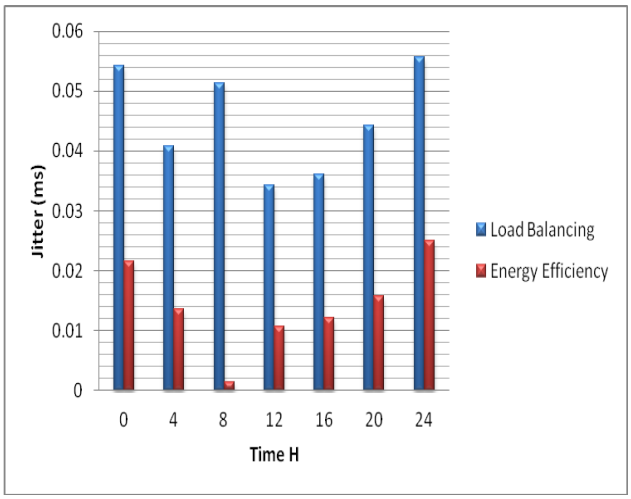


Figure 5-9: Jitter variation (milliseconds) for different times during day

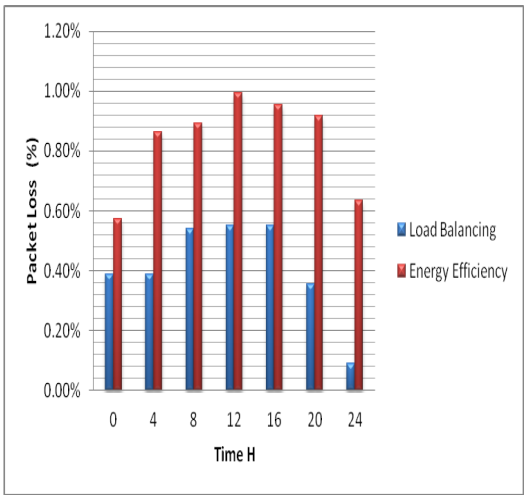


Figure 5-10: Average packet loss (%) for different times during day

5.5 Conclusions

In this section, we identified the need for advanced TE solutions to overcome the complexity of network management. Accordingly, we exploited the developed experimentation framework that combines ANM and SDN and deployed the Core-TE ACL. This autonomic mechanism enforces routing configurations to the network through the northbound API of the Floodlight controller. The conducted experimentation validated the mechanism's performance and its ability to steer network's operation dynamically on run-time, following the operator's high-level policies (e.g. energy efficiency, load balancing). In particular, the experimentation demonstrated that Core-TE succeeds to establish optimal paths to forward traffic in the network, complying with the operator's policies. For the energy efficient configurations, there exist fewer utilized ports with higher utilization levels contrary to the load balancing ones, where there exist more utilized ports but with lower utilization levels. Jitter variations and packet losses were also examined in this experiment for different policies and traffic demands. The load balancing routing configurations facilitated the decrease of the percentage of packet losses in the network by distributing the load to multiple paths and minimizing the probability of network congestion. Contrary, the energy efficient configurations simplified routing of traffic and reduced jitter variation.

6 Experiment #4: Load prediction in SDN/OpenFlow networks

6.1 Introduction

In this experiment, we attempt to investigate whether the ANM and SDN/OpenFlow technologies can be used for short prediction of network load, with the ultimate purpose of using the acquired knowledge of future traffic during the process of decision-making on routing and traffic engineering directly by the network in real time without any human intervention.

By providing valuable information in advance, short-term network load forecasting aims at improving the efficiency of the network and facilitating the real-time management and control of the network functions. Extended research in time-series forecasting has been carried out in the recent years, examining mainly the application of Artificial Neural Networks (ANNs) as prediction tools for modelling non-linear systems. Supervised learning algorithms and regression-based techniques such as Backpropagation, Support Vector Machine and hybrid models have been extensively used for years for the prediction of future electric load demand, monthly river flow, network load etc. [43], [44]. Most cases focus on pre-trained ANNs with historical data, while there are some approaches which use unsupervised techniques such as Kohonen maps and clustering algorithms in order to perform short or long-term forecasting [45], [46] and [47].

In order to carry out the experiment, we implemented LP ACL, an autonomic mechanism which utilizes machine learning techniques to investigate the possibility of continuous forecasting of the future network load based on the monitoring of current network load established by common traffic patterns. Load forecasting is implemented in real time for each source-destination pair using neural networks trained with the Backpropagation algorithm. For each traffic model used, the predicted values are compared to the actual load values and the corresponding diagrams for indicative number of source-destination pairs are presented.

The main objective of the experiment is to evaluate the behavior of the real-time load prediction algorithm based on specific traffic models.

6.2 Actors, setup, topology

Load prediction is implemented in LP Autonomic Control Loop. LP ACL is a software component which is deployed in the ANM framework. For the entire duration of the ACL deployment the four consecutive methods of Monitor, Analyze, Plan and Execute are executed repeatedly, referred to as MAPE loops. All experiments are carried out by generating real traffic on the GÉANT OpenFlow physical testbed. The GÉANT OpenFlow facility consists of five Points of Presence (PoPs) located in Amsterdam, Frankfurt, London, Vienna and Zagreb respectively. An OpenFlow software switch and a Virtual Machine on a physical server are located in each PoP. The five OpenFlow switches are interconnected in a full mesh with 1Gbps links. The network controller is located in a Virtual Machine in Frankfurt. Hosts are accessible through the Internet via SSH on their public IP addresses.

PoP Host VM	Public IP address	Private IP address
Amsterdam	62.40.110.35	192.168.1.1
Frankfurt	62.40.110.71	192.168.1.2
London	62.40.110.137	192.168.1.3
Vienna	62.40.110.3	192.168.1.4
Zagreb	62.40.110.101	192.168.1.5

Table 6-1: Topology set-up

Real traffic is generated between the five hosts, which correspond to a total of twenty (20) Source-Destination pairs (SD Pairs), by running Iperf scripts on each Virtual Machine. For this particular experiment certain traffic scripts were selected to reflect the two different traffic models:

- Traffic Model 1 (TM1): random traffic demand with a uniform distribution with different Average Demand Intensity (ADI). For the specific experiment we utilized TM1 with low, medium and high average load intensity of 40, 280 and 400 Mbps respectively.

- Traffic Model 2 (TM2): traffic demand between source and destination nodes is variable in time according to a sinusoidal function.

In all the experiments described below, monitoring and load prediction are performed *per SD Pair*.

For this particular experiment a two-layer neural network with five neurons in the input layer has been implemented for each source-destination pair. Each neural network is trained using a standard Backpropagation algorithm with a bipolar sigmoid activation function. The output layer consists of one neuron.

The iterative forecasting, according to which the predicted values are used as inputs for the next forecasts, is used as a method for multi-step-ahead prediction.

The algorithm has two discrete phases, the training and the prediction phase. During the training phase a scenario-specific set of real load values per SD Pair is presented to each neural network as the training set. In each epoch of the algorithm, after samples of the training set are presented to the neural network, each neuron's weights are adjusted and the summary mean square error for all neurons is calculated.

During the prediction phase, the already trained neural networks, after being presented with input vectors of 5 sequential network load values per SD Pair (equal to the number of neurons in the input layer), they calculate the vectors of the estimated future load values. The prediction horizon i.e. the size of the prediction vectors varies according to the specific configuration of each experiment, as described in the following sections.

In order to demonstrate the behavior of the Backpropagation algorithm regarding its forecasting capabilities, we tested two different experiment scenarios.

6.3 Scenario 1 – Online forecasting

6.3.1 Storyline

The goal of the first experimentation scenario is to evaluate the ability of the ACL to recognize short-term network patterns in real time without previous knowledge, in order to use them for future prediction.

Initially, traffic is generated according to each traffic model between the 5 PoPs and, at the same, time LP ACL is deployed in the ANM framework.

The two phases of the prediction algorithm (training and prediction phase) are executed consecutively in each MAPE loop repeatedly throughout the lifecycle of the ACL. During the training phase a set of 50 real network load values (measured in Kbps) for each SD Pair are recorded each second by monitoring the network traffic for 50 seconds and used to train each neural network. The size of the training sets (50 values corresponding to 50 seconds of monitoring) was chosen after experimenting as a trade-off in order to minimize the effects of the time-consuming training phase on the runtime efficiency of the prediction phase.

During the prediction phase, an input vector of the next 5 real load values per SD Pair is presented to the trained neural networks, after monitoring the actual network traffic for 5 seconds. Each network produces a prediction vector of 55 estimated future load values per SD Pair. The 55-length prediction vector corresponds to the duration of the next monitoring period (50 seconds of monitoring to form the 50-value training set and 5 seconds of monitoring to form the 5-value input vector to be used in the next prediction cycle).

Actual and predicted load values are recorded in two CSV files for reasons of comparison and result evaluation, as well as possible utilization by other ACLs.

6.3.2 Execution

In this experiment the prediction algorithm was evaluated in four cases with real traffic based on the traffic models described in Section 6.2. To further evaluate the behavior of the Backpropagation algorithm in the context of each traffic model, each experiment case was also executed with different parameters of Learning Rate and Momentum.

Execution steps are as follows:

- The ANM Core component runs and the AUTOFLOW ANM Application starts.
- The LP ACL runs and appears in the right up pane of the ANM Application.
- Real network traffic is generated by running the Iperf scripts of the specific traffic model on each GÉANT PoP.
- The LP ACL is deployed in the ANM framework.
- After 5 minutes of deployment time, two CSV files are updated with the actual and the predicted load values per SD Pair.

Upon completion of each experiment case, the time-series graphs comparing the actual to the predicted traffic for each source-destination pair are drafted. Indicative number of graphs is presented in the following evaluation section.

6.3.3 Results

Case 1:

- Traffic model: 1
- Average Intensity: 40

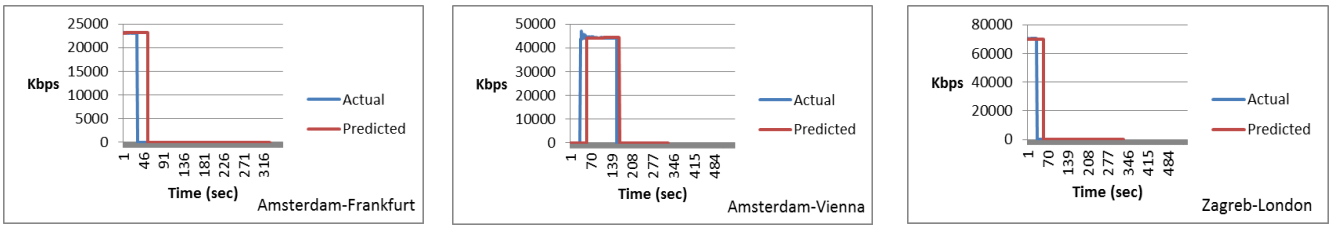


Figure 6-1: Actual and predicted traffic for 3 SD Pairs with real-time training with TM1-ADI 40 (Learning rate: 0.05, Momentum: 0.9)

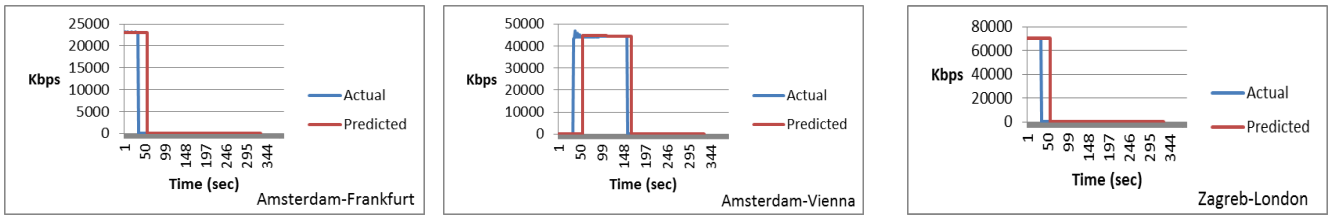


Figure 6-2: Actual and predicted traffic for 3 SD Pairs with real-time training with TM1-ADI 40 (Learning rate: 0.01, Momentum: 0.6)

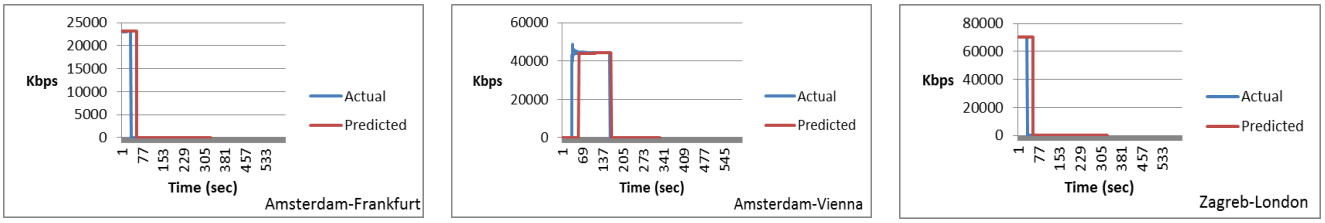


Figure 6-3: Actual and predicted traffic for 3 SD Pairs with real-time training with TM1-ADI 40 (Learning rate: 0.01, Momentum: 0.0)

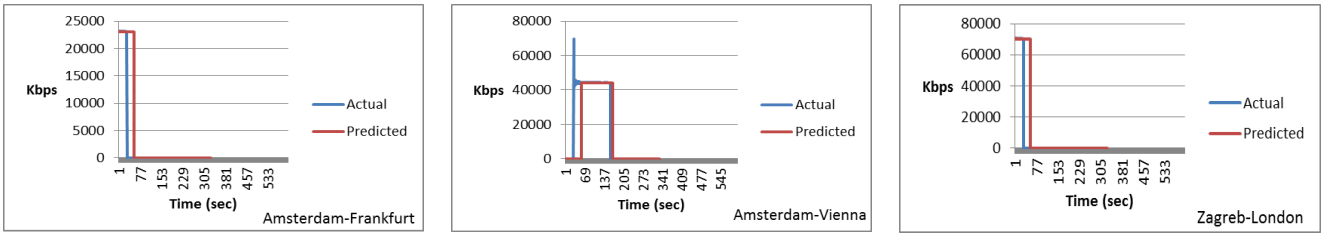


Figure 6-4: Actual and predicted traffic for 3 SD Pairs with real-time training with TM1-ADI 40 (Learning rate: 0.1, Momentum: 0.0)

Case 2:

- Traffic model: 1
- Average Intensity: 280

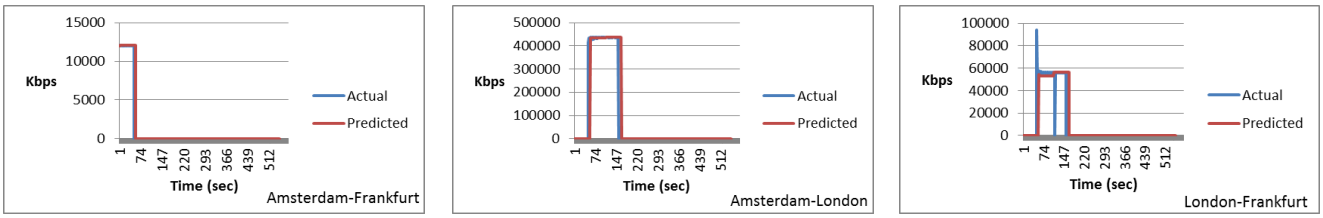


Figure 6-5: Actual and predicted traffic for 3 SD Pairs with real-time training with TM1-ADI 280 (Learning rate: 0.05, Momentum: 0.9)

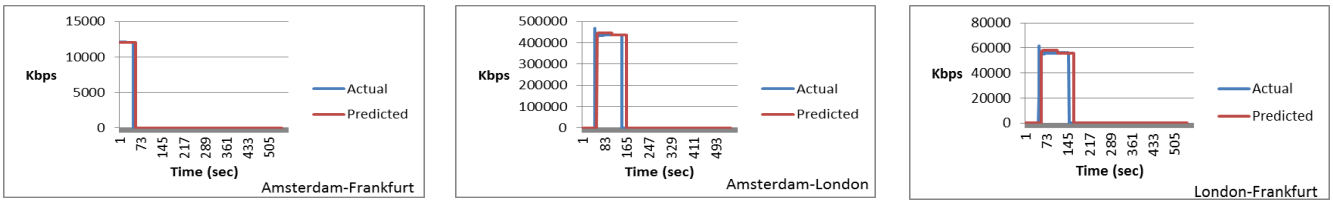


Figure 6-6: Actual and predicted traffic for 3 SD Pairs with real-time training with TM1-ADI 280 (Learning rate: 0.01, Momentum: 0.6)

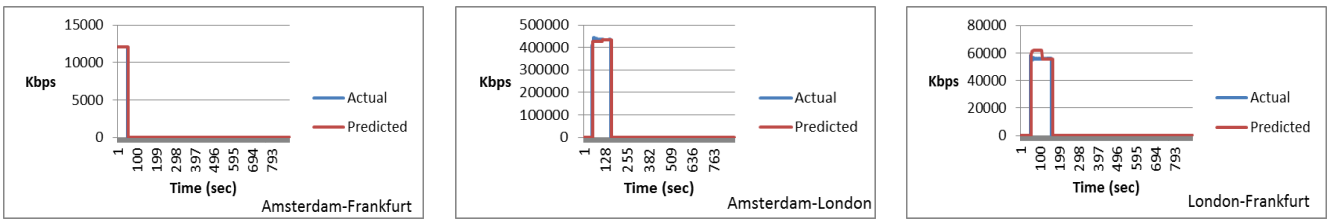


Figure 6-7: Actual and predicted traffic for 3 SD Pairs with real-time training with TM1-ADI 280 (Learning rate: 0.01, Momentum: 0.0)

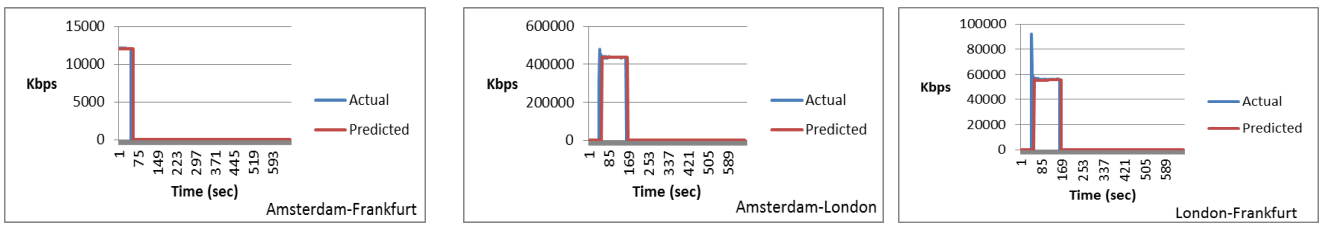


Figure 6-8: Actual and predicted traffic for 3 SD Pairs with real-time training with TM1-ADI 280 (Learning rate: 0.1, Momentum: 0.0)

Case 3:

- Traffic model: 1
- Average Intensity: 400

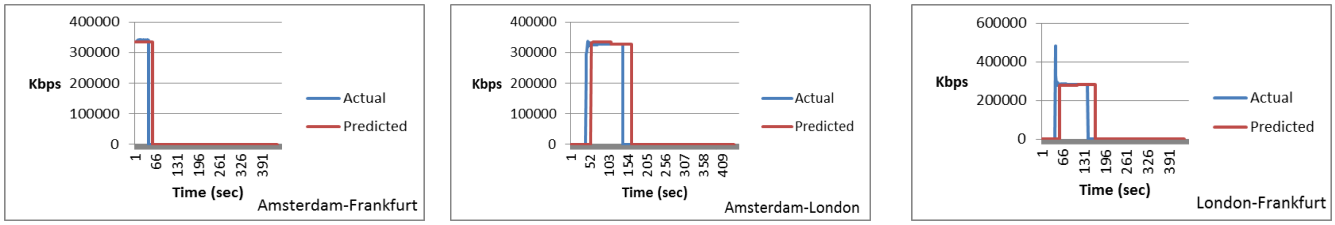


Figure 6-9: Actual and predicted traffic for 3 SD Pairs with real-time training with TM1-ADI 400 (Learning rate: 0.05, Momentum: 0.9)

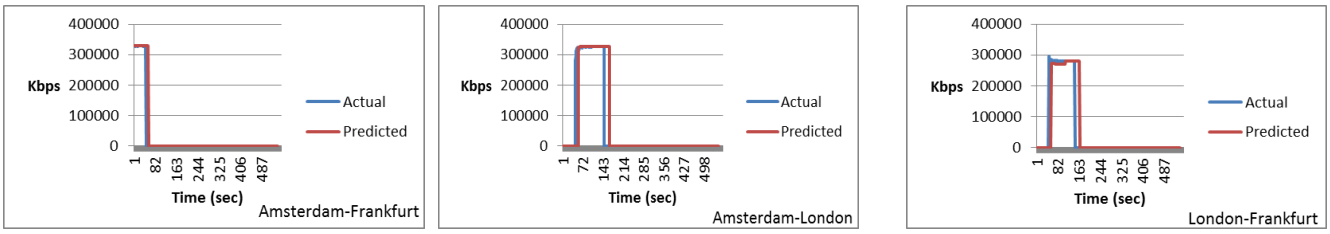


Figure 6-10: Actual and predicted traffic for 3 SD Pairs with real-time training with TM1-ADI 400 (Learning rate: 0.01, Momentum: 0.6)

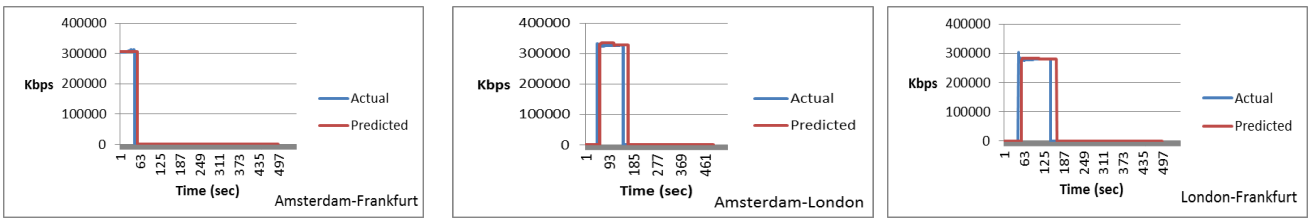


Figure 6-11: Actual and predicted traffic for 3 SD Pairs with real-time training with TM1-ADI 400 (Learning rate: 0.01, Momentum: 0.0)

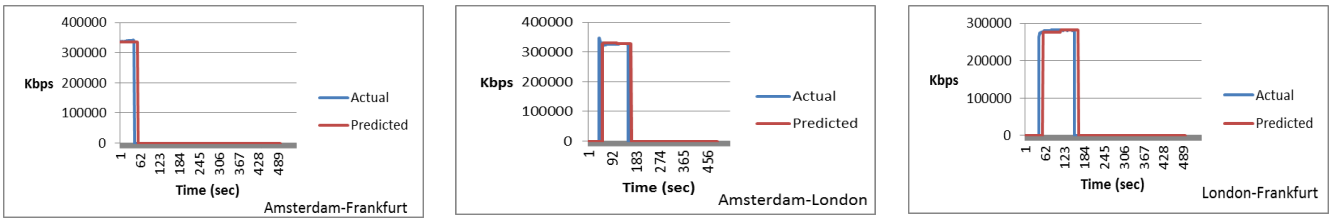
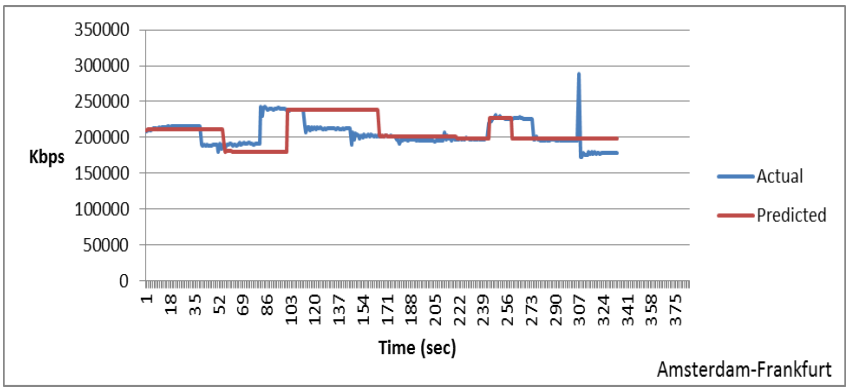
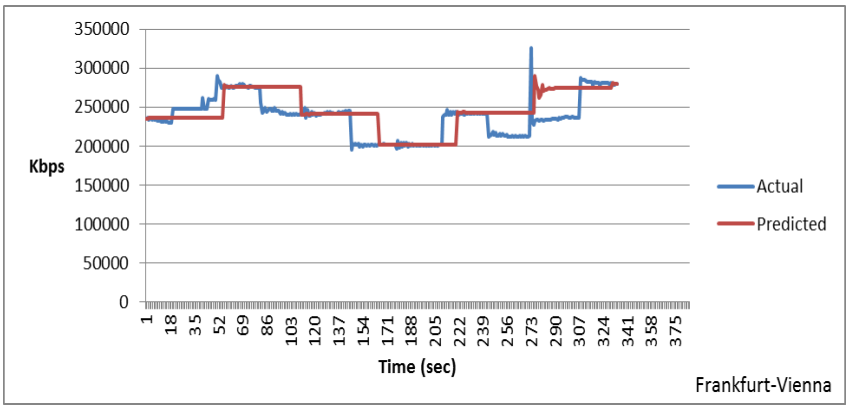


Figure 6-12: Actual and predicted traffic for 3 SD Pairs with real-time training with TM1-ADI 400 (Learning rate: 0.1, Momentum: 0.0)

Case 4:

- Traffic model: 2





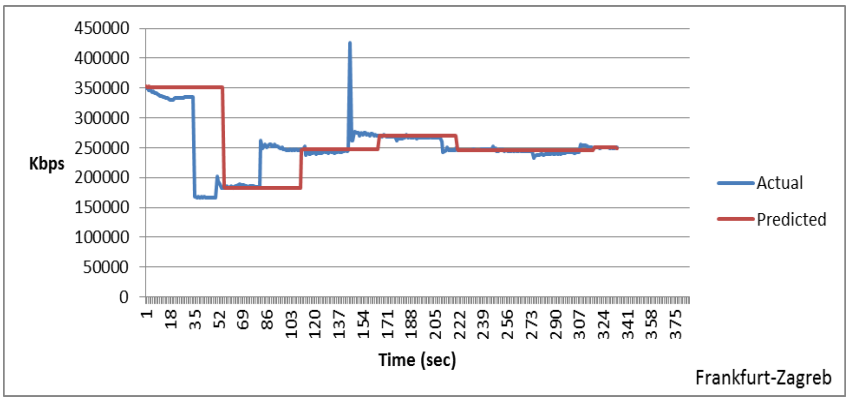
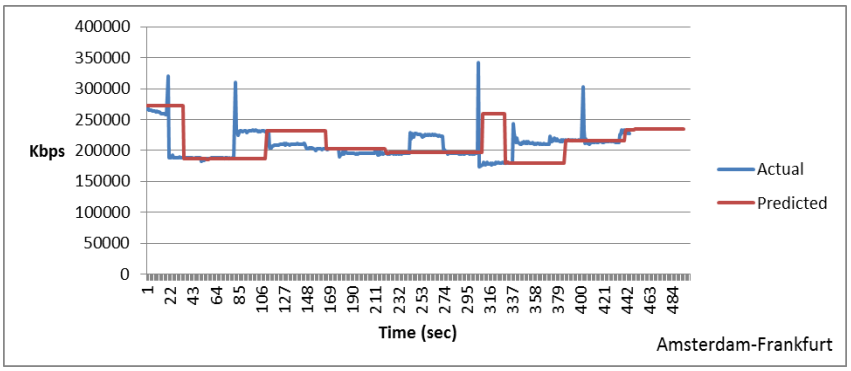
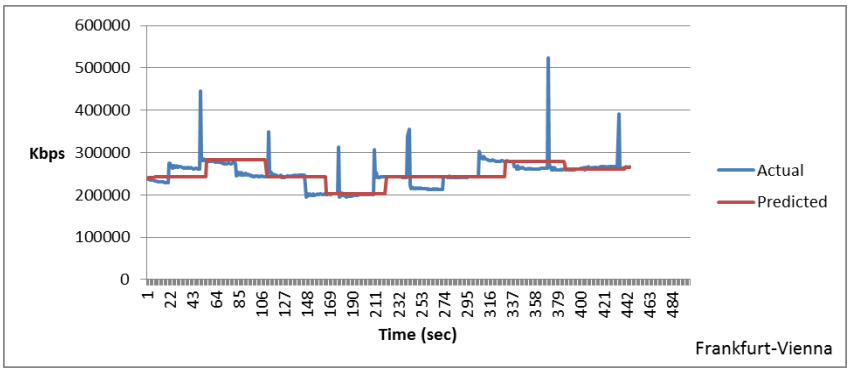


Figure 6-13: Actual and predicted traffic for 3 SD Pairs with real-time training with TM2 (Learning rate: 0.05, Momentum: 0.9)





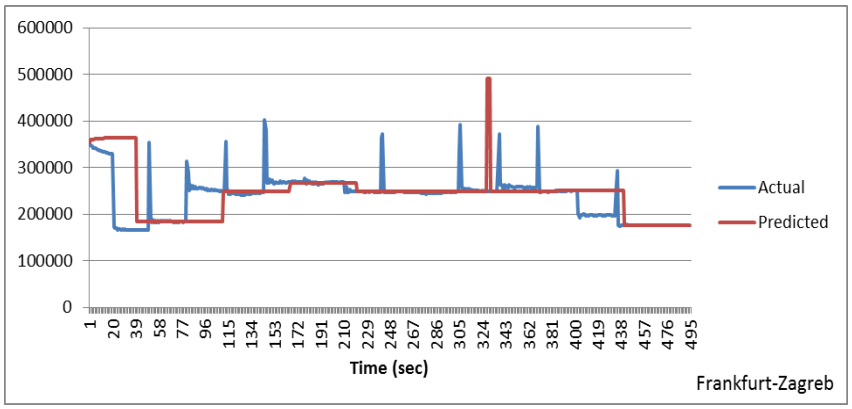
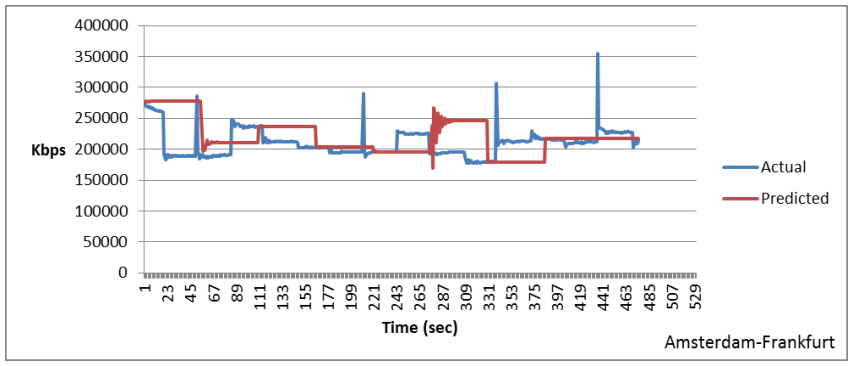
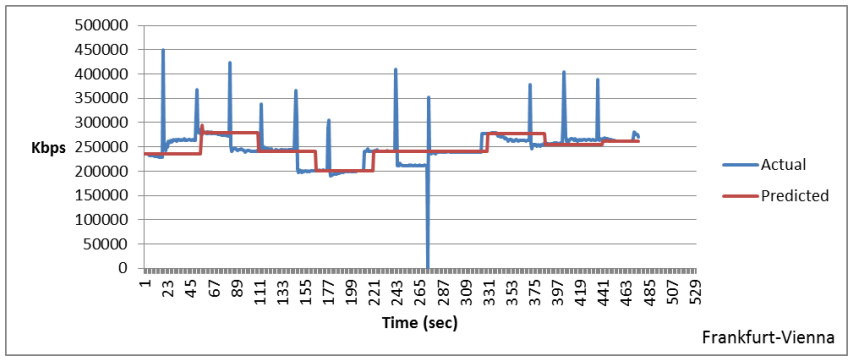


Figure 6-14: Actual and predicted traffic for 3 SD Pairs with real-time training with TM2 (Learning rate: 0.01, Momentum: 0.6)





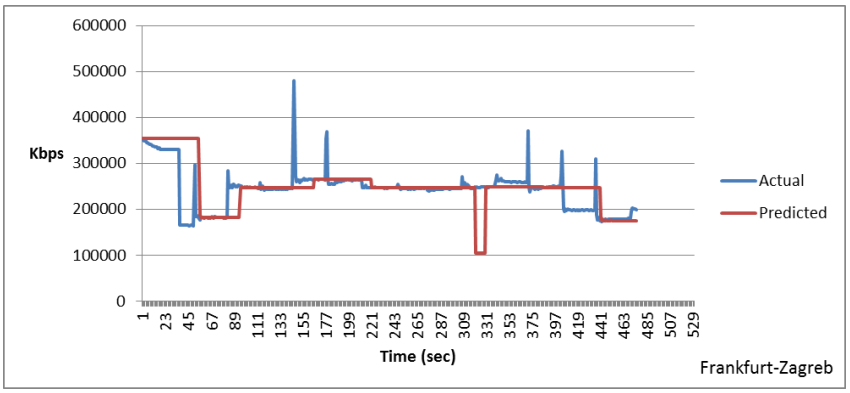


Figure 6-15: Actual and predicted traffic for 3 SD Pairs with real-time training with TM2 (Learning rate: 0.01, Momentum: 0.0)

The experimental results indicate that the load intensity of the network traffic has little to no effect on the behavior of the prediction algorithm. Moreover, tuning Backpropagation parameters such as the learning rate and the momentum has no significant effect on prediction accuracy in the specific context.

The objective of the implemented algorithm is to predict the general trend of network traffic in a long period of time (at least as long as the LP ACL is running) by utilizing small periods of traffic as training sets for the neural networks. Nevertheless, in Case 4, in which Traffic Model 2 was tested, apart from the less accurate results compared to the results of Traffic Model 1, we observe that there is a time shift of the predicted time series. In other words, prediction is delayed as the result of the variation of the real traffic produced by the specific model, which forms a pattern that spans multiple training phases. Since Backpropagation is a supervised learning algorithm, a change in load which occurs at the end of the training phase will be realized by the algorithm after the completion of the current training and prediction phases, i.e. during the next prediction cycle. In worst case scenarios short-term load variations will not be detected by the algorithm at all.

In most of our experiments the length of the time shift was half the duration of the training phase (25 seconds). Increasing the length of the training set would prolong the training phase without any significant effect on real-time prediction of network load that varies over time. On the other hand, it would prove useful in case of network monitoring and prediction based on known network traffic patterns observed in predefined time periods (days or weeks). This case is examined in Section 6.4 (Scenario 2) below.

The deviation of the numeric values of the actual and the predicted load is limited. Given that the network load is measured in Kbps, this deviation is actually insignificant.

Short-term bursts or instantaneous load variations cannot be estimated by the algorithm.

6.4 Scenario 2 – Forecasting by pre-trained networks

6.4.1 Storyline

The goal of the second experimentation scenario is to investigate ways to improve the algorithm's accuracy and performance, as well as to optimize the ACL's efficiency, based on the assumption that network load is not subject to random high variations, but it follows some general trends throughout certain time periods (i.e. certain weekdays or certain dates of a year).

Therefore, a different approach is attempted in the second scenario, where real-time prediction is performed by pre-trained neural networks.

For scaling reasons, we assume that the four traffic models described in Section 6.2 reflect four different traffic patterns that occur within four different days of the week. Prior to forecasting, the LP ACL is deployed four times to train neural networks with the four above-mentioned traffic scenarios (TM1 with three different average load intensities and TM2) after five minutes (300 seconds) of traffic monitoring with training sets of 300 load values per SD Pair. No changes have been made to the algorithm other than increasing the training set size to 300.

For each traffic model the training phase took place within a different day of the week. The state of each neural network after final weight adjustment has been serialized and saved in a binary file named after the specific day (one file for each traffic model, representing the state of the neural network for each SD Pair at the corresponding day).

The LP ACL was deployed four times within the same days of the next week. In order to evaluate the forecasting algorithm and validate the prediction results against real load values, the corresponding traffic for each model is being generated before each deployment of the LP ACL.

Upon detection of the current day, the neural network state is de-serialized and loaded from the file that corresponds to the specific day. This technique allows for the ACL to perform immediate prediction based on already known network traffic patterns, completely omitting the training phase. The already trained neural networks continuously calculate prediction vectors of 5 future load values, after being provided with input sets of 5 actual traffic values (5 seconds of monitoring the network), throughout the deployment duration of the ACL.

As in the first scenario, actual and predicted load of the network is recorded in CSV files for comparison, evaluation and future exploitation.

6.4.2 Execution

Training

- The ANM Core component runs and the AUTOFLOW ANM Application starts.

- The LP ACL runs and appears in the right up pane of the ANM Application.
- Real network traffic is generated by running the Iperf script of the specific traffic model on each GÉANT PoP.
- The LP ACL is deployed in the ANM framework.
- After 5 minutes of training, a file named after the specific day is generated, containing the serialized state of the neural network for each SD Pair.

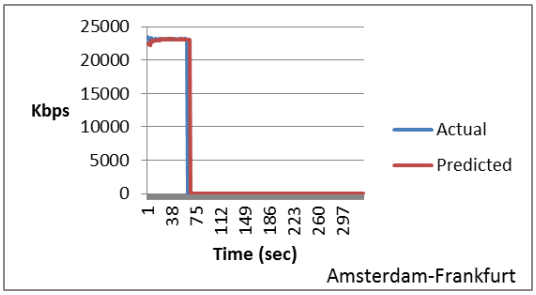
Prediction

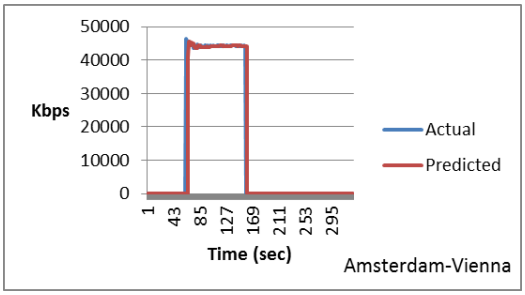
- Traffic is generated according to the traffic model used during the same day.
- The LP ACL is deployed in the ANM framework.
- After 5 minutes of execution, the actual and predicted load values for each SD Pair are recorded in two CSV files.

Upon completion of each experiment case, the time-series graphs comparing the actual to the predicted traffic for each source-destination pair are drafted. Graphs for indicative number of SD Pairs are presented in the following section.

6.4.3 Results

In each experiment case the Backpropagation algorithm was executed with a Learning Rate of 0.01 and a Momentum of 0.0. The specific experiment has also been performed with different learning rate and momentum parameter values. We observed that parameter tuning had no effect on the performance of the algorithm in the specific context. Hence, we do not consider necessary to present additional time-series graphs.





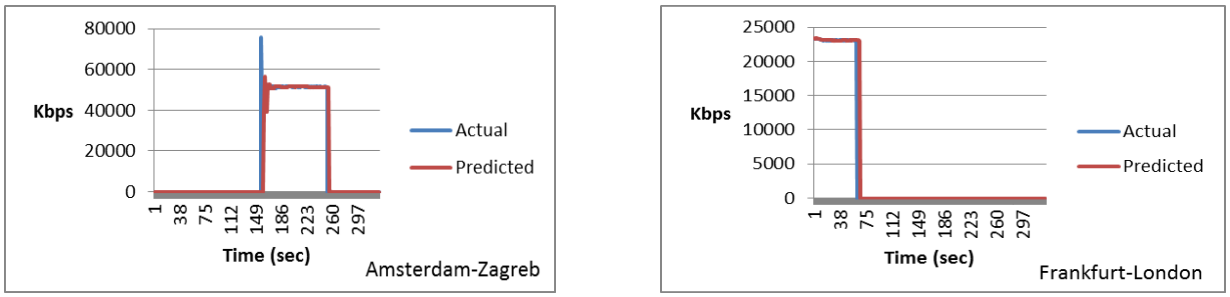


Figure 6-16: Actual and predicted traffic for 4 SD Pairs using pre-trained neural networks (TM1-ADI 40)

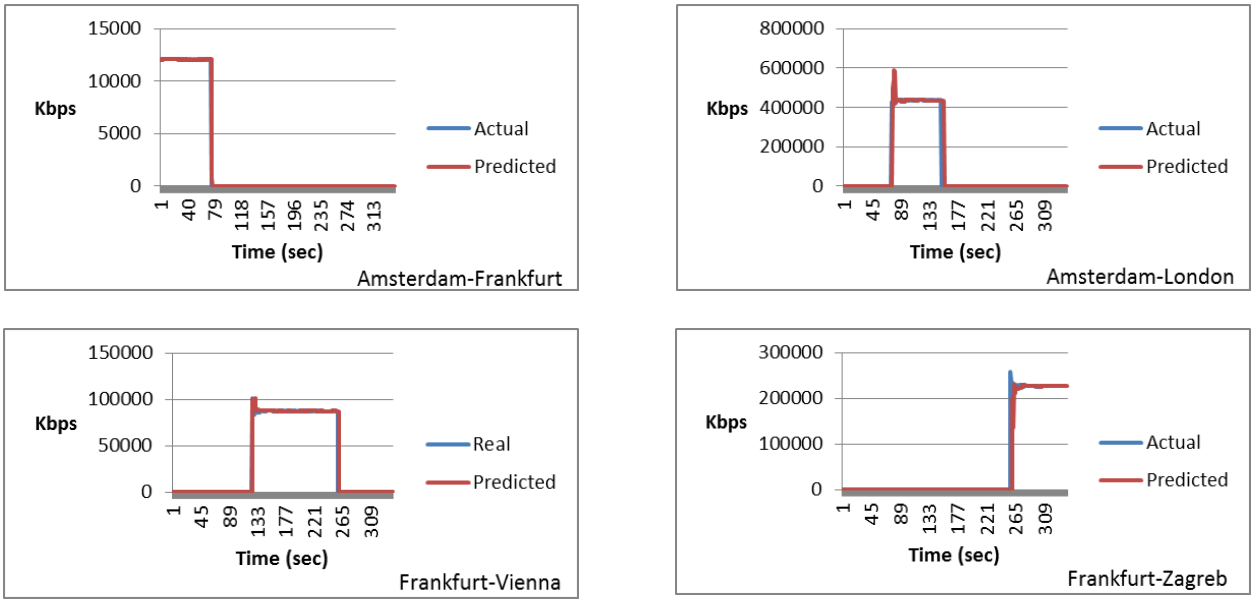
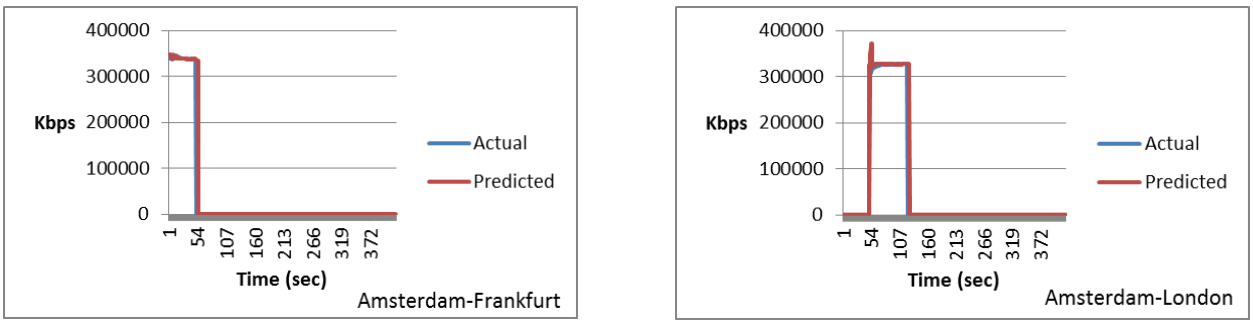


Figure 6-17: Actual and predicted traffic for 4 SD Pairs using pre-trained neural networks (TM1-ADI 280)



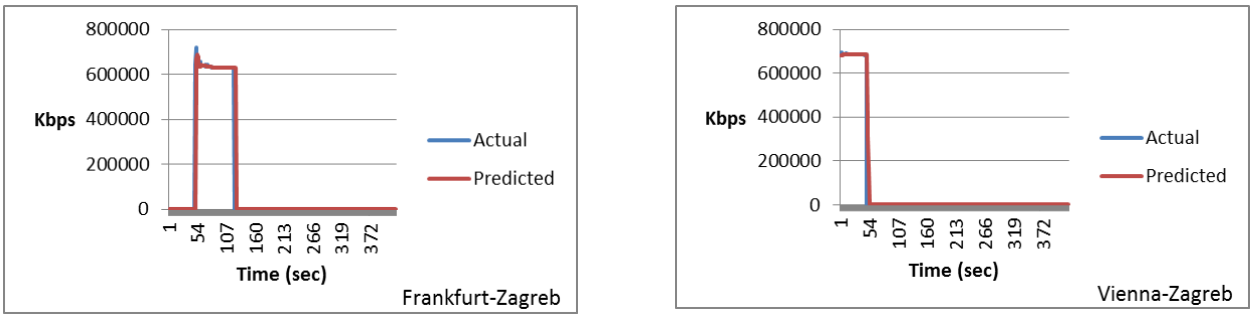


Figure 6-18: Actual and predicted traffic for 4 SD Pairs using pre-trained neural networks (TM1-ADI 400)

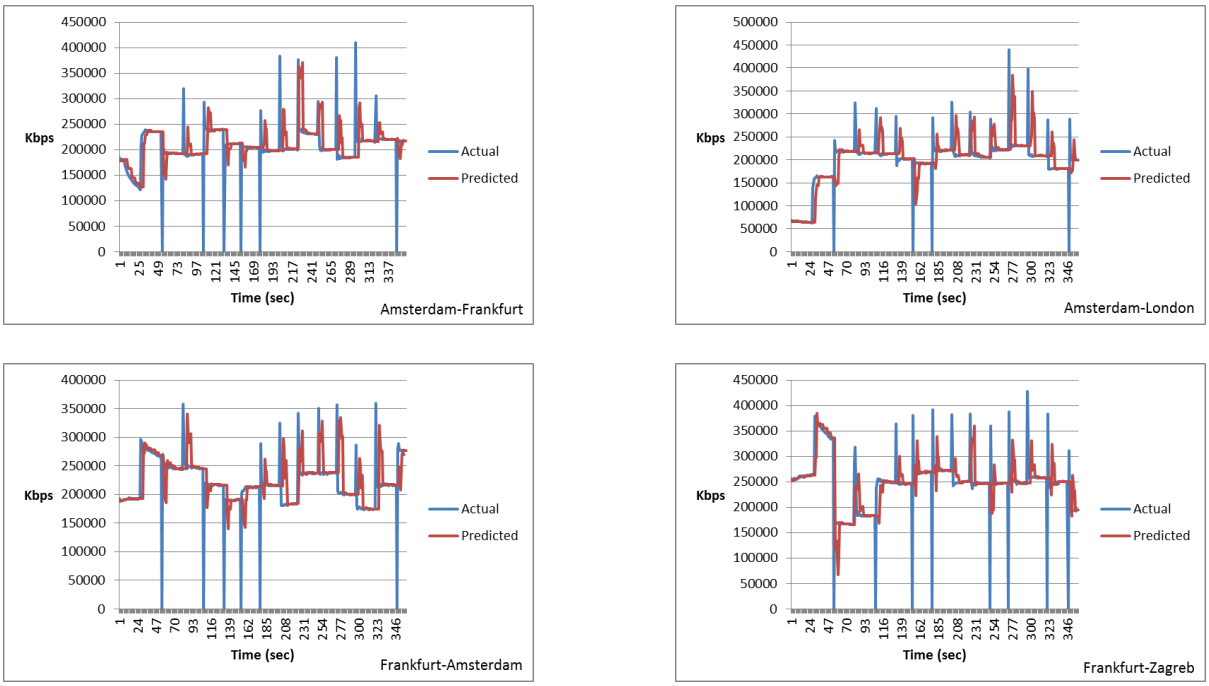


Figure 6-19: Actual and predicted traffic for 4 SD Pairs using pre-trained neural networks (TM2)

The results of the second scenario indicate that there is a high time and value accuracy of the predicted series against the actual ones. Compared to the results of the first scenario of the experiment, the time shift of the prediction is now minimized to 2-3 seconds. This is absolutely normal, since the entire network traffic pattern was used as the training set for Backpropagation algorithm in this case.

Moreover, because the training phase was omitted and the duration of the prediction phase in each MAPE loop was minimized to 5 seconds, we observed that the execution of the algorithm was faster and more efficient. In the first scenario the prediction phase in each MAPE loop is suspended for 50 seconds during each training phase, hence there is the need for a relatively large prediction vector of 55 values, which increases the

probability of prediction error. In this case prediction phases producing small 5-value vectors follow short monitoring periods of 5 seconds, resulting in faster traffic variation detection and subsequently prediction error minimization.

Despite the high accuracy of the prediction results, the main drawback of the method presented in this second scenario is that it requires offline training for various network traffic patterns prior to the actual forecasting.

After running both scenarios, we also made some observations which apply in both cases. While Traffic Model 1 scripts are running, at any moment there is a maximum of eight (8) SD Pairs with concurrent traffic. This fact has no impact on the ACL execution. When using Traffic Model 2 though, especially with scripts running on all five GÉANT PoPs, there are more than 30 active flows and all 20 SD Pairs have simultaneous traffic. In this case, having also to access files on disk, the execution of the ACL appeared to be slower, especially during the training phase in the first scenario. In some cases we observed that the 1-second interval between reading traffic demand of each SD Pair during the training phase actually corresponded to a real interval of 3 to 5 seconds. Although not affecting the quality of the prediction, since predicted values are based on simple vectors of sequential values, this fact is certainly indicative of a scalability issue.

6.5 Conclusions

The goal of the first experiment was to evaluate the performance and the quality of the implemented Backpropagation algorithm in predicting future network load in real time, during the deployment of the LP ACL in the ANM framework. The evaluation method was based on the comparison of the predicted values against the actual values of network load, which were being recorded throughout the experiment execution.

The results of both scenarios clearly show that the algorithm is quite sufficient in predicting network load in cases of traffic with little volatility or generally known traffic patterns and it can be used by SDN components in real time load forecasting, although it is limited by the network size. The algorithm is not able to handle random sudden surges in traffic demand (1-2 seconds).

Further investigation can be carried out on implementing different approximation and classification machine learning algorithms (i.e. Support Vector Machine and Self-Organizing Maps with regression) and comparing prediction results to Backpropagation, as well as utilizing load prediction results from other SDN components in decision making on routing and traffic engineering.

7 Experiment #5: Autonomic traffic engineering with load prediction in SDN/OpenFlow networks

7.1 Introduction

In this experiment we explore possible ways of load prediction exploitation in the ANM/SDN context, hence the ability of the network to continuously adjust its behavior regarding routing and traffic engineering in an autonomic way, based on short-term forecasting of network traffic. The motivation behind this particular experiment is to show whether a prediction algorithm, as described in the previous experiment, can be successfully applied in a SDN framework and contribute to the optimization of resource management and the efficiency of the network operation.

In order to realize the scenario proposed in this experiment, we utilized two different Autonomic Control Loops (ACLs), the LP (Load Prediction) ACL and the Core-TE (Traffic Engineering) ACL. The LP ACL, which has been described in the previous section, performs online load prediction and outputs vectors of estimated future load values for each source-destination pair (SD Pair) of the network. The Core-TE ACL controls the network functions through the enforcement of high-level policies like Energy Efficiency or Load Balancing. In this scenario, after both ACLs have been deployed in the ANM framework, they run concurrently and they exchange information. The short-term load prediction per SD Pair produced by the LP ACL is received by the Core-TE ACL. The Core-TE ACL enforces network configuration to proactively adapt to the future state by adjusting the paths used to accommodate traffic, either by aggregation or diffusion, based on estimated future load, thus resulting in Energy Efficiency or Load Balancing respectively.

The evaluation methodology relies on the comparison of the routing decisions made by the Core-TE ACL when receiving the actual traffic against the paths it chooses when it reads the estimated load of the next moment.

The main objective of this experiment is to highlight any possible practical value of load prediction in autonomous network management, as well as to examine the co-operation of different SDN applications in the ANM framework.

7.2 Actors, setup, topology

The general setup of the experiment resembles that of the previous experiment, with the addition of the Core-TE ACL. The ANM Core is used for the deployment of LP and Core-TE ACLs.

The GÉANT OpenFlow physical testbed with Virtual machines in the five Points of Presence (PoPs) located in Amsterdam, Frankfurt, London, Vienna and Zagreb, connected with OpenFlow software switches in a full mesh with 1Gbps links, was used for traffic generation. The general topology setup is shown in Table 7-1.

PoP	Host Private IP address	Switch name
Amsterdam	192.168.1.1	S1
Frankfurt	192.168.1.2	S2
London	192.168.1.3	S3
Vienna	192.168.1.4	S4
Zagreb	192.168.1.5	S5

Table 7-1: Topology set-up

In this particular experiment we observe what happens to the routing paths of the traffic of the active SD Pairs when traffic is detected on another link. For this purpose traffic was generated between three PoPs as follows.

Two scripts run on the Virtual Machines of Amsterdam and Vienna respectively. Initially, the host at Amsterdam (192.168.1.1) starts sending 30Mbps of traffic to Vienna (192.168.1.4) and after 150 seconds it starts sending 10Mbps of traffic to London (192.168.1.3). After 150 seconds the host in Vienna starts sending 20Mbps of traffic to the host in London.

7.3 Storyline

For the following experiment we assume that load prediction is performed by pre-trained neural networks, as described in the second scenario of the previous experiment (Section 6.4). For this purpose we have trained neural networks for the traffic model described in Section 7.2 and verified the prediction results prior to the execution of the experiment, following the steps described in Section 6.4.2.

In order to examine how load prediction influences traffic engineering, initially we deployed only the Core-TE ACL, after traffic has been generated between the three PoPs (Amsterdam to Vienna, Amsterdam to London and Vienna to London). Energy Efficiency was selected as a high level policy throughout the experiment. During execution the paths chosen by the Core-TE ACL for each SD Pair in each MAPE loop were stored, for future reference. To validate the integrity of the results, the above scenario (Core-TE ACL) was executed five (5) times.

Then, we generated the same traffic again between the three PoPs. This time, both the LP and Core-TE ACLs were deployed in the ANM framework. Upon deployment, the LP ACL started calculating the five future load values for each SD Pair and updated a CSV file in each MAPE loop. Instead of receiving real-time traffic demand of each SD Pair, the Core-TE ACL read the estimated load of each SD Pair from the CSV file in each MAPE loop. The ultimate goal of this experiment is to examine whether the Core-TE ACL will choose to reroute traffic of one or more SD Pairs as soon as it detects estimated additional traffic (by reading the predicted values).

The paths chosen by the Core-TE ACL for each SD Pair in each MAPE loop were also saved. The above scenario (Core-TE ACL with LP ACL) was also executed five (5) times.

Upon completion of execution, the paths chosen by the Core-TE ACL and the number of activated links in both scenarios (Core-TE ACL running with and without LP ACL) were compared. The results are presented in Section 7.5.

7.4 Execution

For validation purposes, each of the following test cases was executed five times. The execution steps are presented below:

Case #1 – Core-TE ACL

- The ANM Core component runs and the AUTOFLOW ANM Application starts.
- The Core-TE ACL runs and appears in the right up pane of the ANM Application.
- Real network traffic is generated by running Iperf scripts on two GÉANT PoPs (Amsterdam and Vienna).
- The Core-TE ACL is deployed in the ANM framework with Energy Efficiency as the high-level policy.
- After 10 minutes of execution, the Core-TE ACL was un-deployed and the file with the recorded paths for each SD Pair was updated.

Case #2 – Core-TE ACL with LP ACL

- The ANM Core component runs and the AUTOFLOW ANM Application starts.

- The Core-TE and the LP ACLs run and appear in the right up pane of the ANM Application.
- Real network traffic is generated by running Iperf scripts on two GÉANT PoPs (Amsterdam and Vienna).
- The LP ACL is deployed in the ANM framework.
- The Core-TE ACL is deployed in the ANM framework with Energy Efficiency as the high-level policy.
- After 10 minutes of concurrent execution, the two ACLs were un-deployed and the recorded paths for each SD Pair were updated.

7.5 Results

In the following tables we observe the active SD Pairs, the traffic route (paths) and the number of activated inter-switch links in two consecutive MAPE loops, as they were recorded during execution. Time values T0 and T1 refer to the loops before and after detecting additional traffic on the third SD Pair respectively.

The results after five executions of the Core-TE ACL running without the LP ACL are presented below.

Time	SD Pairs	Paths	Active links
T0	S1->S4 (Amsterdam-Vienna)	S1-S2-S3-S4	3
	S1->S3 (Amsterdam-London)	S1-S2-S3	
T1	S1->S4 (Amsterdam-Vienna)	S1-S2-S3-S4	5
	S1->S3 (Amsterdam-London)	S1-S2-S3	
	S4->S3 (Vienna-London)	S4-S5-S1-S2-S3	

Table 7-2: Core-TE ACL without LP ACL – Execution #1

Time	SD Pairs	Paths	Active links
T0	S1->S4 (Amsterdam-Vienna)	S1-S2-S3-S4	3
	S1->S3 (Amsterdam-London)	S1-S2-S3	
T1	S1->S4 (Amsterdam-Vienna)	S1-S2-S3-S4	5
	S1->S3 (Amsterdam-London)	S1-S2-S3	
	S4->S3 (Vienna-London)	S4-S5-S1-S2-S3	

Table 7-3: Core-TE ACL without LP ACL – Execution #2

Time	SD Pairs	Paths	Active links
T0	S1->S4 (Amsterdam-Vienna)	S1-S2-S3-S4	3
	S1->S3 (Amsterdam-London)	S1-S2-S3	
T1	S1->S4 (Amsterdam-Vienna)	S1-S2-S3-S4	5
	S1->S3 (Amsterdam-London)	S1-S2-S3	
	S4->S3 (Vienna-London)	S4-S5-S1-S2-S3	

Table 7-4: Core-TE ACL without LP ACL – Execution #3

Time	SD Pairs	Paths	Active links
T0	S1->S4 (Amsterdam-Vienna)	S1-S2-S3-S4	3
	S1->S3 (Amsterdam-London)	S1-S2-S3	
T1	S1->S4 (Amsterdam-Vienna)	S1-S2-S3-S4	5
	S1->S3 (Amsterdam-London)	S1-S2-S3	
	S4->S3 (Vienna-London)	S4-S5-S1-S2-S3	

Table 7-5: Core-TE ACL without LP ACL – Execution #4

Time	SD Pairs	Paths	Active links
T0	S1->S4 (Amsterdam-Vienna)	S1-S2-S3-S4	3
	S1->S3 (Amsterdam-London)	S1-S2-S3	
T1	S1->S4 (Amsterdam-Vienna)	S1-S2-S3-S4	5
	S1->S3 (Amsterdam-London)	S1-S2-S3	
	S4->S3 (Vienna-London)	S4-S5-S1-S2-S3	

Table 7-6: Core-TE ACL without LP ACL – Execution #5

We should clarify the aforementioned routing decisions (Table 7-2-Table 7-6) taken by the Core-TE ACL. Although the selected policy is Energy Efficiency, Core-TE routes traffic not from the direct link between an SD Pair but from paths with more links. Two reasons can be identified for this behavior. At first, Core-TE attempts to re-use already activated links and avoid the activation of unused ones. Therefore, it is possible to select a longer path (in number of hops) for the accommodation of traffic between an SD Pair and not the shortest one. In addition, Core-TE is designed to act proactively. It promotes the activation of specific links in the network, which belong to a ring that connects all switches. In this manner, subsequent requests will be accommodated by the already activated resources, resulting in the increase of the energy savings in the long term. Taking into account these two aspects of the Core-TE mechanism, the aforementioned routing decisions can be justified. More information about the Core-TE mechanism and the heuristic algorithm that is based on can be found in [5].

From the above tables it is clear that the Core-TE ACL shows a steady behavior during the five executions with the specific traffic model. As soon as actual traffic is detected on the third SD Pair (Vienna-London), the Core-TE chooses to route this traffic via the default path for the Energy Efficiency high-level policy, i.e. the ring between all switches (S4-S5-S1-S2-S3), according to the implementation of the traffic engineering algorithm. In each case the number of activated links increases from 3 to 5. Eventually, after some loops the Core-TE decides to reroute the traffic of the third SD Pair via a shorter path (S4-S3) to allow for better use of resources.

The tables below show the results after five executions of Core-TE and LP ACLs running simultaneously.

Time	SD Pairs	Paths	Active links
T0	S1->S4 (Amsterdam-Vienna)	S1-S2-S3-S4	3
	S1->S3 (Amsterdam-London)	S1-S2-S3	
T1	S1->S4 (Amsterdam-Vienna)	S1-S2-S3-S4	3
	S1->S3 (Amsterdam-London)	S1-S2-S3	

	S4->S3 (Vienna-London)	S4-S3	

Table 7-7: Core-TE ACL with LP ACL – Execution #1

Time	SD Pairs	Paths	Active links
T0	S1->S4 (Amsterdam-Vienna)	S1-S2-S3-S4	3
	S1->S3 (Amsterdam-London)	S1-S2-S3	
T1	S1->S4 (Amsterdam-Vienna)	S1-S2-S3-S4	4
	S1->S3 (Amsterdam-London)	S1-S2-S3	
	S4->S3 (Vienna-London)	S4-S2-S3	

Table 7-8: Core-TE ACL with LP ACL – Execution #2

Time	SD Pairs	Paths	Active links
T0	S1->S4 (Amsterdam-Vienna)	S1-S2-S3-S4	3
	S1->S3 (Amsterdam-London)	S1-S2-S3	
T1	S1->S4 (Amsterdam-Vienna)	S1-S2-S3-S4	3
	S1->S3 (Amsterdam-London)	S1-S2-S3	
	S4->S3 (Vienna-London)	S4-S3	

Table 7-9: Core-TE ACL with LP ACL – Execution #3

Time	SD Pairs	Paths	Active links
T0	S1->S4 (Amsterdam-Vienna)	S1-S2-S3-S4	3
	S1->S3 (Amsterdam-London)	S1-S2-S3	
T1	S1->S4 (Amsterdam-Vienna)	S1-S2-S3-S4	3
	S1->S3 (Amsterdam-London)	S1-S2-S3	

	S4->S3 (Vienna-London)	S4-S3	

Table 7-10: Core-TE ACL with LP ACL – Execution #4

Time	SD Pairs	Paths	Active links
T0	S1->S4 (Amsterdam-Vienna)	S1-S2-S3-S4	3
	S1->S3 (Amsterdam-London)	S1-S2-S3	
T1	S1->S4 (Amsterdam-Vienna)	S1-S2-S3-S4	5
	S1->S3 (Amsterdam-London)	S1-S2-S3	
	S4->S3 (Vienna-London)	S4-S5-S1-S2-S3	

Table 7-11: Core-TE ACL with LP ACL – Execution #5

The results above show that in one case the Core-TE used the “ring” path to route the traffic of the third SD Pair, whereas in the other four cases it opted for shorter paths (S4-S3 and S4-S2-S3). In these cases the number of activated links after the detection of new traffic remains the same or increases from 3 to 4, allowing the deactivation of more links (for energy efficiency).

To further examine and confirm the behavior of the two ACLs running concurrently, we executed the experiment another three times. The results were identical to the ones presented in Table 7-7.

Comparing the paths used to route the traffic and the activated links in the above two cases, we observe that there is an average gain of 30% regarding the number of activated links when traffic engineering utilizes load prediction to route the traffic. When Core-TE ACL runs without LP ACL, it decides to reroute the traffic in every loop based on the high-level policy and the state of the network at the specific millisecond during which various network parameters such as active SD Pairs, flows and traffic demand are captured. When traffic engineering algorithm (Core-TE) is in place, the frequent rerouting of traffic, which may occur on some flows even without changing the high-level policy, it is possible to affect the way that the ACL reads the value of traffic demand of a given SD Pair, if it is captured right after the traffic has been rerouted –although the actual traffic demand value of the SD Pair is not affected. On the other hand, load prediction is based on neural networks which have been pre-trained without traffic engineering being present in the network, thus the routing paths are the same throughout the execution and the traffic demand values recorded from each SD Pair are straightforward. When Core-TE ACL runs with load prediction, it is forced to read the predicted load for each SD Pair instead of reading actual traffic demand values. Being presented with more consistent sequential load values from the prediction CSV file, and considering the fact that each load value represents the traffic demand of the future moment, the Core-TE ACL has the ability to rapidly adapt to a future state by making a more efficient routing decision.

7.6 Conclusions

In this section we investigated the co-existence of traffic engineering and load prediction SDN applications, as well as the practical use of real-time network traffic forecasting in an SDN environment. To facilitate the evaluation method and simplify the presentation of the results, we performed a small-scale experiment examining the Energy Efficiency policy with a traffic model designed specifically for this purpose.

The execution process of the experiment clearly proved an absolutely smooth co-operation of two different software components in runtime inside the ANM framework. The results of this particular small-scale experiment demonstrated also that traffic engineering can be positively affected by real-time load forecasting, since the “software-based nature” of the network and its control functions allows for almost instantaneous adaptation to a future different state.

The average 30% gain in utilization of resources, although it seems quite promising, it should be considered as a positive indication in the context of the specific experiment only. Further experimentation is encouraged in order to discover the actual benefits of load prediction on traffic engineering in a network with more nodes and heavier traffic, as well as to examine possible scalability issues.

8 Conclusions and Future Plans

The ultimate objective of the AUTOFLOW project was to investigate the synergy of ANM - SDN and proceed with the integration of these two technologies in order to provide a valuable and realistic solution for the management and control of future networks that would have been validated over the real experimentation facility of the GÉANT. The conducted research resulted in a complete, validated solution that combines ANM and SDN for the operator friendly and efficient control and management of networks. Using the developed framework, autonomic traffic engineering (and other) mechanisms, acting as SDN applications, can be deployed dynamically and on-demand in a few seconds only, triggered either by operator's needs or the change in context of operation. The controller's northbound API provides to these mechanisms the access and control to network entities, but most importantly, it provides the means for subsequent development to other experimenters and/or 3rd Party developers. Moreover, accurate network's monitoring, as well as provisioning of global network's view was also achieved by exploiting the novel level of abstraction developed for this study namely, "end-to-end flow". Finally, the developed framework was validated at the decision level, were decisions e.g. new routing configurations were enforced to the network almost instantaneously, based on the (predicted) context of operation and by guaranteeing the satisfaction of high-level policies (e.g. energy efficiency, load balancing) in the network.

The essence of AUTOFLOW was the integration and experimental validation of ANM with SDN. This target was accomplished during the project's duration. However, this work can be further exploited and extended in terms of both research and service provisioning. The GÉANT OpenFlow facility would be the cornerstone for these studies and their validation. Below, we provide a short list with the most promising areas that AUTOFLOW could contribute:

- Interoperability of legacy systems with SDN.
- SDN to apply intelligence toward 5G communications.
- Provisioning of ANM (core and ACLs) as cloud services.
- Connectivity services based on SDN.
- QoE management.

Certainly, the activities of the past 18 months allowed us to enter the SDN ecosystem. Our intention is to continue our work and to this end, we are looking for opportunities to build upon the conducted studies and develop even more innovative solutions for the management of future networks.

References

[1] AUTOFLOW project milestone: Milestone M1.1. “Detailed experiment description and specifications”

[2] AUTOFLOW project milestone: Milestone M1.2. “Prefinal description of Software”

[3] AUTOFLOW project milestone: Milestone M1.3. “Final description of Software & Manual”

[4] Kephart, Jeffrey O., and David M. Chess. “The vision of autonomic computing,” Computer 36.1 (2003): 41-50.

[5] Foteinos, Vassilis, et al. “Operator-friendly Traffic Engineering in IP/MPLS Core Networks,” IEEE Transactions on Network and Service Management, Vol.: 11, Is.: 3, p. 333-349, 2014.

[6] “POX Controller”, accessed 19-February-2015. [Online]. Available: http://www.noxrepo.org/pox/about-pox/

[7] “Ryu”, accessed 19-February-2015. [Online]. Available: http://osrg.github.io/ryu/

[8] “Trema”, accessed 19-February-2015. [Online]. Available: http://trema.github.io/trema/

[9] “Floodlight”, accessed 19-February-2015. [Online]. Available: http://www.projectFloodlight.org/Floodlight/

[10] “OpenDaylight”, accessed 19-February-2015. [Online]. Available: http://www.opendaylight.org/

[11] Khondoker, Rahamatullah, et al. “Feature-based comparison and selection of Software Defined Networking (SDN) controllers,” Computer Applications and Information Systems (WCCAIS), 2014 World Congress on. IEEE, 2014. (2014).

[12] Sherwood, Rob, et al. “Flowvisor: A network virtualization layer,” OpenFlow Switch Consortium, Tech. Rep (2009).

[13] “Open vSwitch”, accessed 19-February-2015. [Online]. Available: http://openvswitch.org/

[14] Chowdhury, NM Mosharaf Kabir, and Raouf Boutaba. “A survey of network virtualization,” Computer Networks 54.5 (2010): 862-876.

[15] Bolla, Raffaele, et al. "Energy efficiency in the future internet: a survey of existing approaches and trends in energy-aware fixed network infrastructures." Communications Surveys & Tutorials, IEEE 13.2 (2011): 223-244.

[16] Kim, Sungsu. “Cognitive Model-Based Autonomic Fault Management in SDN,” PhD diss., Ph. D. thesis, Pohang University of Science and Technology, 2013.

[17] Kuklinski, Slawomir. “Programmable management framework for evolved SDN,” Network Operations and Management Symposium (NOMS), 2014 IEEE. IEEE, 2014.

[18] Wallner, Ryan, and Robert Cannistra. “An SDN Approach: Quality of Service using Big Switch’s Floodlight Open-source Controller,” Proceedings of the Asia-Pacific Advanced Network 35 (2013): 14-19.

[19] Bari, Md Faizul, et al. “PolicyCop: an autonomic QoS policy enforcement framework for software defined networks,” Future Networks and Services (SDN4FNS), 2013 IEEE SDN for. IEEE, 2013.

[20] Sezer, Sakir, et al. “Are we ready for SDN? Implementation challenges for software-defined networks,” Communications Magazine, IEEE 51.7 (2013): 36-43.

[21] Chowdhury, Shihabur Rahman, et al. “Payless: A low cost network monitoring framework for software defined networks,” Network Operations and Management Symposium (NOMS), 2014 IEEE. IEEE, 2014.

[22] Technical Annex B: GÉANT OpenFlow Facility [Online], Available: http://*geant3.archive.geant.net*

[23] “Xen”, accessed 19-February-2015. [Online]. Available: http://www.xenserver.org/

[24] “OpenFlow”, accessed 19-February-2015. [Online]. Available http://archive.OpenFlow.org/wp/learnmore/

[25] “Iperf”, accessed 19-February-2015. [Online]. Available https://Iperf.fr

[26] Learning switch method. https://github.com/mininet/OpenFlow-tutorial/wiki/Create-a-Learning-Switch

[27] Shen, Gangxiang, and RodneyS Tucker. “Energy-minimized design for IP over WDM networks,” Optical Communications and Networking, IEEE/OSA Journal of 1.1 (2009): 176-186.
[28] Alberti, Antonio Marcos. "Software-Defined Networking: Perspectives, Requirements, and Challenges." (2012).
[29] D. Awduche et al., “Overview and Principles of Internet Traffic Engineering,” IETF RFC 3272, May 2002.
[30] D. Awduche et al., “Requirements for traffic engineering over MPLS,” IETF RFC 2702, September 1999.
[31] Akyildiz, Ian F., et al. “A roadmap for traffic engineering in SDN-OpenFlow networks,” Computer Networks 71 (2014): 1-30.
[32] McKeown, Nick, et al. “OpenFlow: enabling innovation in campus networks,” ACM SIGCOMM Computer Communication Review 38.2 (2008): 69-74.
[33] Open Network Foundation. “Software-defined networking: The new norm for networks,” White Paper, (1):12, April 2012.
[34] Agarwal, Sugam, Murali Kodialam, and T. V. Lakshman. “Traffic engineering in software defined networks,” INFOCOM, 2013 Proceedings IEEE. IEEE, 2013.
[35] Jain, Sushant, et al. “B4: Experience with a globally-deployed software defined WAN,” ACM SIGCOMM Computer Communication Review. Vol. 43. No. 4. ACM, 2013.
[36] Hong, Chi-Yao, et al. “Achieving high utilization with software-driven WAN,” ACM SIGCOMM Computer Communication Review. Vol. 43. No. 4. ACM, 2013.
[37] Coiro, Angelo, Flavio Iervini, and Marco Listanti. “Distributed and adaptive interface switch off for internet energy saving,” Computer Communications and Networks (ICCCN), 2011 Proceedings of 20th International Conference on. IEEE, 2011.
[38] Al-Fares, Mohammad, et al. “Hedera: Dynamic Flow Scheduling for Data Center Networks,” NSDI. Vol. 10. 2010.
[39] Chaparadza, Ranganai, et al. “Engineering Future Network Governance,” European Journal for the informatics professional (2010).
[40] Saleem, K., et al. “A self-optimized multipath routing protocol for wireless sensor networks,” International Journal of Recent Trends in Engineering 2.1 (2009): 93-97.
[41] Almasi, Bela, and Szabolcs Szilagyi. “Multipath ftp and stream transmission analysis using the MPT software environment,” International Journal of Advanced Research in Computer and Communication Engineering 2.11 (2013): 4267-4272.
[42] Anjali, Tricha, et al. “Multipath network flows: Bounded buffers and jitter,” INFOCOM, 2010 Proceedings IEEE. IEEE, 2010.
[43] Park, Dong C., et al. “Electric load forecasting using an artificial neural network,” Power Systems, IEEE Transactions on 6.2 (1991): 442-449.
[44] Papalexopoulos, A. D., Shangyou Hao, and T-M. Peng. “Short-term system load forecasting using an artificial neural network,” Neural Networks to Power Systems, 1993. ANNPS'93, Proceedings of the Second International Forum on Applications of. IEEE, 1993.
[45] Yang, Bo, and Yuanzhang Sun. “An improved neural network prediction model for load demand in day-ahead electricity market,” Intelligent Control and Automation, 2008. WCICA 2008. 7th World Congress on. IEEE, 2008.
[46] Baumann, Thomas, and Alain J. Germond. “Application of the Kohonen network to short-term load forecasting,” Neural Networks to Power Systems, 1993. ANNPS'93, Proceedings of the Second International Forum on Applications of. IEEE, 1993.
[47] Sfetsos, Athanasios, and Costas Siriopoulos. “Time series forecasting with a hybrid clustering scheme and pattern recognition,” Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on 34.3 (2004): 399-405.

Glossary

ACL	Autonomic Control Loop
ADI	Average Demand Intensity
ANM	Autonomic Network Management
ANN	Artificial Neural Network
CSV	Comma-Separated Values
EE	Energy Efficiency
GOFF/Goff	GÉANT OpenFlow Facility
LB	Load Balancing
LP	Load Prediction
MAPE	Monitor-Plan-Analyze-Execute
NB	NorthBound
PoP	Point of Presence
QoS	Quality of Service
QoE	Quality of Experience
SD Pair	Source-Destination Pair
SDN	Software Defined Networking
SLA	Service Level Agreement
TE	Traffic Engineering
TM	Traffic Model
VM	Virtual Machine