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	3.2
Document title	Design and Final Requirements of the Monitoring Platform
Document version	0.9
Editor in Chief	Castro, QMUL
Authors	Castro, Antichi, Boettger, Leao, Dietzel, Bleidner, Uhlig, Kathareios
Date	18/01/2016
Reviewer	Bleidner, Dietzel, DE-CIX
Date of Review	18/01/2016
Status	*Public*

Revision History

Date	Version	Description	Author
01/07/15	0.1	Preliminary draft	Castro
02/07/15	0.2	Monitoring primitives	Antichi, Castro
23/12/15	0.3	First complete draft	Castro, Antichi, Boettger, Leao, Dietzel, Bleidner, Uhlig, Kathareios
04/01/16	0.4	Merge of small and large IXP sections	Castro
08/01/16	0.5	Monitoring limitations	Bleidner, Dietzel, Bruyere
14/01/16	0.6	Final pre-revision changes	Castro, Antichi, Boettger, Leao, Dietzel, Bleidner, Uhlig, Bruyere, Gusat, Kathareios
15/01/16	0.7	Conclusions and executive summary	Castro, Bruyere
18/01/16	0.8	Review	Bleidner, Dietzel
22/01/16	0.9	Addressing of review comments	Castro, Uhlig

Executive Summary

ENDEAVOUR's mission is advancing the Internet interconnection model to a new paradigm through the introduction of SDN technology at one of the central elements of the Internet architecture, the IXP. The implementation of novel SDN capabilities is closely intertwined with the monitoring abilities at the IXP. While the previous deliverable D.3.1 [8] surveyed the monitoring needs relevant for the new capabilities that SDN brings to the IXP, this document continues that work by closely looking at the current monitoring practices and technologies, exposes its limitations and proposes a monitoring architecture for the ENDEAVOUR project.

In view of the drawbacks of legacy switch monitoring and the limitations of the existing instances of SDN switches, we propose a monitoring architecture that is independent of the switching infrastructure. Our monitoring architecture relies on a monitoring controller that coordinates lightweight middleboxes, which in turn complement the SDN-switch monitoring capabilities.

This deliverable analyzes the main functionalities that the middleboxes will offer and discusses the next stages in which we will work toward integrating all the elements of the monitoring architecture under the umbrella of a controller.

Contents

1 Introduction **5**

2 State-of-the-art in IXP monitoring **5**
2.1 Passive monitoring 6
2.1.1 Simple Network Management Protocol 6
2.1.2 Packet sampling - the sFlow approach 7
2.1.3 Flow counters - the NetFlow approach 8
2.1.4 Port mirroring 9
2.2 Active monitoring 10
2.3 Future directions in monitoring 10
2.3.1 Quantized Congestion Notification 10
2.3.2 OpenFlow Switches 13
2.4 Limitations 14

3 ENDEAVOUR monitoring architecture **16**
3.1 Monitoring requirements 18
3.2 Monitoring middleboxes 22

4 Outlook **24**

5 Acronyms **25**

List of Tables

1 Monitoring requirements and middleboxes per use case 21

List of Figures

1 The closed loop of the QCN standard 11
2 QCN's sampling probability 12
3 CNMS heatmaps 13
4 Monitoring architecture 20
5 Architecture for OSNT traffic monitoring subsystem. 23
6 Architecture for OSNT traffic generation system. 24

1 Introduction

Monitoring is a critical functionality in a complex system such as a large-scale Internet eXchange Point (IXP). As the previous deliverable demonstrated [8], Software Defined Network (SDN) introduces both challenges and opportunities and it is critical at enabling the use cases already discussed. Additionally, because the design and implementation phase will need feedback, for instance to identify bugs, monitoring will be a critical element in the building process leading to the final prototype. For instance, active monitoring for debugging will be an extremely useful functionality.

The current practice of scheduled monitoring typical of today's networks, does not match the needs of the dynamism required by the current Internet and do not meet the challenges posed by SDN. However the existing commercial SDN switches fall short of fulfilling the potential of SDN. The hardware and software limitations of available SDN switches are a challenge for the monitoring needs of ENDEAVOUR. As developing our own switching hardware is beyond the scope of the project, we will rely on existing commercial SDN switches and complement them with flexible lightweight monitoring middleboxes. Our monitoring infrastructure will hence rely on SDN switches and middleboxes coordinated by a controller.

This decoupling of the switching and the monitoring infrastructure allows for more comprehensive monitoring capabilities without imposing limitations on the switching fabric, enables a wide range of active measurement capabilities, and makes the monitoring and switching infrastructures more capable to adapt to novel needs stemming from the dynamic nature of the Internet.

This deliverable describes the basic monitoring architecture, its elements, and sets the stage for the future work. The most immediate task of the future work will consist in developing the monitoring controller to integrate the different elements of the monitoring system.

2 State-of-the-art in IXP monitoring

Whether large or small, IXPs typically rely on passive monitoring. Whereas small and medium IXPs focus on essential monitoring because their human and technical resources are more limited, large IXPs usually have a more comprehensive monitoring effort. To have a more comprehensive monitoring, IXPs sometimes complement its passive capabilities with limited active monitoring.

Monitoring at small and medium IXPs typically offers supervision of performance degradation as well as statistics and visualization of the traffic exchanged. Recent initiatives such as IXP-Manager[1], developed by the medium sized IXP, Ireland Internet Exchange Point (INEX)[2], help small and medium IXP to deploy all the necessary well-known monitoring.

Large IXPs such as the German Commercial Internet Exchange (DE-CIX) or Amsterdam Internet Exchange (AMS-IX) operate huge Ethernet based networks with a peak traffic volume of up to five Tbps. Those large IXPs offer strict Service-Level Agreements (SLAs) to ensure a guaranteed level of service for their members. Holistic monitoring capabilities including each component of their large scale network as well as additional components such as the route server or Domain Name System (DNS) server are a key component in ensuring the fulfilment of the SLAs.

With the growing number of members and port speeds, large IXPs have been pushed to a distributed network infrastructure. A single switch does not satisfy the scalability and bandwidth requirements anymore. With the rise of 100 Gbps port speeds, IXPs have also moved from copper cables to a complete optical infrastructure. The central monitoring system also gathers operational information and counters for those optical components interconnecting the distributed switching fabric. Especially, the current light levels of the transceivers must be checked periodically for early detection of possible failures of the optical transmissions.

In the rest of this section we discuss the monitoring standards and tools common at IXPs nowadays.

2.1 Passive monitoring

2.1.1 Simple Network Management Protocol

Most of the IXPs rely on passive monitoring using Simple Network Management Protocol (SNMP). SNMP [7] is a standardized protocol used for collecting and organizing operational switch data and per port information produced by monitoring tools. SNMP gathers information of the counters per port (bits and packets), port status (up/down), Central Processing Unit (CPU) load, memory utilization, temperature, etc. To gather information about the current utilization of each link counters of member and backbone interfaces can be periodically polled as well. All that information is centrally gathered into a monitoring database. A central database allows

[1] `https://github.com/inex/IXP-Manager/wiki`
[2] `https://www.inex.ie/ixp/index/about`

the IXP operator to deploy so-called triggers on the datasets. Triggers can conduct multiple individual values and evaluate them. If a value exceeds a predefined threshold, the monitoring system can issue certain actions, such as sending a notification to an engineer on-call.

Error counters are also polled for an early link failure detection. The same applies for link aggregation groups, which are widely used within an IXP network to scale both backbone and member port capacities.

To provide supervision information to the IXP operator or the IXP member, a variety of SNMP client tools exist. These (graphing) tools are used by most IXP as they are combine in a single view the service performance with the traffic statistics. The main SNMP graphing tool is the Multi Router Traffic Grapher (MRTG). MRTG can be easily integrated on the public IXP website to show the total volume of traffic exchanged by the IXP fabric.

2.1.2 Packet sampling - the sFlow approach

The sFlow [18] standard provides switches with the capability to passively measure Layer 2 to 4 traffic [2].

With sFlow, monitoring is carried out in the form of packet sampling [17]. This effectively means that one out of every N packets on each input port is captured and used for extracting measurements on the total traffic flowing through the switch. A copy of the captured packet is handled to the control plane CPU, which creates a datagram with the header information and extra metadata. This is in turn forwarded to the monitoring controller via the control plane. The extra metadata information includes the switch's ID, the sampling rate used when the packet was captured, the output port(s) selected, along with associated port counters. This way, both flow and counter monitoring is performed with the same method.

The rate of samples created by sFlow equals the packet rate divided by N. Thus, giving the monitoring controller the ability to probabilistically calculate the number of bytes and packets per flow through simple scaling [17]. Assuming fixed packet sizes, the expected error of simple scaling is quantifiable and has been calculated to be inversely proportionate to the square root of the number of samples gathered for a specific flow. More specifically, the error (in percent) can be estimated as $\textit{percent error} \leq 196 \times \sqrt{\frac{1}{s}}$, where s is the number of samples. Thus, in order to decrease the estimated error, the number of samples per flow has to increase. This can be done in one of only two ways: by increasing the estimation time window, or increasing the sampling rate. The first one is less practical, because it would increase the latency of measurements, violating the real-time monitoring principle

needed in modern networks. The ability to increase the sampling rate is however one of sFlow's largest drawbacks. As the switch CPU is tasked with creating the monitoring datagrams, it can be easily overwhelmed on switches with a large number of ports by a high sampling rate. In fact, studies have shown that even on state-of-the-art switches the sample rate peaks at approximately 350 samples per second [22].

Packet sampling as a monitoring practice has several advantages: allows monitoring of both flow and port counters at the same time, scales well with wire speed owing to its probabilistic component, and is supported by multiple manufacturers and monitoring systems[3]. However, its main drawbacks are its lack of programmability and its dependence on the switch CPU (in the sFlow embodiment), which in consequence cannot maintain the monitoring throughput needed for real-time measurements.

2.1.3 Flow counters - the NetFlow approach

NetFlow is a de facto industry monitoring standard (whose standarized version is IP Flow Information Export (IPFIX)) initially developed by Cisco to provide switches the ability to collect statistics about individual Layer 3-4 flows. Most commonly, a packet is identified as belonging to a specific flow via its 5-tuple (source and destination Internet Protocol (IP) address, source and destination port number and IP protocol number).

Implementing NetFlow in hardware requires a dedicated Content-Addressable Memory (CAM) acting as a flow cache to track information about specific flows. As each packet arrives, its 5-tuple is checked against the entries of the CAM, and if it matches any of them, the entries statistics are updated. In the opposite case, a new entry is created for the new flow. Information is extracted from the CAM cache in one of 4 occasions:

- A Transport Control Protocol (TCP) packet with a FIN or RST flag indicates that the flow is completed.
- A flow idle timer expires.
- A hard timeout fires, indicating that an update for a flow should be extracted even though it is still active.
- The cache is full and a new entry needs to be written.

When one of these conditions hold, the switch sends a NetFlow record including the flow's extracted measurements to the monitoring controller through

[3]See: `http://sflow.org/products/index.php`

the control plane. The measurements include information such as number of bytes and packets, timestamps for the flow's start and finishing times, Layer 3 headers, Border Gateway Protocol (BGP) routing information, Multiprotocol Label Switching (MPLS) labels (version 9 only), etc.

From version 5, NetFlow also provides a "sampled NetFlow" mode [9] to reduce the overhead of monitoring on the switch. In this mode NetFlow records are created based only on 1 in N packets that arrive to the switch instead of every packet.

While NetFlow is a scalable solution to flow monitoring, in practice it cannot be utilized for real-time monitoring as it is not programmable and exclusively push based. Timeouts are specified at a granularity of seconds and hence NetFlow provides little to no latency advantage over the low polling rates of OpenFlow's per-rule counters [12, 21].

2.1.4 Port mirroring

Port mirroring is a mechanism for monitoring and data analysis supported by most modern switches. It entails the capability to copy all traffic coming through a subset of the switch's ports and forward it to one or more monitoring ports for telemetry and data analysis. Therefore, this mechanism can be handy in monitoring the data plane of the IXP's switches.

This simple mechanism can provide information on the monitoring controller very fast (on line rate), as it requires almost no effort on part of the switch. The switch CPU is not involved in the mirroring process, other than performing the configuration specified by the network management entity and as such all filtering, information extraction and data processing is delegated to the monitoring controller on the other side of the monitoring port.

Passive monitoring with port mirroring has been proposed by Rasley et al. [19]. In this approach, multiple (or all) switch ports are mirrored to a single or a small number of monitoring ports. Naturally, the monitoring ports capacity will be exceeded and thus mirrored packets will be dropped. This way, each mirrored packet has a probability to be dropped and the ones that are actually forwarded to the monitoring entity can be considered to be sampled from their respective flows. The resulting monitoring mechanism has similar capabilities to packet sampling with a variable (and not controllable) sampling rate, which however always runs at wire speed and does not depend on the switch CPU, resulting in a 10x speedup over sFlow's monitoring speed [19]. The downside of port mirroring is its price in switch ports that need to be reserved for monitoring. As the number of

ports per switch and the throughput of links increase, maintaining a reasonable packet sampling rate would require increasingly many ports of the switch to be used for monitoring.

2.2 Active monitoring

IXPs frequently complement their passive monitoring capabilities with limited active measurements. For instance, DE-CIX utilizes Smokeping [20] and AMS-IX relies on ITU-T Y.1731 [14, 1] to send periodic ping requests to each device within the IXP network, including switches, route servers, and member routers. These tools enable the IXP operator to get insights in latency information across the network and allows the early identification of possible bottlenecks, either on the network layer or at a host level (e.g., router server). While these solutions address some of the limitations of today's monitoring their precision is rather low as the measurements are in the order of milliseconds (ms) far from the high speed monitoring goals of ENDEAVOUR.

2.3 Future directions in monitoring

We have presented so far the most prominent state-of-the-art monitoring methods used for extracting information at IXPs with legacy switches. We now look at the future opportunities in monitoring that new technologies are enabling. In particular we look at how can the Quantized Congestion Notification (QCN) standard be repurposed to achieve real-time monitoring (Section 2.3.1) and how can ENDEAVOUR use SDN switches to achieve its goals (Section 2.3.2).

2.3.1 Quantized Congestion Notification

The QCN standard (IEEE 802.1Qau) [3, 4] has been developed to provide congestion control at Layer 2, as a part of the IEEE Data Center Bridging Task Group's efforts. A congestion control scheme aims at signalling sources that send traffic through a network bottleneck (filled up / congested network buffer), before the congestion spreads and creates secondary bottlenecks.

QCN achieves that by moving congestion detection as close to the oversubscribed link as possible: the Congestion Point (CP), i.e. the switch whose queue(s) are congested, is responsible for signalling the congestion's source(s) (Fig. 1). As such, the switch is responsible to monitor its own output queues and to detect whether any of them is becoming congested. For each queue, the switch calculates a congestion measure F_b and with a

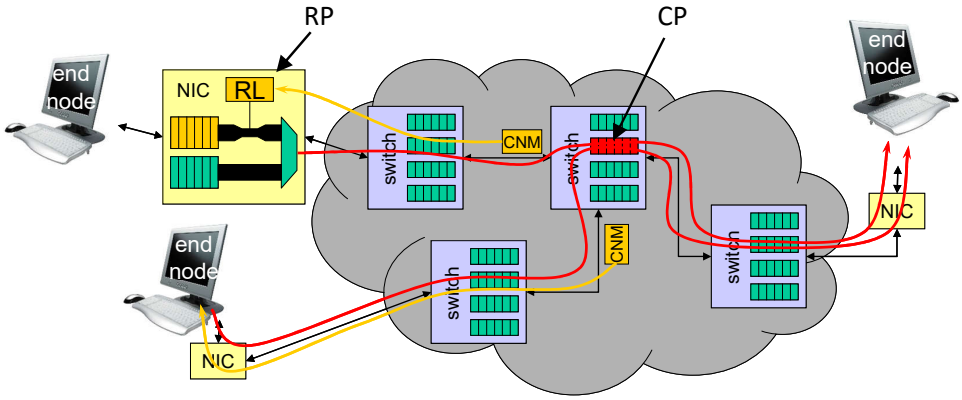

Figure 1: The QCN standard creates a closed loop: when a CP is detected, the end-nodes that participate in the congestion (the RPs) receive negative feedback in the form of CNMs and decrease their traffic based on the RL algorithm.

probability depending on the severity of the congestion, randomly samples an incoming packet and sends the value of F_b back to the packet's source inside a Congestion Notification Message (CNM) (Fig. 2). The value of F_b is calculated as follows (quantized to 6 bits): $F_b = -(Q_{off} + wQ_\delta)$, where w is a constant, Q_{off} is the difference between the queue's current occupancy and a predefined threshold (Q_{eq}, usually $\approx 20\%$), Q_δ is the derivative of the queue's occupancy, calculated as the difference of the current occupancy level and the level at the time that the previous notification was created. The resulting value of the congestion measure captures a combination of the queue size (Q_{off}) and the queue size rate of change (Q_δ), with a negative value indicating oversubscription on the buffer, the link, or both. The sampling rate in turn depends on the value of F_b. It equals 0 for positive F_b values, varies between 1% and 10% while F_b has close-to-0 negative values, and is constant and equal to 10% for even smaller values. Along with the F_b value for the queue, CNMs also contains a leading chunk of the sampled packet with its header, that is used by the receiver to determine the flow causing the congestion. The receiver of a CNM (the Reaction Point, RP) implements a rate limiting algorithm that decreases its sending rate based on the received feedback (the F_b value) and increases its rate voluntarily after a predetermined time interval to probe the network for extra bandwidth. The details of the Rate Limiter (RL) algorithm are beyond the scope of this deliverable, but are very comprehensively presented in [4].

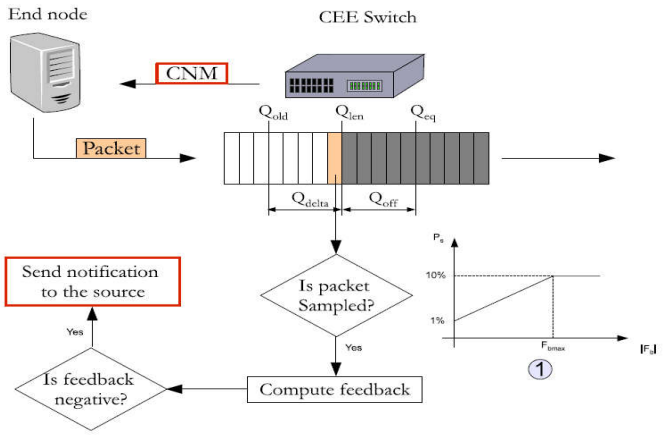

Figure 2: For each queue, a value F_b is calculated, depending on both the current occupancy levels (Q_{off}) and rate of change of these levels (Q_δ). In the case that this value is negative, incoming packets are sampled with the sampling probability depending on the F_b value and a CNM is send to the reaction point.

Repurposing QCN. QCN's principles have the potential to be very effectively repurposed for monitoring while reusing already available hardware of modern switches and surpassing the limitations of state-of-the-art methods. Reasons for that: all of the network's queues are simultaneously monitored, in sub-us speeds (possibly with the arrival of every packet), the CNMs travel through the network in the data plane (at wire speed), and also contain most of the data needed for sufficiently monitoring a switch: queue size, the rate of change of the queue size, and enough header information to infer the flow to which the packet belongs. Another advantage of QCN in monitoring is that it is scalability, as the notifications/monitoring data are by construction distributed to the edge of the network.

Although QCN was designed for congestion management, in the monitoring context it can be adapted for use in the absence of a closed loop, i.e., without RP rate limiters and without a TCP-like primal-dual algorithm. Instead, the CP sampling process could be substantially simplified to increase in speed, while also the subsequent results injected as new packets (CNMs) can be pre- and post- processed and aggregated, e.g., with filtering and compression in the switch (or in an offload engine). This can contribute to the construction of a coherent bitmap - or heatmap of the current state of all output queues in the network (Fig. 3), adding to the network's centralized

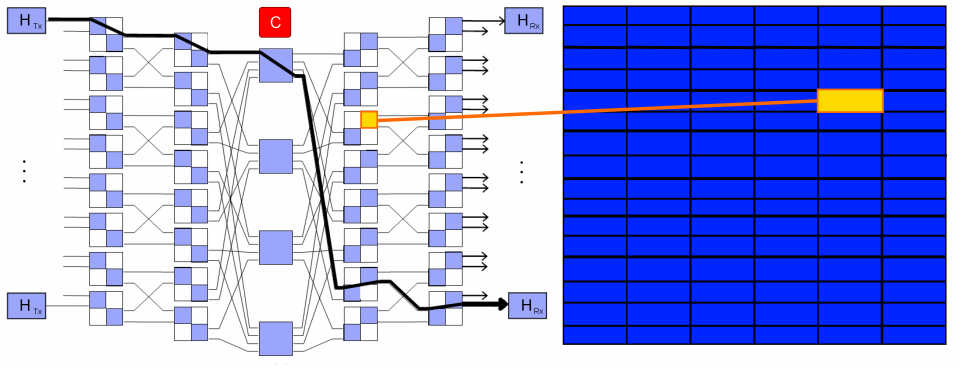

Figure 3: CNMs can be used to make perceptual "heatmaps" of the state of the Layer 2 network: each pixel of the heatmap image represents the state of a single output queue, with its color encoding the respective queue size (Q_{off}) and queue size rate of change (Q_{δ}).

SDN controller's global view of the network state. Thus, the SDN controller is better equipped when making purposeful decisions in fields beyond congestion avoidance, namely (but not limited to) load balancing, traffic engineering and security.

Unfortunately, QCN is not available for monitoring at its current form. The main obstacle is that notifications are not timestamped and therefore an accurate representation of the system state is not possible at this moment. Anghel et al. [5] describe how to solve this problem by synchronizing switches and adding a timestamp to the feedback. The notifications are then used to create time-space correlated heatmaps of the network state at the granularity of 10s of us (simulation only). Another limitation is that QCN operates on Layer 2, and as such it is hard for CNMs to be routed back to the source of the sampled packet. A potential solution to this problem would be for the feedback to not be sent back to the source of the congestion, but to a predefined, pre-routed set of monitoring entities tasked with aggregating the monitoring data of the network.

2.3.2 OpenFlow Switches

SDN switches are meant to be programmable by an external controller through an Application Program Interface (API). The most well-known SDN API, also known as southbound interface, is the OpenFlow protocol [15].

OpenFlow switches must be able to match many protocols fields from Layer 2 to Layer 4. Moreover, every flow might be able to keep a record of traffic counters. This flexibility to combine different protocols fields plus the flow statistics gives the power to define the granularity level of monitoring. It is an advantage when compared to the low granular tuple defined by NetFlow: while NetFlow only allows to monitor IP addresses and transport ports, OpenFlow enables a much more comprehensive view.

Flow statistics in an OpenFlow network are typically collected by the controller at operationally defined rates. It raises the interesting possibility for the controller to react according to the network state. One example is the detection and handling of Distributed Denial of Service (DDoS) attacks [6]. When a large flood attack is detected by a flow monitoring application, the controller can change the current flow rules to black hole the attack related traffic.

While OpenFlow is an elegant solution for traffic monitoring, it might be limited by the hardware and software of the available switches. Notably, counting the number of packets and bytes per flow is a costly operation. For this reason, in the newest OpenFlow specifications, the only required counter is for the duration of the flow. Packet and byte counters, much more interesting from a monitoring perspective, are just optional features.

Overall, for performance issues, OpenFlow switches tend to support NetFlow and sFlow [13]. One example is the Open vSwitch (OVS), a software switch that can be used as hypervisor bridges or as the control stack for hardware switches.

2.4 Limitations

The existing monitoring mechanisms are limited in many regards. The previous deliverable [8] already exposed some of those limitations focusing on the monitoring that could not be achieved with legacy switches due to its conceptual design. Here we continue that work and discuss the main limitations that stem from the current implementations of the state-of-the-art monitoring at IXPs.

The critical limitations of today's monitoring in legacy switches are its lack of programmability and universal deployability, the inability to make targeted measurements and the restricted capacity to for active monitoring.

The current monitoring methods and tools lack programmability and hence depend on network operators. The lack of programmability makes impossible an automatic reaction to network events such as a security threat or congestion in a peering link. As the current technology is conceptually

incapable of providing those reactive solutions this feature is practically impossible in today's IXPs. Introducing SDN at the IXP is the response of ENDEAVOUR to that critical limitation.

Additionally, the type of targeted monitoring necessary to enable the use cases previously described [8] is not possible in the destination based approach (IP based) of today's switches. This limitation impedes, for instance, monitoring per physical port, fundamentally limiting the monitoring of load balancing, traffic engineering, traffic steering or novel peering scenarios as discussed in previous deliverables. The SDN ability to look at any packet field overcomes this shortcoming and allows for targeted monitoring of the traffic flows.

Besides the proprietary solutions typical of nowadays approach to IXP monitoring do not enable the universal monitoring capabilities we aim for due to limited hardware support.

Another fundamental limitation is that today's network monitoring is eminently passive. While counter based information statistics are retrieved very frequently (e.g., every 5 minutes) they just inform of the number of packets and the distribution of its size. Generally, analysis of the sampled data is only ex post and requires large storage capacity and processing power to search through the historical sampled flows. The permanent growth in IXP's traffic volumes further exacerbates this problem.

To make things more complex, the monitoring data does not provide a complete picture of the events taking place within the switching infrastructure. Firstly, because monitoring relies on sampling, statistical modelling of the sampled data is necessary. Furthermore, since the data is not comprehensively sampled through the switching infrastructure it is impossible to follow flows or even packets through certain paths in the IXP network.

IXPs frequently complement their monitoring with active measurements such as smokeping or ITU-T Y.1731 to alleviate the pitfalls of exclusively passive monitoring (e.g., inability to measure path delays). However despite of these mechanisms it remains challenging to measure accurate delays of individual paths. Furthermore, active monitoring will not only be valuable in implementing specific use cases, ENDEAVOUR will strongly depend on active monitoring capabilities during the development phases in order to debug the network. The existing solutions unfortunately do not have the granularity necessary for that goal.

While the future directions in IXP monitoring described in Section 2.3 are promising they have also substantial shortcomings. To date, QCN monitoring has not been implemented in practice due to the limitations earlier described. Adding these capabilities remain very challenging within the

ENDEAVOUR timeframe and budget, given:

1. The lack of QCN, sFlow or other sampling mechanism in the current generation of OpenFlow/SDN switches such as Corsa [11] and NoviSwitch [16].

2. The absence of sampling methods in current OpenFlow versions (sampling requiring both a new OpenFlow framework and hardware offload support).

3. The limited SDN/OpenFlow support in the dominant Ethernet fabrics today, e.g., based on Broadcom's Trident Application-Specific Integrated Circuit (ASIC) family [10].

On the other hand, while the SDN concept has a huge potential existing commercial SDN switches are still in an early stage and the hardware support of some of the functionalities is still rather limited. Due to time frame limitations and the inherent complexities of hardware development, developing an ENDEAVOUR switch that complies with all our requisites is beyond the scope of the project. Nevertheless, through our interactions with the industry we do expect to influence hardware development so it will consider some of our desired functionalities. While those enhancements are unlikely to take place during the development phases of the ENDEAVOUR project, we believe they will benefit the networking community as a whole.

3 ENDEAVOUR monitoring architecture

To match as closely as possible the monitoring requirements previously described [8] we need a highly flexible monitoring scheme capable of both active and passive measurements and that can be supported everywhere.

In achieving this, we will need to depart from the state-of-the-art monitoring as described in the previous sections to overcome its limitations, e.g., closed-source/proprietary solutions, high costs, lack of flexibility, programmability, etc..

This section presents ENDEAVOUR basic monitoring architecture and discusses how our future work will integrate the different elements that compose it. Figure 4 depicts the schematic architecture of our monitoring platform in the broader architecture of ENDEAVOUR. The basic monitoring architecture includes three elements: an SDN switch, light middleboxes and a controller. The light middleboxes complement the monitoring capabilities of the SDN switch, whereas the controller coordinates them both, receives

information from the route server, and interacts with the ENDEAVOUR controller. The software component of our monitoring middleboxes communicates with the ENDEAVOUR controller and provides programmatic capabilities to the architecture by reconfiguring the hardware to meet the controller requests. In turn, the hardware performs the monitoring actions, i.e., timestamping, packets checking and parsing and filters the traffic so as to communicate to the software only the traffic of interest.

Note that one of the advantages of the monitoring middleboxes is the easiness of its deployment. As such they will be deployed as needed (i.e., not necessarily in the specific locations depicted in Figure 4). Additionally we will explore the possibilities of augmenting the switch monitoring capabilities through QCN repurposing techniques, as described in Section 2.3.1. Specifically we will explore whether they can be implemented in the planned middlebox, or/and in a dedicated switching platform using Original Equipment Manufacturer (OEM) ASICs and their respective API

Our monitoring architecture *decouples the monitoring from the switching infrastructure*. This design allows us to overcome the limitations of existing commercial SDN switches (see Section 2.4) by complementing their capabilities with lightweight middleboxes.

The separation of the monitoring and the switching infrastructure thus reinforces each other without creating unnecessary dependencies in the following manner:

- **Reduced limitations and overhead.** The SDN approach of our monitoring design provides highly flexible monitoring capabilities. The controller can request specific actions from a middlebox, e.g., gathering measurements, or redirecting a flow to a middlebox for further analysis. Monitoring can be hence scaled on the fly to meet specific needs. Thus eliminating unnecessary overhead on the whole monitoring system. In this sense, our design allows the controller to move from coarse grain to a fine grained monitoring on demand.

- **Active measurement capabilities.** By deploying our middleboxes in locations that would otherwise witness no traffic flows, we can make targeted active measurements, e.g., by generating ad hoc traffic flows and monitor them. The fact that no OpenFlow based switch has active measurements capabilities is a critical reason for including middleboxes in our monitoring architecture as it will allow for active debugging both in the development phase as well as in latter stages.

- **Future proof.** The ability to place middleboxes in the locations of

choice of the switching infrastructure will not only allow ENDEAVOUR to cover a wide range of the monitoring requirements described in previous deliverables [8], but also to adjust the monitoring capabilities to new requisites. In this sense, by decoupling the monitoring and the switching fabric we provide a greater ability to both infrastructures to adapt to future developments. Untied to the switching fabric architecture, new middleboxes can be placed in any location of the switching fabric. Furthermore, if new monitoring needs require enhanced capabilities, the middleboxes can be improved independently from the switching fabric. As a result the monitoring and the switching infrastructures can evolve independently.

- **Backwards compatibility.** The separation between the monitoring and switching infrastructures facilitates a backwards compatible deployment of a monitoring system. While SDN in principle enables comprehensive monitoring capabilities, the state-of-the-art SDN switches typically falls far from being that extensive. In this sense, our software defined middleboxes monitoring architecture can be deployed on a switching fabric that uses existing SDN hardware to improve the monitoring capabilities of the system without having to deal with the cost, complexities and risks of developing new switching hardware.

3.1 Monitoring requirements

This section discusses how the monitoring requirements could be fulfilled by our envisioned architecture and Table 1 summarizes them. Our middleboxes have the monitoring capabilities necessary for each of the use cases previously analyzed [8]. While in our envisioned scenario with an SDN switch some of this functionalities might be implemented at the switch level rather than using the middleboxes, the middleboxes not only enable extra new uses but will be critical in the development phase due to their active debugging capabilities.

While monitoring load balancing, traffic engineering monitoring or advanced peering forms can be –in principle– implemented using an SDN switch, placing our middleboxes at the right location allows us to provide the required monitoring capability as well as perform active measurements. For instance, find the most convenient paths –latency wise, for instance– to optimize traffic engineering. In the early stages of the development phase, active debugging of the network will be a critical functionality.

In the case of Routing As a Service (RAS), outsourcing the routing

decision to the IXP operator implies that the IXP has to keep record of every member routing table. While currently it is possible to monitor the routing and forwarding information base of network equipments there is no monitoring application to view the global state of the tables. While in principle an SDN controller could achieve such full vision, it might become problematic number of members implementing RAS is large enough. Our monitoring middleboxes are a simple manner to alleviate this problem.

Furthermore, some other use cases do need our middleboxes in order to be feasible. For example, in the case of overlay monitoring, thanks to the reconfigurable properties of our middleboxes they can easily track new protocols and match different encapsulation layers. This is a critical functionality for overlay monitoring that can be hardly implemented by exclusively relying on an SDN switch.

The existing SDN switches do not include the reactive capabilities we aim for and are hence not capable of automatically altering the forwarding behaviour, for instance to react to a detected security threat. Instead our middleboxes can capture the whole packet for further inspection, thus enhancing the security capabilities. Moreover, in order to detect different types of security threats new rules have to be implemented in the SDN switch, saturating the capacity of the flow table. Hence, our separation of the monitoring from the switching infrastructure reduces the limitations inherent to the latter. For instance, the amount of rules necessary to detect a SYN flood, a form of Denial of Service (DoS) attack, could overwhelm the size of the flow table. By relying on our middleboxes, we can detect this attack without suffering the limitations of existing SDN switches.

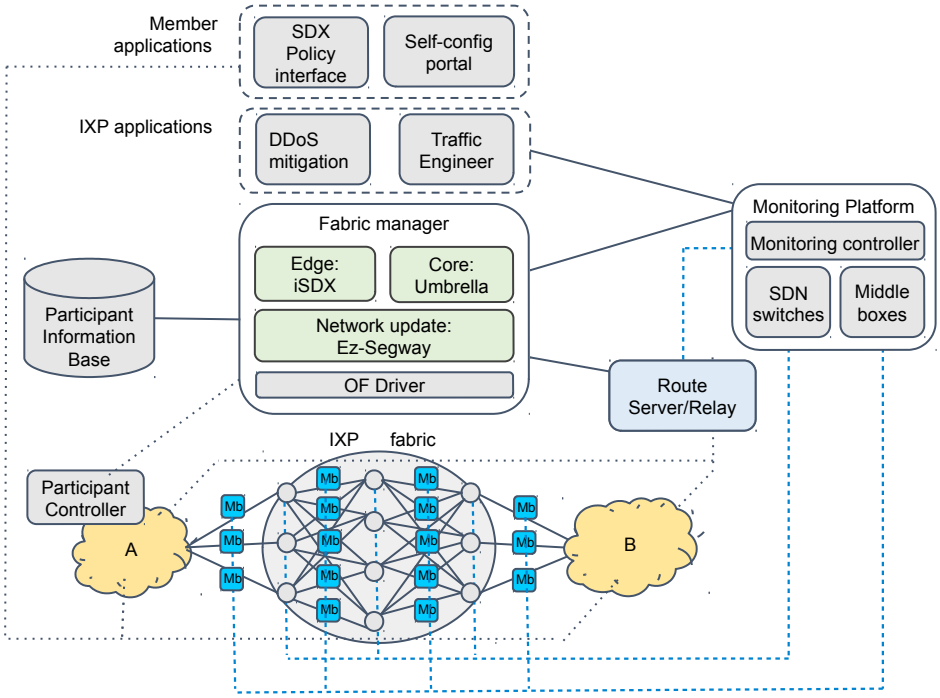

Figure 4: Monitoring architecture

Use cases	Monitoring requirements		Monitoring capabilities	
	Control plane	Data plane	Primitives	SDN switch
Load Balancing	✗	traffic volume per physical port	✓	✓
Traffic engineering	✗	traffic volume per physical port	✓	✓
Peering	control/data plane consistency	traffic volume per physical port	✓	✓
Overlay	changes in routing entries	Forwarding Information Base (FIB)	✓	limited
Security	control/data plane consistency	contingent on the specific security aspect	✓	limited
Traffic steering	BGP announcements	traffic volume per physical port	✓	✓
Routing As a Service RAS	-FIB (routes) -convergence time	-consistency with FIB -topology	✓	limited

Table 1: Monitoring requirements and middleboxes per use case

3.2 Monitoring middleboxes

To achieve the aforementioned goals ENDEAVOUR will look for lightweight middleboxes similar to Open Source Network Tester (OSNT). OSNT is an open hardware/software co-design for traffic generation and monitoring based on the NetFPGA card. The combination of traffic generation and traffic monitoring capabilities into a single FPGA-equipped device allows a per–flow characterisation of a networking system or a an entire network regarding end-to-end latency, jitter, packet-loss, congestion, etc.

In the reminder of this section we look at the main characteristics from OSNT that we envision for the ENDEAVOUR middleboxes.

Hardware and software task division. While the hardware side of OSNT provides all the means for a fast and accurate packet processing, the software side provides APIs for an easy hardware management. In this fashion, an user can adapt the hardware to his needs or take advantage of the existing APIs to create a software application that instruct OSNT on what packets generate/receive.

Monitoring subsystem. The OSNT traffic monitor subsystem provides functions for packet capturing, hardware packet filtering (*i.e.*, only the traffic-of-interest is sent to the host), high precision, accurate, packet timestamping and high-level traffic statistic gathering.

Figure 5 illustrates the architecture of the monitoring subsystem in OSNT. The 5-tuple (protocol, IP address pair and Layer 4 port pair) extraction is performed using an extensible packet parser able to recognize both VLAN and MPLS headers along with IP and IP encapsulation. Further flexibility is enabled by extending the parser implementation-code as required. A module instantiated immediately after the physical interfaces and before the receive queues timestamps incoming packets as they are received by the hardware. The design is an architecture that implicitly copes with a workload of full line-rate per port of minimum sized packets. However this will often exceed the capacity of the host, e.g., regarding its processing or storage capabilities, or may contain traffic of no practical interest. To this end OSNT implements two traffic-thinning approaches:

- 5-tuple filter: only packets that are matched to a rule are sent to the software, while all other packets are dropped.
- packet cutting: optionally, it is possible to record a fixed-length part of each packet (sometimes called a *snap-length*) along with a hash of

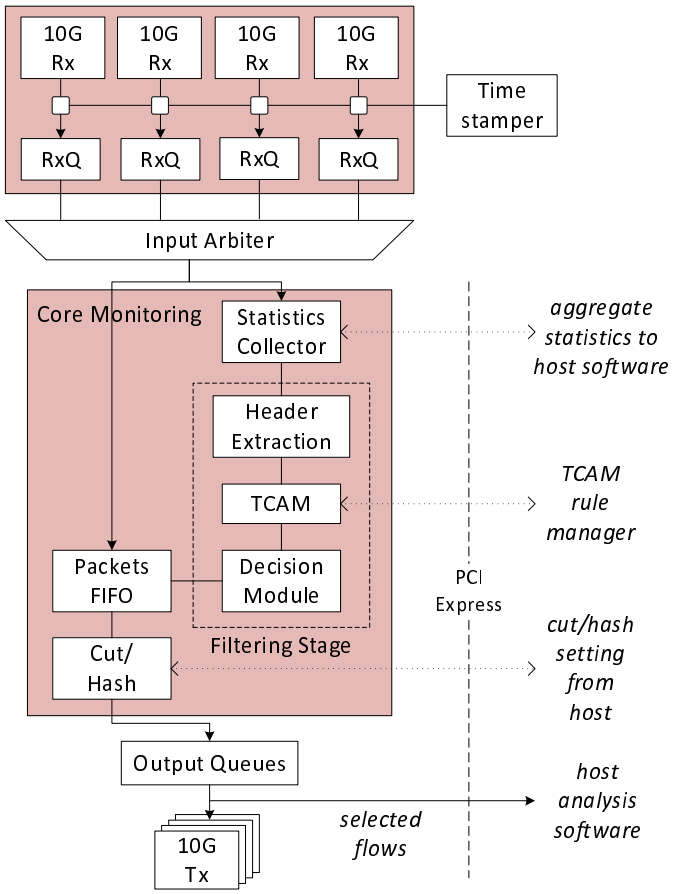

Figure 5: The architecture for OSNT traffic monitoring subsystem.

the entire original packet.

Providing an accurate timestamp to (incoming) packets is a critical objective of the traffic monitoring subsystem. Packets are timestamped as close to the physical Ethernet device as possible to minimize FIFO-generated jitter and permit accurate latency measurement. A dedicated timestamping unit stamps packets as they arrive from the physical (MAC) interfaces. Motivated by the need to have minimal overhead while also providing sufficient resolution and long-term stability, OSNT uses a 64-bit timestamp divided into two parts. The upper 32-bits count seconds, while the lower 32-bits provide a fraction of a second. Integral to accurate timekeeping is the need to correct the frequency drift of an oscillator.

Generator subsystem. The OSNT traffic generator subsystem is designed to generate full line-rate per card interface, scalable so that it allows for multiple traffic generators to work in parallel within a single environment. Figure 6 illustrates the high-level architecture of the traffic generation

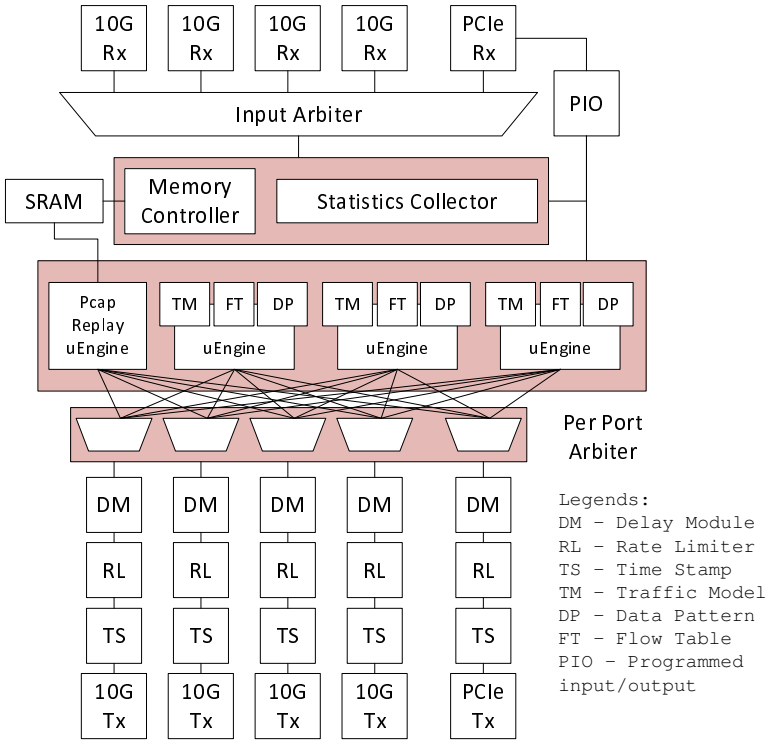

Figure 6: The architecture for OSNT traffic generation system.

pipeline. The key idea behind is to provide a large degree of modularity envisioning a set of micro-engines, each used to support one or more protocols at network and transport-layers such as Ethernet, TCP or User Datagram Protocol (UDP) and application-protocols such as BGP. Each micro-engine either generates synthetic or replays captured traffic (the actual feature being implemented) for one or more of the selected egress interfaces.

4 Outlook

So far we have looked at the state-of-the-art monitoring at IXPs and its limitations with regard to our envisioned monitoring capabilities. In partic-

ular we examined packet sampling based monitoring with sFlow (see Section 2.1.2), the flow counters approach of NetFlow (see Section 2.1.3), and the port mirroring option (see Section 2.1.4), as well as the existing active monitoring solutions (see Section 2.2). We then explored the opportunities of technical solutions at the edge of the state of the art (see Section 2.3), i.e., repurposing QCN for real-time monitoring (see Section 2.3.1) and SDN switches (see Section 2.3.2).

Realizing of the limitations of the state-of-the-art, we proposed a monitoring architecture that integrates the virtues of an SDN switch, and complements the drawbacks of its existing commercial embodiments by deploying open sourced lightweight flexible middleboxes in the switching fabric. Additionally QCN repurposing techniques (see Section 2.3.1) have the potential to enhance our capabilities with real-time monitoring.

In the near future we will work toward integrating the different elements of our monitoring architecture as described in Section 3. The next critical step is developing a monitoring controller capable of managing and coordinating our monitoring middleboxes and the SDN switch. An API will also allow this controller to receive information from the route server. Ultimately, the monitoring controller will interact with the fabric manager to feed the applications (both for members and IXP operators) with the necessary monitoring data.

5 Acronyms

SDN Software Defined Network

BGP Border Gateway Protocol

IXP Internet eXchange Point

SLA Service-Level Agreement

IP Internet Protocol

DE-CIX German Commercial Internet Exchange

AMS-IX Amsterdam Internet Exchange

RAS Routing As a Service

DDoS Distributed Denial of Service

DoS Denial of Service

ms milliseconds

DNS Domain Name System

FIB Forwarding Information Base

TCP Transport Control Protocol

UDP User Datagram Protocol

OSNT Open Source Network Tester

SNMP Simple Network Management Protocol

MRTG Multi Router Traffic Grapher

API Application Program Interface

OVS Open vSwitch

QCN Quantized Congestion Notification

MPLS Multiprotocol Label Switching

CAM Content-Addressable Memory

CP Congestion Point

RP Reaction Point

IPFIX IP Flow Information Export

CPU Central Processing Unit

CNM Congestion Notification Message

RL Rate Limiter

INEX Ireland Internet Exchange Point

ASIC Application-Specific Integrated Circuit

OEM Original Equipment Manufacturer

References

[1] AMS-IX. `https://ams-ix.net/technical/statistics/real-time-stats`.

[2] Traffic Monitoring using sFlow. `http://www.sflow.org/sFlowOverview.pdf`.

[3] 802.1Qau - Virtual Bridged Local Area Networks - Amendment: Congestion Notification. Technical report, 2010.

[4] Mohammad Alizadeh, Berk Atikoglu, Abdul Kabbani, Ashvin Lakshmikantha, Rong Pan, Balaji Prabhakar, and Mick Seaman. Data center transport mechanisms: Congestion control theory and ieee standardization. In *Communication, Control, and Computing, 2008 46th Annual Allerton Conference on*, pages 1270–1277. IEEE, 2008.

[5] Andreea Anghel, Robert Birke, and Mitch Gusat. Scalable high resolution traffic heatmaps: Coherent queue visualization for datacenters. In *Traffic Monitoring and Analysis*, pages 26–37. Springer, 2014.

[6] Rodrigo Braga, Braga, Edjard Mota, Mota, and Alexandre Passito, Passito. Lightweight ddos flooding attack detection using nox/openflow. LCN '10, pages 408–415, Washington, DC, USA, 2010. IEEE Computer Society.

[7] Fedor Case and Davin Schoffstall. A simple network management protocol (snmp). Technical report, RFC 1157, 1990.

[8] Castro, Fernandes, Antichi, Gusat, Kathareios, Uhlig, and Dietzel. D.3.1: Monitoring. *ENDEAVOUR*, 2015.

[9] Cisco. Sampled NetFlow. `http://www.cisco.com/c/en/us/td/docs/ios/12_0s/feature/guide/12s_sanf.html`.

[10] Broadcom Corp. Broadcom BCM56850 StrataXGS Trident II Switching Technology. `https://www.broadcom.com/collateral/pb/56850-PB03-R.pdf`.

[11] Corsa Technology. Corsa Product Overview – DP64xx Data Plane Family. `http://www.corsa.com/wp-content/uploads/2014/11/Corsa-Product-Overview.pdf`.

[12] Andrew R Curtis, Jeffrey C Mogul, Jean Tourrilhes, Praveen Yalagandula, Puneet Sharma, and Sujata Banerjee. Devoflow: Scaling flow management for high-performance networks. In *ACM CCR*, volume 41, pages 254–265. ACM, 2011.

[13] K. Giotis, C. Argyropoulos, G. Androulidakis, D. Kalogeras, and V. Maglaris. Combining openflow and sflow for an effective and scalable anomaly detection and mitigation mechanism on sdn environments. *Comput. Netw.*, 62:122–136, Apr 2014.

[14] ITU. Oam functions and mechanisms for ethernet-based networks. Standard G.8013/Y.173108/15, ITU-T, 2015.

[15] Nick McKeown, Tom Anderson, Hari Balakrishnan, Guru Parulkar, Larry Peterson, Jennifer Rexford, Scott Shenker, and Jonathan Turner. Openflow: Enabling innovation in campus networks. *ACM CCR*, 38(2):69–74, Mar 2008.

[16] NoviFlow. NoviSwitch 2128 High Performance OpenFlow Switch datasheet. `http://noviflow.com/wp-content/uploads/NoviSwitch2128Datasheet.pdf`.

[17] Peter Phaal and Sonia Panchen. Packet Sampling Basics. `http://www.sflow.org/packetSamplingBasics/`.

[18] Peter Phaal, Sonia Panchen, and Neil McKee. Inmon corporation's sflow: A method for monitoring traffic in switched and routed networks. Technical report, RFC 3176, 2001.

[19] Jeff Rasley, Brent Stephens, Colin Dixon, Eric Rozner, Wes Felter, Kanak Agarwal, John Carter, and Rodrigo Fonseca. Planck: millisecond-scale monitoring and control for commodity networks. In *Proceedings of the 2014 ACM conference on SIGCOMM*, pages 407–418. ACM, 2014.

[20] Smokeping, 2015.

[21] Brent Stephens, Alan Cox, Wes Felter, Colin Dixon, and John Carter. Past: Scalable ethernet for data centers. In *CoNEXT*. ACM, 2012.

[22] Junho Suh, TT Kwon, C Dixon, W Felter, and J Carter. Opensample: A low-latency, sampling-based measurement platform for sdn. IEEE ICDCS, 2014.