ENDEAVOUR: Towards a flexible software-defined network ecosystem

ENDEAVOUR

Project name	ENDEAVOUR
Project ID	H2020-ICT-2014-1 Project No. 644960
Working Package Number	3
Deliverable Number	D.3.4
Deliverable Number	3.4
Document title	Deployment of the prototype of the monitoring platform
Document version	0.8
Editor in Chief	Castro, QMUL
Authors	Fernandes, Boettger, Deng, Castro, Antichi, Lapeyrade, Owezarski, Clauberg, Gusat
Date	21/11/2017
Reviewer	Antichi, Chiesa
Date of Review	21/12/2017
Status	*Public*

Revision History

Date	Version	Description	Author
21/11/17	0.1	First draft	Castro
1/12/17	0.2	Main content	Fernandes, Boettger, Deng, Castro
04/12/17	0.3	Some corrections	Antichi
13/12/17	0.4	IBM testbed	Clauberg
14/12/17	0.5	Corrections	Castro
19/12/17	0.6	New results	Antichi
20/12/17	0.7	Review	Chiesa
21/12/17	0.8	implementation of review comments	Castro

Executive Summary

This is the accompanying report of the demonstrator of Work Package 3 for month 36, where the deployment of the prototype of the ENDEAVOUR monitoring platform will be demonstrated. In this report, we briefly discuss the prototype and its scalability.

Contents

1 Introduction

In this report, we document the deployment of the prototype of the ENDEAVOUR monitoring platform. Further details of its architecture were presented in the previous deliverable [3], while the evaluation in a emulated environment in [2].

2 Description of the monitoring platform prototype

2.1 Hardware involved

In this demonstrator we use three test-beds: one located in DE-CIX, one set-up in Cambridge where OSNT middlebox has been also deployed, and another one at the IBM Research - Zurich premises where additional experiments were conducted.

2.1.1 The DE-CIX Testbed

Figure 1 presents the logical topology at the DE-CIX site as used in the first two demonstrators of the ENDEAVOUR Monitoring Platform (see Sections 3.1 and 3.2). In the topology presented in Figure 1, the ENDEAVOUR platform connects 7 routers from 6 participants. Note that member B owns 2 routers. Further details of the testbed at DE-CIX can be found at [4]

2.1.2 The Cambridge testbed

As access to the DE-CIX datacenter is not immediate due to the permission clearance there required, a separate testbed that includes the OSNT component was deployed at the University of Cambridge. Note that the experimental nature of the OSNT hardware alongside the rescritions related to the physical access at the DE-CIX testbed have played a key role in such a decision. Accordingly, it was agreed with the Project Officer to have a separate testbed. Figure 2 depicts the logical view of this testbed, which is physically located at the premises of the Computer Laboratory of the University of Cambridge.

As the purpose of this testbed is exclusively demonstrating the integration of the OSNT and the anomaly detection system, the testbed only includes the components required for this purpose. Figure 2 presents the logical view of this testbed along with its main functions.

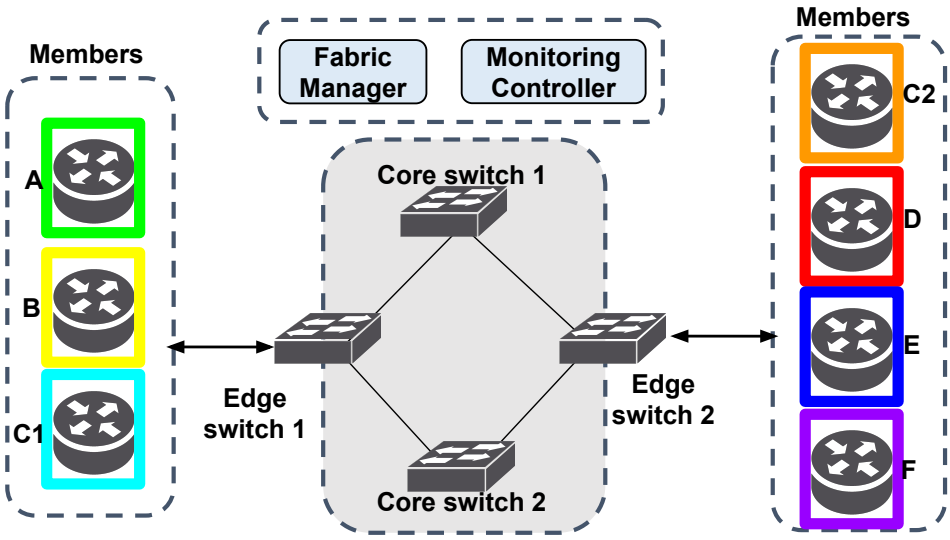

Figure 1: Testbed at the DE-CIX site.

Preliminary tests showed limited scalability: the combination of OSNT hardware and anomaly detection software were not able to cope with demanding input traffic rates. As a result, to improve the capabilities of the system we went through some modifications of the components:

- **Hardware improvement (header extraction):** the hardware preparses the packet and only sends the information needed by the anomaly detection software alongside with the hardware timestamp. This feature allows us to squeeze even more the bytes for a single packet, reducing it to 24 bytes instead of the previous 64 bytes.
- **Hardware improvement (packet batching):** the OSNT implementation has been modified to batch multiple packets in single big chunk of data. This feature allows us to send multiple packets to the software chain in a single transaction when the packet rate is high. This has been done to cope with the limited memory access rate from

the OSNT card to the machine userspace.

- **Software improvement (feature extraction process):** the processing of the packets to extract the flow statistics is now separated from the anomaly detection process. This allows us to optimize the feature extraction and provide more flexibility.

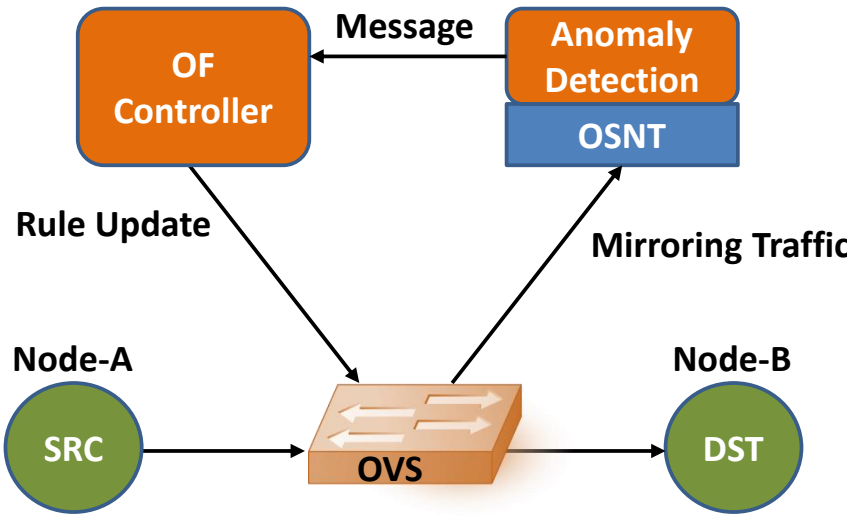

Figure 2: Testbed at the University of Cambridge.

Figure 3 presents the configuration of the OSNT with the Open vSwitch (OVS) switch at the testbed.

2.1.3 The testbed at IBM

Alternatively to the main implementations from the two previous testbeds, IBM used a testbed similar to the Cambridge testbed, but relying on an Intel FM10480 switch instead of the OVS switch and the C-GEP Field Programmable Gate Array (FPGA)-based accelerator [6] instead of the OSNT unit. Two different solutions where explored at this testbed, namely:

1. **zMon Heatmaps:** This solution monitors the switch queues to build a global topology-dependent view of the switch fabric utilization and

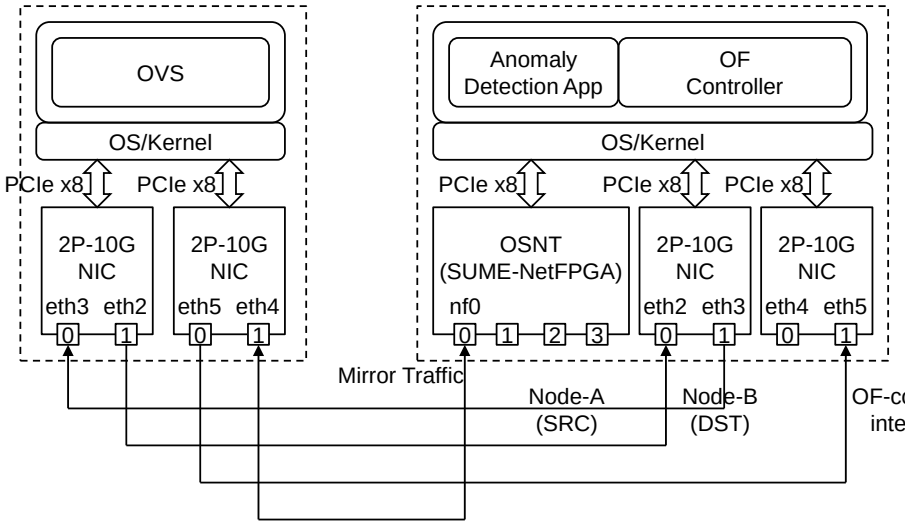

Figure 3: Configuration of OSNT hardware with the OVS switch.

sends congestion notifications to the traffic sources to adapt the input flows to the available capacity of the fabric.

2. **zMon Traffic Matrix:** This solution uses the switch capability of port mirroring to monitor the Layer 3 Internet Protocol (IP) source and destination address pairs and, thus, to identify topology-independent flow communication patterns.

Additionally, a special use case of the IBM testbed demonstrating extremely fast Distributed Denial of Service (DDoS) mitigation has been published recently [5].

As this solutions result from a later exploration to the main solutions reported here, we now briefly describe them before discussing its theoretical scalability in Section 3.4.

zMon Heatmaps. The size of the switch fabric of an Internet eXchange Point (IXP) depends on the number of routers from different Internet Service Provider (ISP) it connects to. The fabric may be built just of a single switch or of multiple complex switches. The goal of the zMon Heatmap solution was to build space- and time-coherent snapshots of the fabric by visualizing the occupancy of all the switch queues of the fabric. More precisely, this means that zMon Heatmap generates global network images from sparsely

sampled queues. Sequences of such images build so-called "movies" that allow to visualize the evolution of the network state in time. Details of the heatmap principle were already described in [3].

The limitations of this method are enumerated in Section 3.4. In order to address such limitations, we have changed our method to use port mirroring instead of the Quantized Congestion Notification (QCN) sampler and to calculate throughput per IP source and destination pair. This implementation is described in the next section.

zMon Traffic Matrix. Port mirroring is supported on all modern switches. We make use of the simplicity of building a mirroring system for traffic monitoring. There are different ways of measuring the network activity or utilization at a switch level:

1. per switch port (link utilization);
2. per Layer 4 flow;
3. per IP source and destination pair.

The link utilization per switch port is quantified by measuring the traffic rate from every input port to every output port. The monitoring software then must determine the output port to which the switch input port sends the packet, thereby repeating the same switching, routing, and load balancing decisions as the switch has previously done. The results are then visualized in a topology-dependent heatmap. Measuring utilization per Layer-4 flow is difficult since the number of Layer-4 flows changes dynamically and can be very large for Tbps switches. On the other hand, the IP source-destination flows are topology-independent and easier to generate (traffic matrices). Therefore, we chose to proceed with option 3 due to its relative simplicity.

For an IXP fabric all the IP addresses are external. Therefore, instead of monitoring all the individual IP addresses, we monitor only IP subnets as source and destination addresses. In an IXP, these subnet addresses correspond to the ISP using the IXP for direct packet forwarding from one ISP to a neighbouring ISP. Subnet addresses that do not belong to an ISP directly connected to the IXP are treated as unknown addresses for the traffic matrix and shown in a last row/column added for traffic with unknown IP subnet addresses.

In the collected traffic matrix, every 'pixel' accumulates the number of bytes seen since the last snapshot. A collector corrects the pixel values. An

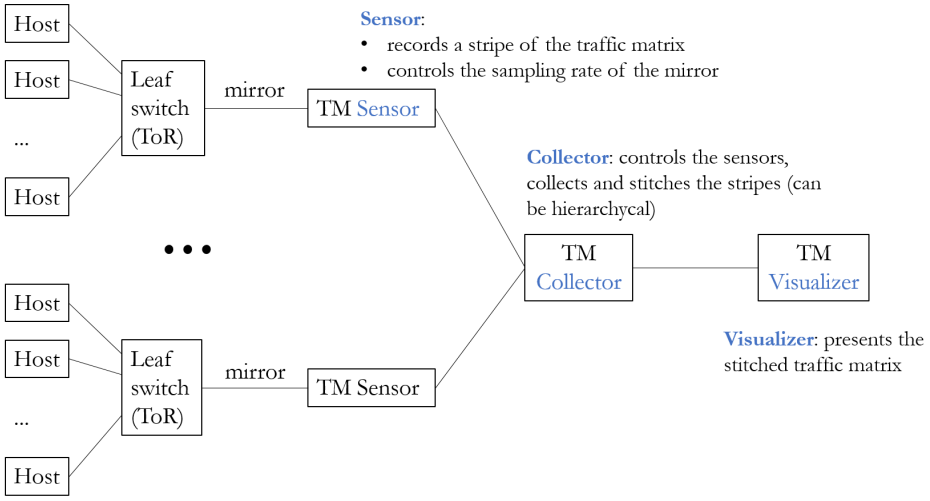

Figure 4: zMon Traffic Matrix Architecture Overview.

overview of the architecture of the zMon traffic Matrix solution is shown in Figure 4. A traffic matrix row or column monitored by a single throughput sensor (TM Sensor) is called a "stripe".

The TM Sensor of the traffic matrix implements four processing stages as follows:

- T1: Packet Capture
  1. Receive a burst of packets.
  2. Transfer the packets to the parser T2.
  3. Repeat from step T1.1.
- T2: Packet Parser
  1. Get the packets from the capture processing stage.
  2. Parse the packets and extract the headers.
  3. Transfer the extracted headers with the corresponding metadata to T3.
  4. Repeat from step T2.1.
- T3: Throughput Sensor

1. Get headers and metadata from the parser T2.
2. Map addresses to traffic matrix coordinates.
3. Filter out unneeded traffic which is captured by another sensor.
4. Accumulate the traffic to the active stripe.
5. Repeat from step T3.1.

- T4: Trigger and Communicate

1. Wait and receive command from the collector.
2. If a trigger is received, then continue from step T4.3. Else if a request for data is received, continue from step T4.5.
3. Swap the active and passive stripe(s) of the throughput sensor.
4. Repeat from step T4.1.
5. Send the passive stripe(s) to the collector.
6. Clear the passive stripe(s).
7. Repeat from step T4.1.

Furthermore, we list below the internal operations of the collector:

1. Wait for the next sampling time.
2. Send trigger to all sensors (broadcast or multicast).
3. Read the data from all sensors (one by one).
4. Amplify the values (multiply by the frame rate—picture, not Ethernet—to convert to bytes/second).
5. Transfer the complete matrix to the visualizer.
6. Repeat from step 1.

The internal operations of the visualizer are the following ones:

1. Receive a complete matrix from the collector.
2. Convert the values of the matrix to log-scale.
3. Display the result as a traffic matrix (heatmap).
4. Repeat from step 1.

Figure 5 shows an example of a traffic matrix for the specific communication pattern.

In addition to these two solutions, we studied a specific use case for very fast DDoS mitigation within the IBM testbed [5].

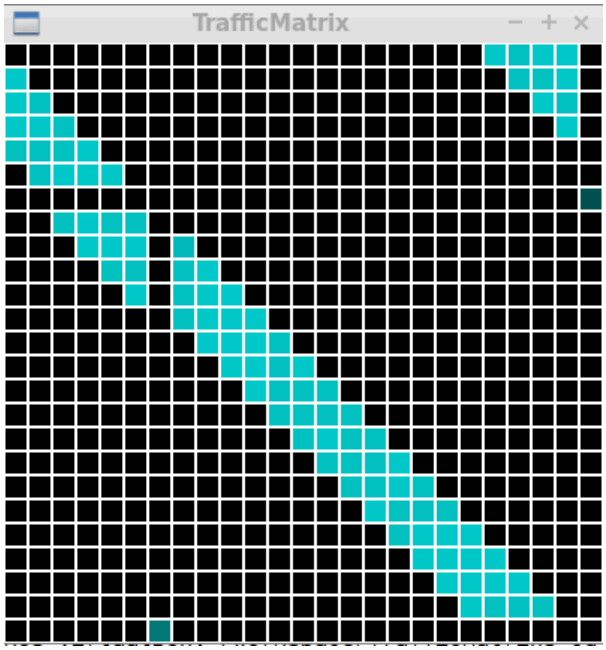

Figure 5: An example of a traffic matrix for the case when every ISP sends packets to the next 4 ISPs.

2.2 Access to the testbeds

Contrary to the case of the platform simulator (see [2]), access to the testbeds mentioned above is not public, being instead restricted to the members of the consortium.

3 Scalability of the monitoring platform prototype

3.1 Polling-request scalability

In this demonstrator, we use the testbed in DE-CIX to show that for a constant increase in the number of flows in the dataplane, the performance of the controller in terms of CPU and memory remains stable and does not disrupt the traffic exchanged by the members in the dataplane.

In this experiment, 1000 flows per second are installed by the ENDEAVOUR controller and the Monitoring Controller requests statistics from the switch in almost real time, every second. While this is likely to be an ex-

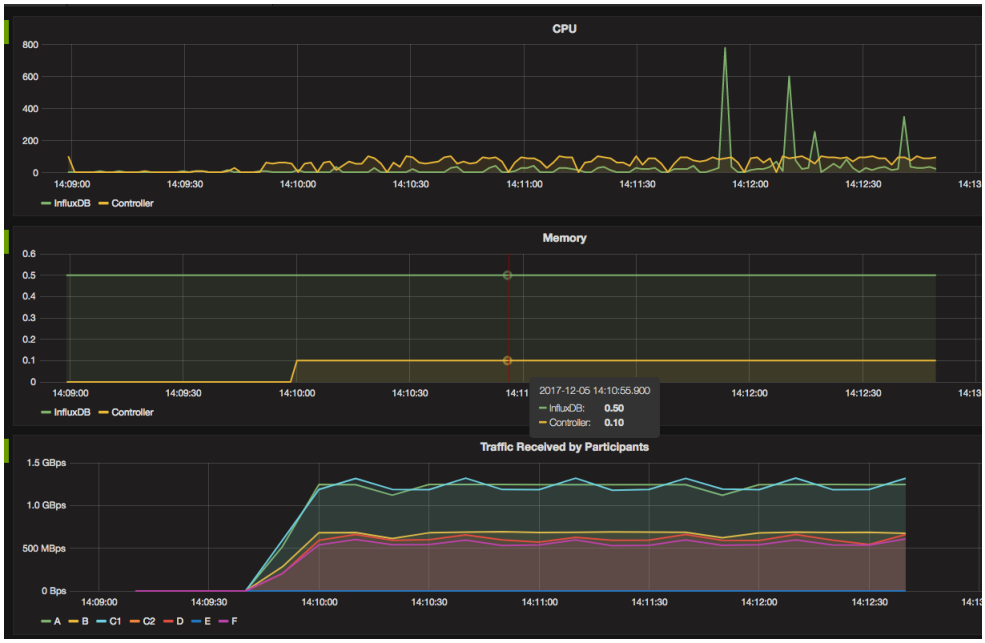

Figure 6: Memory and CPU of the Monitoring Database (1 polling request per sec.).

cessive amount of requests, we can show how the testbed can support it. Note that if the memory occupied by the database or the CPU consumption resulting from the frequent polling requests would grow exponentially, the monitoring architecture would be deem as non-scalable.

Figure 6, presents a caption of the Grafana visualizer used in the demonstrator, which shows a clear and stable pattern of the memory occupied by the Monitoring Database. The CPU presents spikes because the database requires more processing when writing data. Overall, CPU is stable for the most time of the demonstration.

We experimented with lower frequencies of polling requests, which naturally resulted in less demanding consumption of memory and CPU. If near real time statistics is not crucial for the IXP activities, a lower polling interval is recommended. As the results for the experiment with 1 request per second encompass the others, we do not describe them in any further detail.

The video demonstrating the scalability of the Monitoring Database regarding the polling-requests is available at: `https://www.youtube.com/watch?v=3SNgBLLITtE&feature=youtu.be`

3.2 Monitoring Policies scalability

In this experiment, we use the testbed in DE-CIX to demonstrate that ENDEAVOUR monitoring platform can support the installation of a large number of policies.

The demonstrator starts with each participant having 100 outbound policies. From there on, 10 new policies per member are added every 30 seconds. As more policies are installed, the traffic for each participant increases accordingly.

Similarly to the previous case, the CPU usage spikes as the database writes the data collected by the monitoring platform collects statistics. Despite this overheads, the demonstrator shows that there is no impact on the dataplane.

Note that, in this experiment, we keep the near-real time monitoring approach, i.e., polling every second. Lower polling-frequency would naturally result in a lower CPU impact.

The video demonstrating the scalability of the Monitoring Database regarding the number of installed policies is available at: `https://youtu.be/jH1Mv6QAB7A`

3.3 Integration of OSNT in the monitoring platform: scalability

In this demonstrator, we discuss the scalability of the monitoring platform when the OSNT middleboxes are required. We rely on the Cambridge testbed. Note that, while we illustrate it in the context of the anomaly detection, the findings would be valid for any other use case depending on the OSNT middleboxes.

The main scalability aspect that we explore here relates to the ability of the OSNT probes to process the required information in a timely fashion. The main dimensions of scalability that we explore are packet rate and number of flows. More precisely, we look at the impact that more demanding conditions, i.e., higher packet rate or number of flows, have on the ability of the monitoring platform to provide the adequate information. In this case, we look at how the number of errors in the detection of traffic anomalies varies along the aforementioned dimensions.

In order to perform such evaluation, we use network traffic from an ISP entry point and insert attacks in the traffic traces, which we then run through our testbed. The trace is 16GB long and represent 300 seconds of real traffic. We repeat the test at a varying packet rate to understand the

impact of an increasing input rate on the detection capabilities.

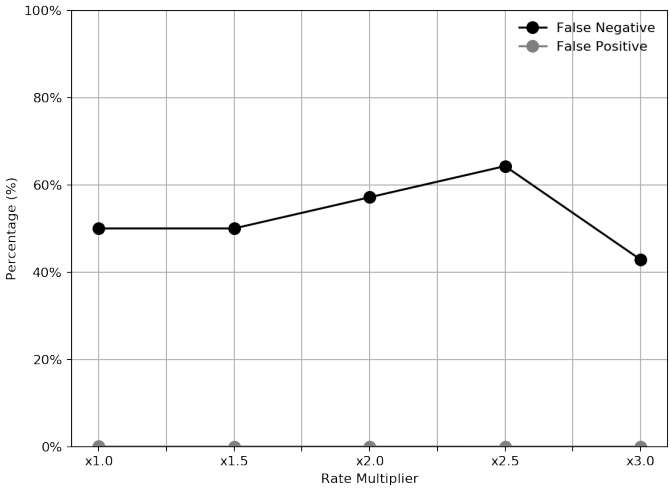

Figure 7: False positives/negatives and packet rate.

Figure 7 shows how regardless of the input packet rate, the testbed does not incur in a significant difference in the detection properties. Note that the anomaly detection system performs its computation in a statistical manner. This means that false positive or negative ones can still appear. However, increasing the generation rate of the trace, does not change significantly with respect the baseline, i.e., the `x1.0` case.

Finally, we add more artificial legitimate flows into the actual traces through the same interface.

Figure 8 shows that, also in this case, there is not a significant increased number of false negative or negative when we add more flows into the traffic. This demonstrate the scalability of the system with respect to the input flow volume.

The video `https://www.youtube.com/watch?v=qSVF58htEIg` demonstrates the anomaly detection monitoring capabilities in action. Note that this was already demonstrated in the previous deliverable (see [2]). This is because we directly implement such a system using real working hardware without the need to go through the intermediate step of simulation.

3.4 Scalability of additional solutions at the IBM testbed

Note that as the zMon Heatmap and Matrix solutions result from a later exploration to the main ones already here reported, they were only eval-

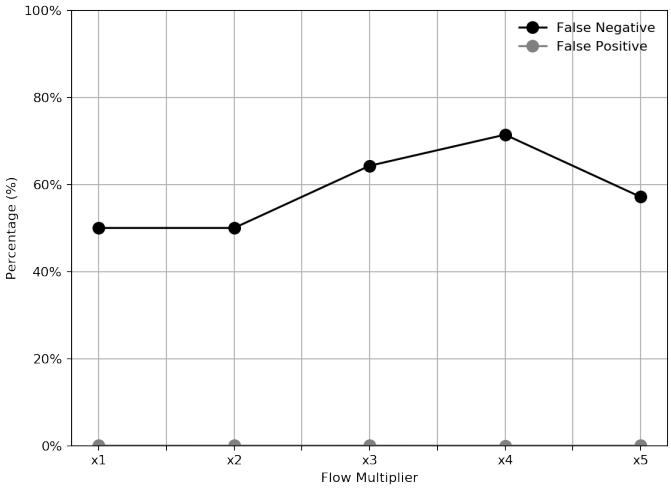

Figure 8: False positives/negatives and number of flows.

uated in the IBM testbed and the scalability of these implementations is theoretically analyzed.

zMon Heatmap The scalability of this solution mainly depends on the number of queues whose occupancies must be monitored and the time coherence between the different switches in the fabric necessary to provide a global time-coherent view of all the fabric queues.

Number of Queues: The number of queues to be controlled in a leaf-spine-core architecture is equal to the number of links from the leaf-ports to the uplinks of the leaf switches, plus the number of links from the spine-switchers to the core-switches, and doubling these numbers for counting all the queues from the core switches down to the leaf-ports. Only if each leaf-port sends packets to all other leaf-ports in the system will all queues be active. In normal IXP operation, only a small amount of the queues will be active at a specific point in time. Since the number of leaf-switches, spine-switches, and core-switches grow linearly with the number of total leaf-ports in a leaf-spine-core architecture, the number of links and correspondingly queues to monitor grows linearly with the total number of leaf-ports N.

Time coherence: If theIXP fabric consists of multiple switches, time coherence between these switches is an important aspect for our snapshots in time. However, by using the hardware Precision Time Protocol (PTP) [1], we can achieve nanosecond granularity. So, time coherence is not limiting

scalability.

This implementation of the monitoring platform was built using the IEEE 802.QCN standard.

A demo video of the application is available at `https://youtu.be/jxfpVYzbGI8` and `https://youtu.be/PAGyW9sJkvg`.

We observed three main limitations with this method:

- for bursty traffic the queue occupancy can be high both at low and high throughput;
- if the queue occupancy changes fast, we must sample at Tbps rates;
- 802.Qau QCN functionality support is not as common as port mirroring.

zMon Traffic Matrix In order to asses the scalability of this solution,

1. we mirror N ports to 1 monitoring (mirroring) port ($N:1$). Modern switches have between 32 and 128 ports of 40/100 Gbps. We truncate each switch packet to 64 or 96 bytes to prohibit over-subscription of the monitoring port. Nevertheless, for high packet rates, at a sufficiently large number of input ports, mirroring the entire traffic from N input ports to a single mirror port is impossible. Statistical sampling of the input ports with a rate adapted to the utilization of the mirroring port is unavoidable. If the limitation from the mirroring port leads to a sampling rate too low for detecting traffic patterns, it is necessary to increase the number of mirroring ports. This finally leads to an increase of switches within the switch fabric. However, considering the number of ISPs usually connected to the same IXP, we do not expect to reach this point.
2. the traffic matrix grows quadratically with the network size and is usually sparse. Matrix compression can be used to reduce the matrix size, but, in the worst case, with an all-to-all traffic pattern, the matrix is dense and grows with a complexity of $O((M+1)x(M+1))$ when M is the number of connected ISPs.
3. time coherence: as in the previous case, PTP allows for nanosecond granulatrity and therfore, tie coherence poses no constraints upon the scalability of this solution.

4 Summary

This report documented the demonstrator of Work Package 3 for month 36, where the deployment of the prototype of the ENDEAVOUR monitoring platform has been evaluated. In this report, we briefly discussed the prototype and its scalability together with some additional solutions.

5 Acronyms

ISP Internet Service Provider

IXP Internet eXchange Point

IP Internet Protocol

DE-CIX German Commercial Internet Exchange

DDoS Distributed Denial of Service

OVS Open vSwitch

QCN Quantized Congestion Notification

OSNT Open Source Network Tester

FPGA Field Programmable Gate Array

PTP Precision Time Protocol

References

[1] IEEE Standards Association. Precision Time Protocol (PTP), 2008.

[2] ENDEAVOUR consortium. D.2.4: Implementation of the monitoring platform. *ENDEAVOUR deliverable*, December 2016.

[3] ENDEAVOUR consortium. D.3.2: Design and final requirements of the monitoring platform. *ENDEAVOUR deliverable*, January 2016.

[4] ENDEAVOUR consortium. D.4.6: Final report about tests. *ENDEAVOUR deliverable*, December 2016.

[5] P. Varga, G. Kathareios, A. Mate, R. Clauberg, A. Anghel, P. Orosz, B. Nagy, T. Tthfalusi, L. Kovcs, and M. Gusat. Real-Time Security Services for SDN-based Datacenters. In *13th International Conference on Network and Service Management (CNSM)*, 2017.

[6] P. Varga, L. Kovcs, T. Tthfalusi, and P. Orosz. C-GEP: 100 Gbit/s Capable, FPGA-based Reconfigurable Network Equipment. In *IEEE 16th International Conference on High Performance Switching and Routing (HPSR)*, 2015.