# On the feasibility of converting AMS-IX to an Industrial-Scale Software Defined Internet Exchange Point


**University of Amsterdam**

*System and Network Engineering*
MSc Research Project

July 29, 2016

**Authors**

Siem Hermans
siem.hermans@os3.nl

Jeroen Schutrup
jeroen.schutrup@os3.nl

**Supervisors**

Ariën Vijn
arien.vijn@ams-ix.net

Joris Claassen
joris.claassen@ams-ix.net



## Abstract

Software Defined Internet Exchanges (SDX) promise to greatly increase the flexibility of performing fine-grained interdomain traffic engineering at Internet exchange points (IXP). This novel concept combines participant's traffic policies with Border Gateway Protocol (BGP) routing information in OpenFlow constructs in order to override commonly coarse BGP forwarding behavior. Recent advancements in flow rule compression algorithms have potentially made SDX applicable at an industrial scale (iSDX). However, with the constant and rapid growth of the AMS-IX platform, concerns regarding the practical scalability of iSDX remain present. In this paper, the deployment feasibility of iSDX with regard to the current Brocade MLXe switch platform is evaluated and a model for integrating iSDX in future iterations of the AMS-IX platform is presented. Subsequently, prior iSDX scalability claims are validated and expanded upon. Limitations regarding the initial experiments are identified and more extensive experiments are conducted. This report reveals that the currently employed switch platform is incapable of supporting iSDX due to the lack of sufficient flow tables. Although the policy compression algorithm used by iSDX has made great advancements with regard to prior iterations, practical scalability of the concept is still heavily constrained by current hardware capabilities. Currently, iSDX does not scale to the size of AMS-IX without severely constraining the IXP members in the number of policies they are allowed to define.



***Keywords*** – Industrial scale, Software Defined Networking, Internet exchange point, OpenFlow


## Contents

# 1 Introduction

The Border Gateway Protocol (BGP) features a relatively limited amount of possibilities for traffic engineering (TE) [1, 2]. More specifically, inbound traffic engineering is currently limited to attributes such as the Multi-Exit Discriminator (MED), Autonomous System (AS) Path prepending and BGP Communities. However, these preferences only affect aggregate traffic and don't have to be respected by the upstream BGP peer. As such, it can be challenging to define and enforce policies between neighbouring ASes. Furthermore, due to lack of fine-grained traffic engineering mechanisms it's even harder to setup large scale application specific peering.

Feamster et al. [3] aim to alleviate these problems by enhancing existing Internet exchange points (IXP) with Software Defined Networking (SDN) in order to create 'Software Defined Internet Exchange Points' (SDX). SDN is a relatively new paradigm which primarily allows for programmatic control of the network, centralized network management and increased flexibility. However, due to the wide variety of interpretations of this technique and the rapid adoption by major network vendors, SDN has evolved from being a promising academic concept to an ambiguous buzzword. With SDX, a return is made to the original concept of SDN, in which the control plane of the network is placed in a centralized controller. For this purpose SDX leverages the standard BGP route server at an IXP which is augmented to incorporate an SDN control plane. Each participant of the IXP is then provided with an isolated segment of the controller in which it can define its own fine-grained interdomain traffic policies.

SDX as a whole is transparent to the participants of the IXP. Members of the IXP already exchange topology information with the route server. Additionally, as each participant has its own share of the SDN controller, they can define traffic policies in their slice of the IXP. Subsequently, these interdomain policies are combined with BGP route information, validated, and translated into corresponding OpenFlow forwarding rules which are installed in the IXP switch fabric. This may change the way IXPs operate, as deploying SDX would allow participants more control over in- and outbound traffic paths within the IXP. Example use cases of such an IXP fabric are port-based application specific peering relationships, upstream Denial of Service (DoS) prevention or load balancing over the IXP.

Initial iterations of the SDX controller struggled with scalability as combining BGP route information with a large amount of policies caused the resulting set of forwarding rules to exceed current hardware capabilities. However, recent developments by Gupta et al. exhibit new methods for compressing forwarding table sizes by consolidating similar rules between participants[4]. These enhancements have been incorporated in an open-source 'industrial-scale' SDX controller (iSDX).

Although iSDX claims to be ready for deployment on at industrial scale IXPs, we find the scalability of this concept to be questionable. The evaluation of Gupta et al. hinges on a relatively limited amount of policy definitions between participants of the IXP[4]. Additionally, the scalability assessment is limited to 500 participants at most. At the time of writing the amount of route server peers at AMS-IX is rapidly approaching the 700 mark. As such we seek to validate these scalability claims at one of the world's largest IXPs. Additionally, we are interested in the implications of incorporating this novel concept in the platform of AMS-IX.

**Contributions**
The contributions of this research are twofold. Primarily, this paper presents the first deployment model for a large scale multi-hop iSDX infrastructure. Additionally, the practical scalability of the iSDX controller and its compression methods is validated for the environment of AMS-IX.

# 2 Research questions

The goal of this project is to answer the following research question:

> *What is the feasibility of converting the AMS-IX to an Industrial Scale Software Defined Internet Exchange Point?*

In order to further delimit the project, the main research question has been divided into the following sub-questions:

- What alternatives exist for defining enhanced BGP policies when peering at Internet exchange points?
- How would applying iSDX affect the infrastructure of AMS-IX's switch fabric?
- Can an iSDX environment scale at the same rate as the amount of AMS-IX members?

With regard to the scalability validation, this research project will exclusively assess the effect of iSDX policies on the amount of OpenFlow entries in the flowtable. The computation time required to perform policy compression and the effect of frequent BGP updates on the flows in memory are beyond the scope of this project. Further scoping of the project is discussed in more detail in Section 5.

# 3 Related work

Although the Border Gateway Protocol has historically had a number of shortcomings, it is still *the* de facto routing protocol used in the core of the Internet [1, 5, 6, 7, 8, 9, 10, 11]. Primarily the available methods for performing (inbound) traffic engineering in BGP have been limited. Various proposals and Requests for Comments (RFC) have been written [12] that aim to establish a more robust and predictable interdomain routing protocol with fine-grained traffic engineering capabilities [13]. Commonly, these capabilities are proposed in the form of extensions to the BGP protocol, while others propose a complete overhaul of the Internet backbone as we know it today. The Routing Control Platform (RCP) [8] for example aims to completely replace external BGP in the Internet backbone by creating a logically centralized platform. In summary, it allows for performing enhanced traffic engineering based on granular route-selection. Similarly, Interdomain Route Validation (IRV) is a more generic concept that primarily focuses on improving on BGP's security and can be extended such that it supports traffic engineering capabilities like load balancing [14]. FlowSpec on the other hand extends BGP rather than aiming to replace it. This extension allows peers to perform traffic steering based on transport layer semantics [2]. FlowSpec provides similar functionality to SDX and is therefore discussed in more detail in Section 4.2.

Due to the ossification of the Internet, introducing new protocols or implementing radical changes in existing protocols is hard if not impossible to achieve on a large scale [7]. As such, regardless of the potential gain in functionality, these and other proposed BGP enhancements never achieved far-reaching adoption [15, 16]. Internet exchange points however commonly form an innovative neutral party which interconnects large amounts of Autonomous Systems (AS). Their unique position in the Internet may allow for rapidly adopting novel concepts without burdening individual participants. Recently, deploying SDN at IXPs has been subject to research [17, 18, 19]. The AtlanticWave SDX for example is an experimental Software Defined Internet Exchange among a number of Universities in the United States [20]. Likewise, Google was one of the first organizations to deploy an all OpenFlow fabric at a commercial IXP in New Zealand, named 'project Cardigan' [21]. Additionally, research efforts have been put into investigating IXP Route Servers functionality and their role in the global Internet [22, 23].

In 2015, Gupta et al. [24] combined these prior research efforts in their design of a Software Defined Internet Exchange (SDX). SDX aimed to improve on Google's Cardigan project with regard to scalability. In their concept, extensive traffic engineering is enabled at IXPs by replacing the traditional Layer 2 switch fabric with an OpenFlow fabric and extending the IXP route server with an SDN controller. OpenFlow allows for performing traffic steering via means of granular match/action rules. The Toulouse Software Internet Exchange (TouSIX) has applied a similar concept in practice by replacing traditional protocols like the Address Resolution Protocol (ARP) and the Neighbor Discovery Protocol (NDP) with OpenFlow [25]. However, SDX extends further than that and allows participants to define interdomain traffic policies based on transport layer semantics. Besides improved traffic engineering capabilities, introducing OpenFlow on an IXP fabric opens up new possibilities for a wide range of use cases. For example, centralized controllers could allow for systematically detecting and eliminating BGP misconfigurations, which otherwise could potentially have led to route hijacks. Bailey et al. propose a validation step in which the IXP route server validates route announcements using Resource Public Key Infrastructure (RPKI) [26].

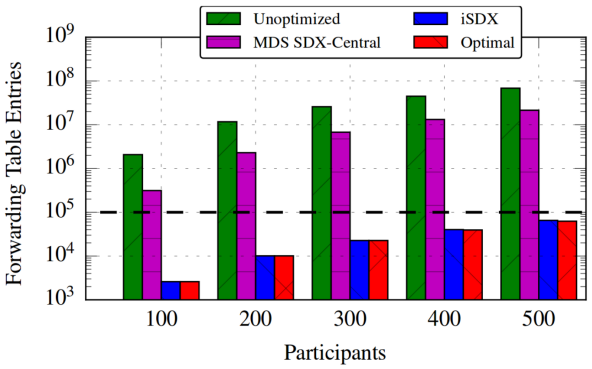


**Figure 1:** Prior scalability evaluation of SDX and iSDX up to 500 participants with varying compression algorithms [4]

Although SDX improved scalability in comparison to project Cardigan, the flow table size required for deploying SDX at an industrial scale still exceeded the limitations of current hardware capabilities. According to initial scalability evaluations of Gupta et al. as shown in Figure 1, scaling an SDX up to 500 participants with a relatively limited amount of traffic steering policies required over *10 million* unique flow entries in the switch fabric. Extensive research has been put into identifying scalability issues related to SDN and internetwork routing [27, 28, 29, 30]. One of the latest advancements in this field is the incorporation of a new flow rule compression algorithm in SDX. This would allow SDX to achieve an industrial-scale (iSDX) [4], whilst only requiring approximately 65000 unique flow entries for the same amount of participants

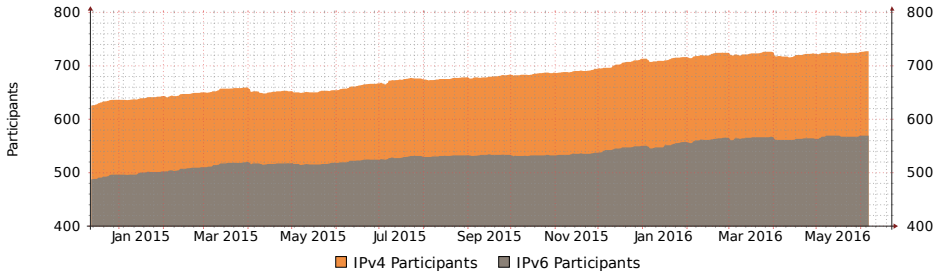


**Figure 2:** Gradual growth pattern of the amount of AMS-IX participants

and policies. According to Gupta et al. iSDX is currently ready to operate at the world's largest IXPs.

# 4 Technical overview

In order to set the stage for iSDX, this section will first briefly delve into the target infrastructure of AMS-IX. Subsequently, the technical concepts behind the controller and its components are discussed. Furthermore, the custom addressing scheme used by iSDX and the supported deployment formations are examined.

## 4.1 The AMS-IX platform

AMS-IX is a distributed Internet exchange point which provides peering services in multiple independent parts of the world. These services allow connected parties such as Internet Service Providers (ISPs) and Content Delivery Networks (CDNs) to exchange traffic between peers of the IXP without requiring –commonly costly– upstream transit. Additionally, peering through an IXP is commonly performed to increase the resilience of networks as the IXP can be used as a backup path.

**Growth pattern**
The AMS-IX has seen a significant growth in the number of participants in recent years. Over the course of the past one and a half years both IP versions have shown a steady growth pattern with an increase of circa 100 participants. Therefore, the applicability of iSDX at the AMS-IX is tightly coupled to the practical scalability of the provided controller. Gupta et al. [4] deemed SDX to have reached an industrial scale at 500 unique participants. However, as depicted in Figure 2, the current amount of peering IPv4 participants at the AMS-IX already greatly exceeds this number and the amount of IPv6 participants is expected to do so as well. Over the past years, AMS-IX has seen a continuous growth in traffic demands and the requirement for an increase in port capacity. In order to cope with this increase in demand, AMS-IX has periodically upgraded their infrastructure. Due to the spine-leaf topology of the fabric, the infrastructure allows for simple horizontal scaling.

**Services**
Although the demands for capacity have increased tremendously over the past years, the core components of an Internet exchange point have largely remained the same. In its simplest form an IXP consists of a network fabric and participant border routers [31]. In order to exchange traffic with other connected parties each participant connects to the underlying switch fabric with one or more ports. Then, it sets up either a bilateral BGP session with another participant or establishes a BGP connection to the IXP's route server. Primarily, AMS-IX provides a Layer 2 peering Local Area Network (LAN) which allows large amounts of independent Autonomous Systems (ASes) to exchange routing information via BGP. The current implementation of the peering platform is based on a Multi Protocol Label Switching (MPLS) and Virtual Private LAN Service (VPLS) infrastructure, which is deployed on top of Brocade MLXe series switches. This architecture allows for a resilient and highly scalable infrastructure.

**Route server**
A substantial amount of IXP participants establish an eBGP peering session with the route server and exchange routing information. The route server provides eBGP route reflection to all participants, effectively multiplexing the routing information to all peers. This is beneficial as it allow for creating an extensive peering matrix with all other participants without requiring a full mesh of $n(n-1)/2$ individual peering relationships [23].

## 4.2 Alternative technologies

Converting AMS-IX to an iSDX might require a fundamental change in the operational model of the IXP and the way in which the platform is built. Furthermore, incorporating the concept of iSDX would mean that the AMS-IX has to

start operating on higher network layers. From a neutrality standpoint it would be preferable if interdomain Traffic Engineering (TE) between participants remains transparent to the IXP. This section reviews FlowSpec its advantages and disadvantages as it is a potential alternative to iSDX.

**BGP FlowSpec**
As previously alluded to in Section 4, the most promising alternative to iSDX for performing fine-grained interdomain traffic engineering is the BGP Flow Specification (FlowSpec). This is an extension to the BGP protocol which adds additional Network Layer Reachability Information (NLRI) containing traffic flow specifications. Originally FlowSpec was designed to mitigate (distributed) denial-of-service (DoS) attacks. However, the flexibility of its match-action rules allows for extending this concept further, providing similar traffic engineering functionality as OpenFlow. Similar to Openflow matching, FlowSpec allows for matching traffic based on header fields such as source and destination addresses, L4 parameters and packet characteristics such as length, fragment sizes and so forth. These matching criteria can then for example be propagated through a route server to a large set of BGP peers. The receiving peers will then convert the received NLRI into Access Control Lists (ACL) or Policy Based Routing (PBR) entries. Subsequently, incoming traffic is filtered according to the specified rule set.

On an architectural level, FlowSpec is largely analogous to Software Defined Networks. A FlowSpec environment consists of a central controller, connected clients (BGP peers) and optionally a route-reflector for scalability. The controller component is responsible for sending or injecting the FlowSpec NRLI entry. The client (acting as a BGP speaker) receives that NLRI and programs its Ternary Content Addressable Memory (TCAM) to act on the instruction from the controller. Similar to iSDX, participants could be allowed to define policies in the central controller via means of a web portal or an Application Programming Interface (API). Validation of the individual policy definitions is handled by ensuring that the originator of the flow specification matches the originator of the best-matching unicast route for the destination prefix embedded in the flow specification [2].

**Advantages**
Deploying BGP FlowSpec over iSDX poses several significant advantages for AMS-IX. Primarily, as the intelligence is moved to the customer edge, the IXP does not need to interfere with semantics of layer three and above. This allows an IXP to remain in a neutral position. Another advantage of FlowSpec is the ease of implementation as BGP constructs are well known. However, in this scenario all participants on the IXP would have to support BGP FlowSpec on their edge routers. This is problematic as FlowSpec is currently exclusively supported by Juniper, Cisco and Arbor Networks[32].

**Limitations**
From an IXP perspective, FlowSpec is preferable over deploying iSDX. However, some technical limitations apply to FlowSpec. Contrary to OpenFlow, FlowSpec does not allow for pipelining multiple flow specifications. Instead, when multiple flow specifications match a certain Forwarding Equivalence Class (FEC), only the first matching FlowSpec rule will be applied. Another major downside is that the TCAM allocation for ACL, PBR and firewall rules in edge routers is commonly fairly limited. Cisco for example only supports up to 3000 flow entries in TCAM for its top of the line ASR edge router [33].

## 4.3 iSDX technical concepts

iSDX leverages OpenFlow to implement fine-grained policy definitions in the IXP fabric. OpenFlow was originally introduced as a method for researchers to experiment with new networking concepts. The standardization of the protocol by the Open Networking Foundation and the great interest of prominent network vendors has caused the protocol to gain major traction in a wide variety of environments. iSDX offers controller software which enables IXP members to perform traffic engineering on the fabric by leveraging OpenFlow.

**Main challenges**
Previous iterations of SDX and initiatives like Project Cardigan suffered from scalability issues due to their inefficiency in calculating and merging participants' policies. This resulted in a flow table becoming too large to fit in the TCAM of available hardware. Also, frequent updates could overload the fabric forwarding plane. Furthermore, computing the forwarding table could take hours. The main technical challenge iSDX aims to overcome is to perform outbound traffic engineering for aggregate prefixes, while preventing inflation of OpenFlow policies. Consider two IXP members, $A$ and $B$. Participant $A$ wants to steer HTTP traffic to $B$, even for prefixes for which the BGP path selection algorithm has decided $B$ is not the default next hop. Since $B$ should only receive traffic for prefixes it's advertising, a simple OpenFlow rule matching all HTTP traffic and outputting it to

$B$'s port on the traffic is insufficient. That is to say, this would cause B to also receive traffic for alien destinations. Prior initiatives would create a flow for every distinct prefix $B$ announces. That is, if $B$ advertises four prefixes, a separate flow for each prefix would be pushed to the fabric to satisfy participant $A$'s needs. As this approach entails scalability issues, iSDX improves on this by virtualizing Data Link and Network Layer reachability information.

**Virtual addressing scheme**
In order to overcome flow inflation while still allowing to steer aggregate traffic and override BGP behavior, iSDX encompasses a virtual addressing scheme. By using an identical MAC destination address for traffic belonging to the same FEC, steering can be performed for packets with the same IP destination prefixes. Note that this approach allows for overriding BGP forwarding behavior. Ethernet source addresses are augmented when frames enter the fabric. The augmented destination addresses are learned by the participant edge routers by means of ARP. OpenFlow rules are used to transform broadcast ARP traffic into unicast traffic, which is then redirected to the participant's iSDX controller. Within the fabric, forwarding decisions are made based on these augmented MAC addresses.

The iSDX controller encapsulates the next-hop AS for each FEC in the first 10 bits of the destination MAC address, and the set of ASes which are also eligible for receiving traffic for this flow's FEC are encoded in the remaining part. Due to the encoding mechanism, only 27 of these remaining 38 bits can be used to wrap this list of ASes. One bit is used for every AS, which effectively limits the number of ASes announcing the same prefix to 27. If more than 27 bits are required, multiple flows are created and a six bit bitmask value is incremented for every distinct flow. iSDX aims to map each FEC to a distinct virtual MAC. With an increasing amount of participants and outbound flows, the limited bit space may not be able to address this information in a single MAC address. In such an event, two virtual MACs are created and hence the number of flows for this FEC will increase. When no explicit flows are set-up for a particular destination, the next-hop AS is used to determine the egress port the frame should be forwarded to. The set of authorized ASes is used in combination with the bitmask to choose a different egress AS hence, overriding default BGP route selection. This scheme is solely used for outbound TE.

**Switch fabric formations**
In its current form iSDX can be deployed in either a multi-switch or in a multi-table formation. Since flow matching is performed repeatedly for a single frame, a mechanism is needed to forward traffic between sets of flow rules. Also, distributing flows over multiple tables is more efficient than representing them in a single forwarding table[4]. This is discussed in more detail in Section 6.1. In Figure 3, flow decisions in the iSDX fabric are depicted when using a multi-table configuration. In this example, participant A has set an outbound policy which forwards all traffic with TCP destination port 80 to participant B, provided that B has advertised the destination network to the IXP route server.

The Ethernet frame incoming to the fabric is sent by A. For the sake of simplicity, network layer properties have been omitted from this example. A received a virtual MAC address $::X$ as an answer to its ARP request for this particular FEC. The input table sets the source MAC address of this frame to A's augmented address. As participant A has set at least one outbound policy, the frame is forwarded to the output table. If A would not have had any outbound flows defined the frame would directly be forwarded to the inbound table, saving unnecessary lookups in the outbound table. In the outbound table the combination of Ethernet source, destination and TCP destination are matched. Accordingly, the destination MAC address is rewritten to participant B's virtual MAC address. Next, no inbound flows are matched in its corresponding table, after which the frame is ultimately forwarded to the output table. Here, the augmented destination MAC address is replaced by B's authentic MAC.

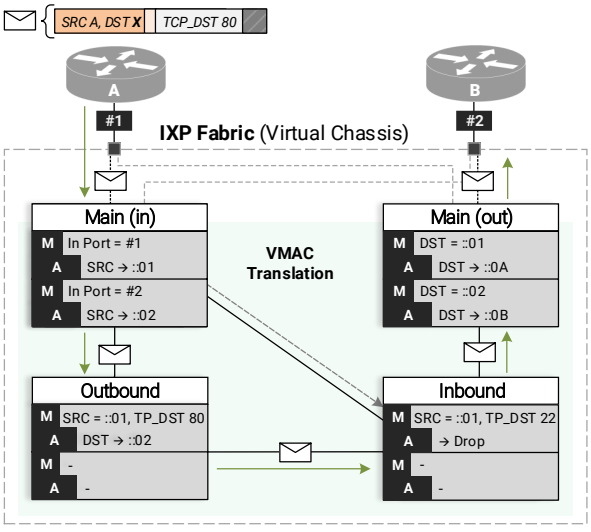


**Figure 3:** Representation of the iSDX table pipeline and its match/action behavior

**Layout & Controller Elements**
The iSDX controller software consists of multiple components as depicted in Figure 4. When deployed, participants would just like they're doing

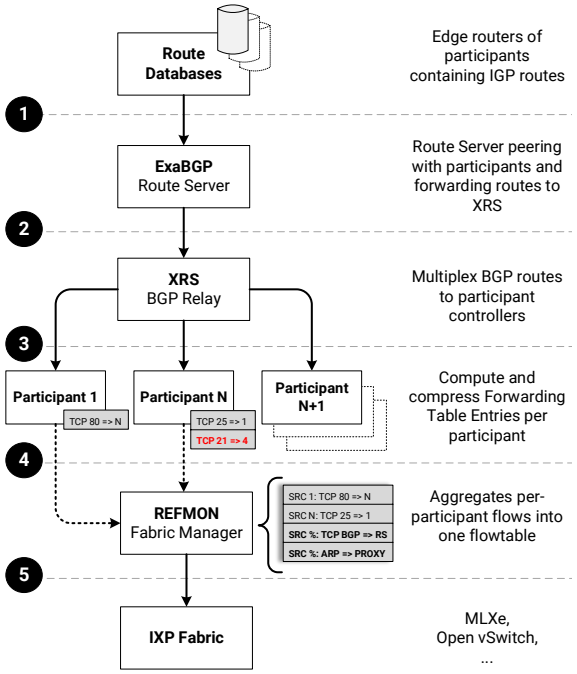


**Figure 4:** An overview of the original iSDX controller

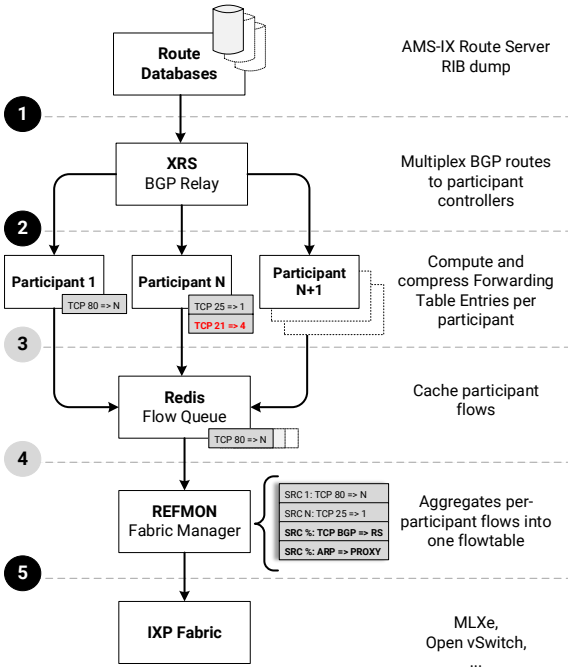


**Figure 5:** An overview of the modified iSDX controller

on a traditional IXP let their edge routers peer with the Route Server of the IXP. ExaBGP is used as Route Server as it provides granular control of distributing routes by leveraging user-defined external scripts. Upon receiving a participant's BGP update, ExaBGP forwards the update exclusively to the iSDX controller. The *XRS* is the component receiving ExaBGP's BGP updates. When receiving such an update, it's multiplexed to all participant controllers that are configured to peer with the update's originated IXP participant.

Flow computation can now be parallelized across all IXP's participant controllers. They each maintain their own RIB and set of flow rules. Based on their contents, a participant controller (pctrl) computes and forwards a set of flow table entries. As iSDX uses MAC addresses to encapsulate forwarding information, the participant controllers also alter the BGP next-hop to a virtual IP address. The modified BGP update is then returned to the XRS which in turn forwards it to the ExaBGP Route Server. This way all participants on the IXP learn each others routes. The computed flows are propagated downstream to the Fabric Manager (Refmon). Note that the connection between each participant and the *Refmon* is not persistent, thus, connection set-up and termination takes place for every received BGP update at the *Pctrl* side. The *Refmon* is based on the Ryu controller and listens for updates coming from the participant controllers. On receiving flows, it aggregates them, checks if they comply with BGP route advertisements whereafter the flows are pushed to the IXP fabric. In order to allow participants to communicate with one another, a set of default forwarding rules is installed in the fabric when the iSDX controller starts. These rules allow participants to establish BGP peering sessions and allow ARP traffic on the fabric.

# 5 Methodology

This section presents the methodology of the project and discusses several enhancements made to the iSDX controller. Additionally, the experimentation environment is discussed and several distinct scalability test scenarios are defined.

## 5.1 Experimentation design

To perform the experiments we have been provided with the raw IPv4 and IPv6 Routing Information Base (RIB) dump of AMS-IX's route servers. For the scalability experiments the IPv4 dump will be used exclusively as at the time of writing iSDX does not yet support IPv6. Even though the participant controllers *do* accept IPv6-like routes and return flows on receiving such a BGP update, they also do so when receiving arbitrary strings sent along with the BGP prefix field. As such, route calculation within the *pctrl* can be deemed unreliable for IPv6. And so are the computed flows, as they depend on the participant controller its internal RIB. Therefore, only the IPv4 RIB dumps of both route servers were used for the experiments.

The IPv4 dumps contain ±200.000 peering routes for ±150.000 unique prefixes. With regard

to ethical concerns, the data set used for the experiments described in this report contains information which can be traced back to specific participants. As such this data set will not be publicly provided. Still, reproduction of the experiments can be performed by utilizing RIB dumps from the RIPE Routing Information Service Raw Data service [34]. A full workflow for reproducing the results in this paper has been included in Appendix A.

The data set as provided by AMS-IX is not sufficient to perform the extended scalability tests as the RIB dump only contains information of 595 unique peering IPv4 participants. In order to test for scalability beyond this number, additional IPv4 peering data is required. Therefore the data set is extended with uniformly random generated participants. Every generated participant is guaranteed to have a unique AS number. Based on characteristics of the original RIB dump, each of the fictitious participants advertises five prefixes that is neither a more specific to an already existing prefix in the RIB dump, or vice versa. For the next hop, sequential addresses are distributed from the class C private IP space. However, as on average one-fifth of all participants of the route server peers have a secondary port on the fabric, the generated participants must also meet this average. Therefore, approximately 20% of the artificial participants has an additional next-hop hence, a second port on the fabric. Extending the data set allows for extrapolating beyond the current amount of peering participants, while maintaining participant characteristics. A limitation to this approach however, is that it does not account for any future variation in the number of ports per participant nor in the amount of prefixes advertised by participants.

**Hardware**
The iSDX controller software is deployed on a server provided by AMS-IX in a lab environment, containing 56GiB of Random Access Memory (RAM) and two Intel E5640 Quadcore CPU's. Furthermore a Brocade MLXe-8, MLXe-32, VDX 6740 and SLX 9850 were made available to connect to the SDN controller. All of these switches include support for the OpenFlow 1.3 specification.

## 5.2 Controller enhancements

In order to validate the flow table sizes for a set of participants and at the same time simulate traffic among participants, a disproportionate amount of computing resources would be required. To overcome the limited amount of available computing resources, a number of modifications were made to the iSDX controller implementation. These enhancements are depicted in Figure 5.

**Eliminate traffic simulation**
Since this research aims to validate the claims with regard to the flow table size, there is no need to actually simulate traffic among IXP participants. Therefore the virtualized participant routers could be left out of the simulation, saving resources. However, as there's still a need to process their BGP advertisements down the iSDX pipeline, we inject them directly into the XRS. Thus, omitting the ExaBGP route server (Figure 5, step 1). In summary, instead of launching a Quagga instance for every IXP participant, mimicking their edge router and advertising Interior Gateway Protocol (IGP) routes to the ExaBGP route server, we generate those BGP advertisements from the RIB dump, and inject them directly into the XRS.

**Queue participant flows**
As discussed in Section 4, the connection between each participant controller and the fabric manager is not persistent thus, set-up and terminated for each flow transmitted to the fabric manager. This resulted in a storm of TCP connection initiations which led to latency and stability issues when simulating over 300 participants. To enhance troubleshooting and stability, this connection was replaced by a persistent connection between each *pctrl* and an in-memory data structure store, Redis. All flows created by the participant controllers are temporarily stored in a Redis list structure, functioning as a queue. This way, outstanding flows can easily be debugged, and new custom flows can be inserted on the fly.

## 5.3 Switch fabric connection

At the time of writing, the iSDX controller[1] solely includes detailed instructions for integrating iSDX with either Open vSwitch instances or switches based on Broadcom chipsets. The original experiments as performed by Gupta et al. make use of Open vSwitches. On GitHub, additional instructions are provided for using Quanta LY2 hardware switches. These whitebox switches are built with Broadcom merchant silicon and thus require Broadcom's OpenFlow Data Plane Abstraction (OF-DPA) layer to push flow rules to the proprietary hardware tables. The Brocade MLXe platform on the other hand uses Brocade's in-house MaxScale-160 Packet Processor chips which allow for directly injecting OpenFlow rules into the TCAM tables.

Ryu addresses the switches it manages by referencing their datapath identifiers. These identifiers uniquely identify each switch in the Open-

[1]Industrial Scale Software Defined IXPs (iSDX) controller: `https://github.com/sdn-ixp/iSDX`

| Scenario | Up to IXP Participants | Flows to % of participants/prefixes | Maximum no. of policies per entity |
|---|---|---|---|
| 1 | $\leq 800$ | 10 | 4 |
| 2 | $\leq 800$ | 10/30/50 | 4/8/16 |
| 3 | $\leq 800$ | 10 | 4 |

**Table 1:** Summary of all the described scalability experiments

Flow fabric. Due to the fact that the Brocade MLXe switches are directly addressable, modifying the datapath identifier in the Ryu configuration file is sufficient to push OpenFlow rules to the desired physical switch. During the course of performing the experiments, Open vSwitches are used as a fall-back scenario whenever the MLXe platform lacks certain functionality.

## 5.4 Test cases

In order to evaluate the scalability of the iSDX controller the results of Gupta et al. will be validated by reproducing the experiment as described in [4]. Subsequently, a series of distinct test scenarios is defined in which factors such as the amount of peering participants, outbound policies and policy parameters are varied. As we extend on the original iSDX paper, and inbound policies don't leverage the iSDX encoding scheme, we only simulate outbound policies. Additionally, we are interested in the effect of specifying policies on various network layers as this may have an effect on the finite amount of TCAM available [35]. Therefore, multiple rulesets will be generated and evaluated.

With regard to the experiments, scaling up the amount of participants inherently means that the time required to perform a single measurement increases exponentially. As such, all measurements are performed for a total of five times. Due to the nature of the experiments, the results can exclusively vary between the boundaries as defined in the test, meaning that outliers are non-existent. Because there is no skew of results, all measurements are averaged over their total amount of runs. The exact workflow of performing the scalability experiments and varying the testing parameters is described in more detail in Appendix A.

**Experimentation constraints**
Due to the instability of the iSDX controller software, it was not possible produce reliable test results for more than 400 participants. As the iSDX developers have been contacted about these issues and no solution has yet been found, only the generated policies will be taken into account for our experiments with over 400 participants, instead of the compressed flows as calculated by the iSDX controller. This limitation and its impact on the results is discussed in Section 8.

**Encoding scalability**
As mentioned in Section 4.3 policy compression efficiency is constrained in an event where more than 27 distinct Autonomous Systems announce the same prefix. In order to determine whether this would cause any flow inflation for AMS-IX, the first part of this research aims to identify these prefixes. Therefore, the RIB dumps were analyzed for duplicate announcements of the same prefix by distinct ASes.

**Scenario 1: Scalability validation**
In the original iSDX paper, Gupta et al. [4] perform scalability experiments in which each participant has between one and four outbound policies for 10% of the total participants. The amount of participants is scaled up to 500 with increments of 100 participants. As previously discussed in Section 4.1, in order for iSDX to properly scale for the AMS-IX, the amount of participants should at least scale to around 800. Therefore, this first test scenario will reproduce the initial experiment and simultaneously extend on the amount of participants following the growth pattern as observed in Figure 2.

**Scenario 2: Expansion of policies**
The first test scenario solely varies the number of peering participants. Due to the size of the AMS-IX and their participants, the settings of this scalability test might be too small-scale. In order to determine the scalability limits of iSDX, the number of policies each participant sets are expanded. As shown in Table 1, each participant sets up outbound policies to up to 50% of the other IXP participants. For each selected participant, up to 16 outbound TCP policies are created. Although these could just as well be UDP policies, we reproduce the scenario in the original iSDX paper and create TCP-only policies.

**Scenario 3: Granular policies**
The previous scenarios defined flow rules for aggregate traffic in which all traffic for a given TCP destination port is forwarded to *one* participant. Situations are imaginable in which a given participant would for example forward HTTP traffic (port 80) for a specific IP prefix to a certain participant. Hereby overriding default BGP behavior for

a specific prefix and TCP destination port, while leaving forwarding behavior for other traffic untouched. Figure 6 visualizes this scenario in more detail, in which scenario one and two are drawn on the right side, while traffic engineering based on network layer characteristics is depicted on the left side. The latter is tested in this scenario. Each participant creates up to four outbound policies for each prefix in 10% of the IXP prefixes.

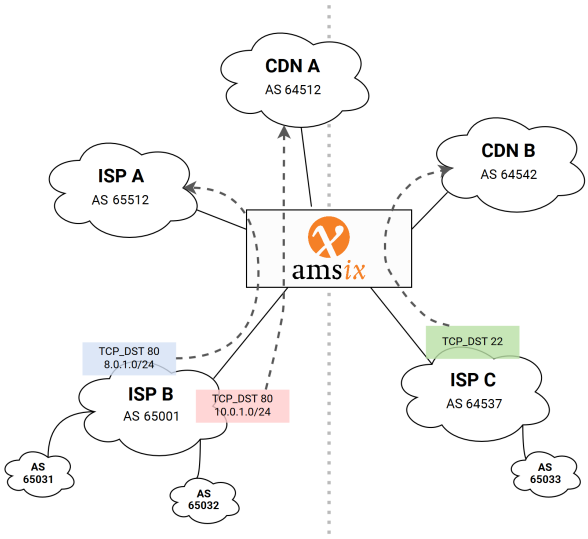


**Figure 6:** Prefix-based peering scenario (left) and default iSDX behaviour (right)

## 5.5 Impact analysis

Introducing Software Defined Networking at an Internet exchange point to enhance traffic engineering capabilities is a fairly novel concept. Therefore, prior to performing the scalability experiments, methods for deploying and migrating the IXP to a Software Defined IXP will be discussed.

# 6 Deployment impact

As discussed in Section 4.1, the services of AMS-IX are currently built on top of the Brocade MLXe switch platform. With regard to a possible implementation of iSDX within the IXP, these switches have a series of drawbacks which limit the feasibility of an IXP-wide deployment. Prior to presenting the results of the scalability experiments, this section discusses limitations with regard to the current switch platform and goes into more detail on possible workarounds. Subsequently, it delves into the impact of deploying iSDX at the AMS-IX. For a full discussion of the experimentation results the user is referred to Sections 7 and 8.

## 6.1 Brocade MLXe platform

Although iSDX introduces new functionality at the IXP, it does not require support for exotic OpenFlow match fields and actions. This is important since a significant amount of OpenFlow instructions described in the specification are optional for hardware vendors to implement. iSDX mainly requires support for matching and modifying relatively simple layer 1 (logical) ports, layer 2 and 3 source and destination addresses and layer 4 source and destination ports. Additionally, as described in Section 4.3, matching on the layer 2 destination with arbitrary bit masks is required for traffic engineering. The core of AMS-IX's network is built around Brocade Networks MLXe-32 switches. The chassis used for the experiments as described in Section 5.1 is deployed with the NetIron R05.7.00 firmware. According to the technical documentation as provided by Brocade, this firmware version is fully capable of supporting all the OpenFlow match and action fields required to implement iSDX.

**Flow installation**

Initially, every flow table in the switch is empty and does not contain any flow rules. Therefore the controller will add a table-miss entry which defines a series of actions to perform when not a single flow entry in the table matches the traffic. The test environment of Gupta et al. makes use of a series of Open vSwitches (OVS) to perform their experiments. This is significant due to the fact that at the time of writing, OVS does not fully support the table-miss flow entry. To circumvent this limitation, Ryu, part of the *Refmon*, injects a normal flow entry with a low priority level and broadly defined matching criteria, causing it to function as a table-miss entry.

However, when attempting to create a session with the Brocade MLXe this non-default behavior causes problems. After a successful OpenFlow handshake the controller attempts to add the table-miss entry by means of a `OFPT_FLOW_MOD` message. This causes the Brocade MLXe-32 switch to respond with an `OFPT_ERROR`, indicating that the received flow modification is a malformed packet. As this is unintended behavior, a support case has been opened with Brocade to resolve the non-standard behavior. A workaround is to upgrade the firmware of the switch as the issues perceived with regard to OpenFlow in the NetIron R05.7.00 firmware are largely rectified in the later R05.9.00 version. However, upgrading the MLXe-32 switches would require upgrading the current Management modules in the chassis from version MR to MR2 which may be a costly decision, considering the age of this platform. Subsequently, preliminary tests were performed

on an MLXe-8 switch in the AMS-IX lab. This switch was equipped with the correct firmware and allowed for injecting the required flow rules in TCAM. Listing 1 presents an example of a flow rule as injected by iSDX.

```
curl -X POST -d '{
    "dpid": 14721744138777133056,
    "cookie": 1,
    "table_id": 0,
    "priority": 10,
    "flags": 0,
    "match":{
        "dl_dst":"00:00:00:00:00:04/00:00:00
            ↪ :00:ff:ff",
        "dl_type":2048,
        "tp_dst": 4322, "ip_proto": 6
    },
    "actions":[{
        "type": "SET_FIELD",
        "field": "eth_dst",
        "value": "00:00:00:01:00:04"
    }]
}' http://localhost:8080/stats/flowentry/
    ↪ add
```

**Listing 1:** Injecting arbitrary flow rules in the MLXe platform with Ryu

**Table allocation**

The MLXe switch platform used by the AMS-IX was originally designed to be compliant with the OpenFlow 1.0 specification. At that point in time, multi-table pipeline processing was not yet present in the specification. This means that the MLXe switches only support a single OpenFlow flow table in TCAM. This is a major limiting factor in implementing iSDX at the IXP because, as discussed in Section 4.3, the lack of additional flow tables limits the IXP to performing iSDX in the highly inefficient multi-switch setup. Theoretically it would be possible to deploy iSDX in a multi-switch formation in which multiple MLXe chassis are deployed to distribute the main, inbound and outbound table over. However, this would require a significant investment. Additionally, due to the way the flow rules are set up by iSDX, all participants would have to be coupled to the main MLXe switch. Although the switches have been made compliant with the OpenFlow 1.3 specification, the use of multiple flow tables is optional.

To alleviate this problem, a series of abstraction layers have been developed which allow earlier OpenFlow switches to perform multi-table processing with legacy hardware. Pan et al. [36] describe such a software abstraction in the form of 'FlowAdapter'. This technique is placed between the SDN controller and the data plane of the switch and translates incoming OpenFlow instructions from the controller to a format that is conform the fixed hardware configuration of the underlying switch. Inherently, the major drawback of this proposed solution is that FlowAdapter has to be implemented in the firmware of the switch. As such, the vendor of the switch platform has to incorporate the software layer in its updates. Due to the modular composition of an OpenFlow system, this problem could also be solved on a higher layer. Monsanto et al. [37] present Pyretic: an abstract programming language which allows for writing modular SDN applications. These abstract languages are theoretically capable of combining the in- and outbound policy definitions of all participants into a single forwarding table. However, due to the fact that there is only one flow table available, the resulting set of flow rules would be a Cartesian product of all the defined policies. Naturally, this process is disproportionally costly with regard to the available TCAM. Conclusively, the current AMS-IX switch platform is severely limited due to the lack of multiple flow tables. In the future the AMS-IX will utilize the new Brocade SLX platform which *does* have support for multiple flow tables. Due to the perceived limitations in the MLXe platform, the scalability experiments as described in Section 5 will be performed on Open vSwitches.

## 6.2 Migrating to iSDX

iSDX was originally designed to be deployed on top of a virtual chassis switch fabric. In such a setup the underlying switch fabric functions as a single logical entity. Brocade implements this functionality under the name 'VCS' and most major network vendors offer similar technologies. Functioning as such is beneficial to iSDX as each switch is aware of all other chassis members output ports due to the inter-switch communication. This allows the iSDX controller to define OpenFlow rules for *all* known output ports in the switch fabric. However, for its operational model, AMS-IX heavily relies on the capability of making a distinction between participant's traffic on the layer 2 network. Currently, MPLS and VPLS are being deployed in order to aggregate traffic with differing VLAN tags in a single broadcast domain. This functionality would be lost when a VCS is configured to simulate a layer 2 switch. An additional disadvantage is that virtual chassis techniques are commonly vendor specific. Building an infrastructure based on VCS in this case would inherently mean a vendor lock-in. With regard to the future, this is obviously an unfavorable scenario.

Therefore the current iteration of the AMS-IX platform has been deployed as a multi-hop[2] switch fabric based on MPLS and VPLS. In a multi-hop setup the switches in the underlying

[2]Multi-hop in this scenario should not be confused with *multi-switch* as described in Section 4.3. Also refer to Section 4.1.

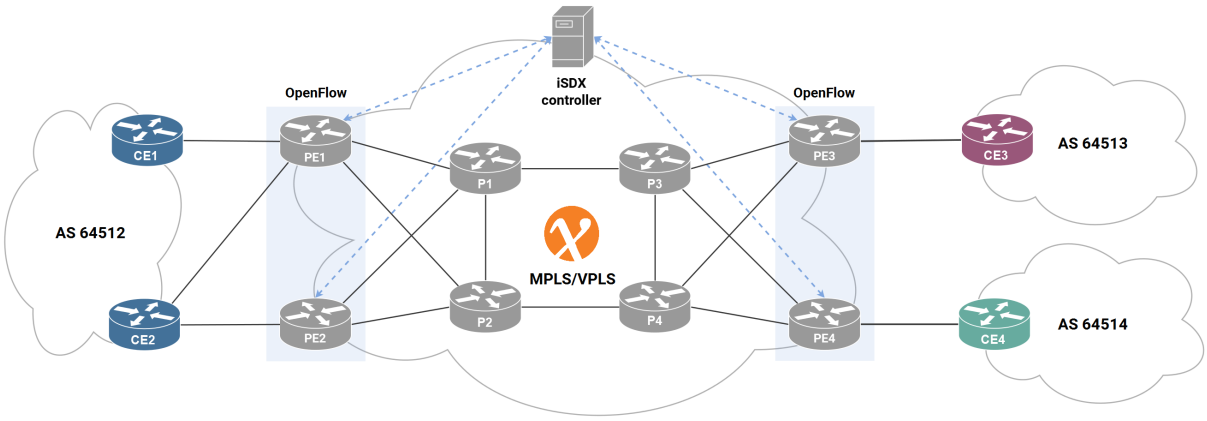


**Figure 7:** Deploying iSDX at the AMS-IX Provider Edge switches

platform are disjoint and each form a single entity by themselves. This inherently means that the iSDX controller *can not* directly define actions for output ports on other switches, thus complicating a deployment scenario for iSDX at the IXP. Given these constraints, and the limitations of the current MLXe switches, the following section proposes a model which allows for integrating iSDX with future iterations of the AMS-IX infrastructure.

**Proposed deployment model**

Although the current MLXe switch platform is dated, deploying iSDX at AMS-IX in the future would still be feasible. This is due to the fact that the future Brocade SLX platform is being released as OpenFlow-hybrid switches, meaning that they support both an OpenFlow pipeline, as well as a legacy forwarding pipeline. Based on this capability, we envision a model in which the edges of the current AMS-IX platform, the Provider Edge (PE) switches, are controlled by the iSDX controller whereas the core, the Provider (P) switches maintain normal MPLS/VPLS forwarding behavior. This model is also depicted in Figure 7.

For example, consider a scenario in which AS 64514 wants to send traffic to AS 65412. In this scenario Customer Edge router 1 (CE1) is the default next hop for AS65412. In order to send traffic over the IXP, CE4 performs an ARP request for the virtual next hop address of AS 65412 and receives a virtual MAC address in return. Subsequently CE4 sends its traffic destined for CE1 towards PE4 with a layer 2 destination of that VMAC address. On PE4, traffic comes into an interface configured as OpenFlow-hybrid, meaning that it supports both OpenFlow traffic forwarding as well as normal traffic forwarding. By default, all traffic that is not shielded off by a specific series of Private VLANs is handled by the OpenFlow pipeline. Thus, when the traffic comes into PE4, it would be directed through the main (in and out), outbound and inbound tables as defined by iSDX. When traffic leaves the main outbound table, it will have the real physical MAC address of CE1 in its Ethernet destination header.

Because the P and PE switches are not deployed as a virtual chassis, this would normally pose problems since PE4 has no knowledge of CE1's real MAC address. However, because VPLS is used to simulate a layer 2 Ethernet broadcast domain, all switches learn all MAC addresses of the connected peers over the Virtual Circuits (VC). At this point the switch still has to forward the traffic over the MPLS/VPLS backbone. This can be achieved by reconfiguring the output action in iSDX to send traffic to the NORMAL reserved OpenFlow port. This causes the traffic to be sent over the legacy forwarding pipeline, thus resulting in PE4 to perform an MPLS label lookup for the physical MAC address and subsequently switching the traffic over the backbone. Although this is unconventional behaviour, it is defined as an optional feature in the specification for OpenFlow-hybrid switches and is supported by both Brocade switch platforms:

> *An OpenFlow-hybrid switch may also allow a packet to go from the OpenFlow pipeline to the normal pipeline through the NORMAL and FLOOD reserved ports* [38]

On the inside of the MPLS/VPLS backbone the interfaces on the switches can be configured as normal ports. Because the iSDX policy is applied on the edge, zero knowledge of OpenFlow is required in the P switches. When traffic arrives at the opposing end's PE -*PE1 in this scenario*- the MPLS label will be popped and traffic is forwarded via the hybrid port's normal pipeline out towards CE1.

**Advantages & disadvantages**

The primary advantage of deploying the proposed model is that with this setup the current MPLS/VPLS infrastructure can be retained whilst only the

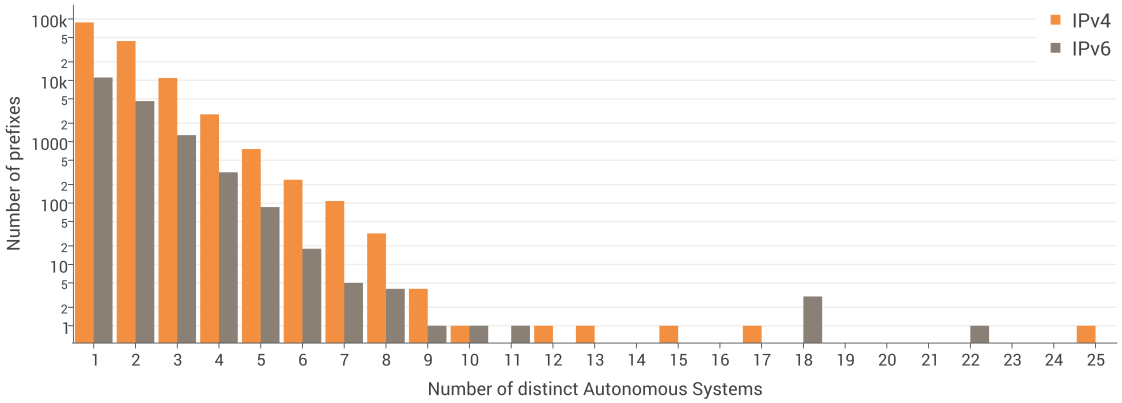


**Figure 8:** Frequency of prefixes announced by unique Autonomous Systems on a logarithmic scale

edge of the network has to be modified to support OpenFlow. All policy decisions are made on the edge and the backbone can remain a 'simple' forwarding element. However, as of right now it is not possible to separate the flow tables for the PE switches based on the connected participants. As such, performing OpenFlow rule matching on the edge of the network inherently means that *every* PE switch needs to be able to house the complete iSDX flow table. Naturally, as the amount of policies per participant grows this becomes increasingly more difficult. The implications of scaling up the amount of policies on flow table sizes is evaluated in more detail in Section 7. In the current AMS-IX platform configuration, the edge of the network is formed by the relatively less capable MLXe-8 and MLXe-16 switches. Due to the large TCAM requirements of iSDX, incorporating this concept in future platform iterations would most likely require the PE switches to be upgraded. Although not yet possible, partitioning the flowtable such that it would only contain the flows relevant to the connected IXP participants of the switch in question could greatly improve the deployment feasibility of iSDX.

An adverse effect of the scenario as shown in Figure 7 is that combining both forwarding pipelines results in a more costly lookup: after the OpenFlow matching, an MPLS label lookup has to take place. Arguably, a more elegant approach for integrating iSDX would be to let the controller determine the correct MPLS label and include the tagging of traffic as an action in the OpenFlow pipeline. However, this would require the controller to become aware of the Label Forwarding Information Bases (LFIB) of the PE switches in order to apply the correct label. In such a scenario, consistency of the LFIBs and the speed of the controller become significant factors in traffic forwarding and network resilience. Another option would be to fully implement the current MPLS/VPLS behaviour in OpenFlow. However, this would require all switches in the fabric to support OpenFlow. Additionally, this would most likely cause a significant inflation in the total number of flow rules in the fabric. Either way, both of these latter approaches would require a significant modification of the iSDX controller whereas the first model only requires modification of the output action.

# 7 Scalability evaluation

This section presents the results of the test cases as described in Section 5. Additionally, the AMS-IX Route Server RIB is analyzed in order to determine whether all prefixes could be supported by iSDX's hierarchical encoding.

## 7.1 Encoding scalability

According to the RIB dump of the AMS-IX Route Servers, over 99% of all prefixes are advertised by five or less Autonomous Systems. Of all prefixes advertised by at least six ASes, only two percent is advertised by more than 10 participants. Although absolute numbers differ, these trends hold true for both IPv4 and IPv6. As presented in Figure 8, two prefixes exist for which more than 20 participants announce a route.

## 7.2 Policy scalability

The results for every tested scenario as described in the methodology are discussed in this section.

**Case 1: Scalability validation**
Figure 9 shows that up to 700 IXP participants, the number of flows follow a slight exponential growth. Whenever the number of IXP participants doubles, the number of flows quadruples. In that sense, the results for seven and eight-hundred participants slightly deviate from this trend.

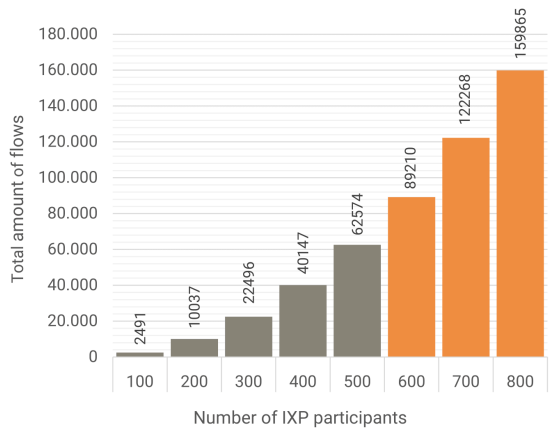


**Figure 9:** Scalability validation (Case 1)

**Case 2: Expansion of policies**
When doubling the number of outbound policies per participant from four to eight, the number of flows ending up in the fabric doesn't. Doubling from eight to 16 policies exhibits a similar trend. Figure 10 also shows a linear growth when increasing the percentage of participants each IXP member creates outbound policies for.

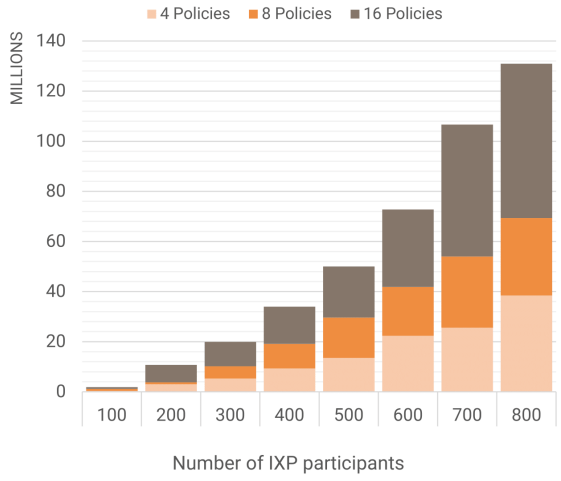


**Figure 11:** Granular policies (Case 3)

**Case 3: Granular policies**
For prefix instead of participant based policy creation, the fabric will need to support up to tens of millions of flows. Like in scenario one, the amount of flows doubles on duplicating the number of IXP participants. Similar to the scalability validation, the simulations concerning seven and eight-hundred participants deviate from this trend. Moreover, the trend witnessed in case two is also visible in Figure 11, in which increasing the number of policies from four to eight doesn't result in a double amount of flows.

# 8 Discussion

The following sections analyze the experimentation results and elaborates on whether these findings affect the scalability of iSDX.

## 8.1 Encoding scalability

As there are no prefixes announced by more than 27 distinct Autonomous Systems, reachability information can be embedded in the MAC destination field using iSDX encoding. Although one prefix exists for which the number of next-hop candidates nears the threshold, this does not constitute an immediate scalability concern for deploying iSDX. Hence, compression would be suboptimal for this prefix exclusively.

## 8.2 Policy scalability

The results of the scalability validation shows that we were able to reproduce thus validate the original iSDX scalability claims. The divergence observed in the experimentation results when testing over 700 participants can be attributed to rounding the average number of policies per participants upwards. For this reason the deviation is only observed for over 700 participants, which is when the the fictitious participants come into play. Furthermore, we noticed the amount of flows did not double when the maximum tolerated amount of outbound policies did. This is caused by the way random numbers in a certain range are generated, and may require background information to fully comprehend the cause of this problem. When an IXP member has selected an $X$ percent of IXP participants it wants to create outbound policies for, it randomly creates one up to $Y$ outbound policies for each of the participants in this set. With $Y$ being either four, eight or sixteen. When performing this random selection a number of times, the average number of created policies with $Y$ being four, is 2.5. Unlike one might expect, when doubling $Y$, the average does not do so. Instead, when $Y$ doubles to eight, the average number of created policies will be 4.5, which explains the observed results. The growth observed when creating prefix instead of participant based policies has two causes. First is the number of prefixes being about two orders of magnitude larger than the number of IXP participants. Second is the use of iSDX's superset encoding scheme, which was designed for compressing reachability information for an aggregate of prefixes, instead of a single prefix. Setting up prefix based policies basically defeats the purpose of this encoding scheme. The algorithm used in the initial SDX proposal would be more suited to compress this type of policies,

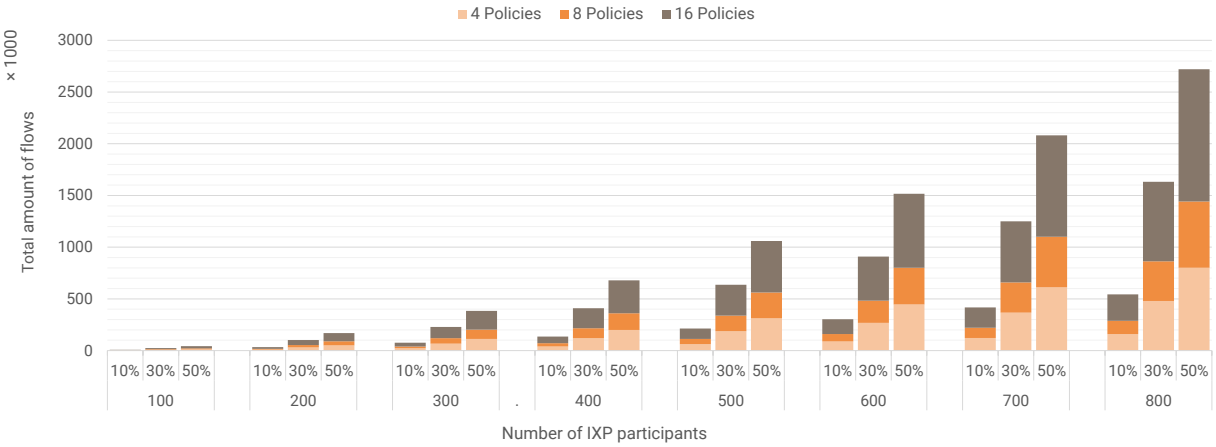


**Figure 10:** Expansion of policies (Case 2)

as it was designed to merge policies with identical Network/Transport layer matching criteria.

**Limitations**
As described in the Methodology, the iSDX controller cannot perform simulations for more than 400 IXP participants. Therefore it is impossible to deduce whether policy inflation would take place at an IXP with over 400 members. Thus, results for over 400 participants represent an optimal situation in which policy compression could be performed at all times. With regard to future growth, the assumption was made that with 800 participants the average number of announced prefixes per IXP member will remain as it currently is.

**Advancement**
Performing iSDX policy compression on an IXP with up to 400 participants works without inflation. It's however questionable whether iSDX could be considered industrial scale. Although the AMS-IX has almost 600 participants peering with its route server, larger Internet exchange points exist. Even if the iSDX policy compression would be able to cope with the world's largest IXPs, current hardware is not capable to store such an amount of OpenFlow rules in memory. For example, the MLXe-32 can store 128.000 flows in its TCAM [39]. Additionaly, its successor, the SLX platform based on the Jericho chip, is expected to be able to handle at least double. However, deploying iSDX at this scale is still precarious. Unless an IXP constrains their members in the amount of policies they are allowed to apply, overflowing the TCAM is a serious risk. This can result in traffic being either forwarded to the controller or flooded over all ports. On the other hand, limiting customers in the way they setup their policies could bring harm to the neutral status Internet exchange points have acquired. Summarizing, scalability concerns when deploying iSDX are mainly caused by OpenFlow not yet being ready to be deployed at a large scale.

**Closing figments**
As mentioned previously, the controller is not able to process flows for more than ±400 participants. We found that this might be caused by a race condition in the implementation of the participant controller. The socket communicating to the XRS is shared among multiple threads without implementing a mutex. The iSDX developers have been contacted regarding this issue, yet no solution has been found. Also, when performing the experimentations we found a number of other limitations. First is a queue in the Refmon that continually filled up until it blocked, preventing us from simulating more than ±100 participants. Moreover, the current design of iSDX performs all traffic engineering based on the egress AS. If an IXP member has multiple ports connected to the fabric, other participants cannot differentiate between them. When dealing with a variety of link speeds, one might want to create an outbound policy to a multi-port AS and steer traffic over the faster link. We've been in contact with Prof. M. Canini from the University of Louvain, who is working on project ENDEAVOUR [40]. As part of this project, the Umbrella controller is being developed, which aims to improve on iSDX and also addresses the multi egress port problem.

# 9 Conclusion

In this paper we have evaluated the feasibility of converting the current AMS-IX infrastructure to a Software Defined Internet Exchange Point. More specifically, we assessed the effect of defining an incrementally increasing amount of interdomain traffic policies on the number of generated OpenFlow rules. Additionally we propose a model which allows for iSDX to be transparently incorporated in future iterations of the AMS-IX platform.

Due to the position of an IXP in the Internet, iSDX is a unique concept with few comparable technologies. The BGP FlowSpec extension provides similar features but lacks the transparency towards participants that iSDX offers. However, the scalability of iSDX is limited. Our experiments indicate that for 800 unique IXP participants, in a scenario where each participant defines between one and four outbound policies for 10% of the total participants, approximately 160.000 unique flow entries are required. The situation worsens when prefix based policies. At that point tens of millions of unique flow entries are required to embody the full set of traffic policies. Additionally, due to current limitations with regard to multi-hop environments, the state computed by iSDX needs to be copied to all edge switches so as to ensure consistent processing. As such, practical scalability of the concept is still heavily constrained by current hardware capabilities. In the future iSDX can be integrated by performing iSDX forwarding decisions on the edge of the network. By leveraging both the OpenFlow and normal traffic pipeline in OpenFlow-hybrid switches, iSDX can be implemented whilst retaining the current MPLS/V-PLS infrastructure. However, the current Brocade MLXe switch platform employed by the AMS-IX has been found to be incapable of integrating with iSDX due to the lack of multiple flow tables.

Considering the time frame and limitations imposed by the scope of this project, we conclude on the notion that iSDX in its current state does not scale to the size of AMS-IX without policing the amount of allowed policy definitions per participant. However, limiting this number would intrinsically impact the neutral position of AMS-IX as an Internet exchange.

# Future work

In this report we propose a model for deploying iSDX in a multi-hop environment over an MPLS backbone. However, native support for multi-hop environments with proper division of the flow tables would allow the concept to take on a much larger scale. Preliminary efforts in this area are being made in the ENDEAVOUR project [40], which introduces a subcontroller named 'Umbrella'. Although these efforts seem promising, thorough research into the features and limitations of this approach is required. In the future it may be interesting to evaluate the impact of defining interdomain traffic policies in P4. P4 is a relatively new high-level programming language [41] that shares many of its characteristics with OpenFlow but supposedly supports even more fine-grained matching criteria. Due to the fact that the language is in its early development stages, P4 has not been listed as a viable alternative for iSDX in this report. Yet, this new packet-processing method might be able to overcome the scalability limitations perceived in OpenFlow.

At the time of writing, the provided iSDX controller is not yet suited for a production deployment. Moving forward we suggest a thorough review of the current code base in order to eliminate current stability issues. Additionally, the inclusion of mission-critical features like IPv6 support is essential for the adoption of iSDX as a whole. Lastly, this paper primarily assessed the effect of policy definitions on the amount of flows in the flowtable. The computation time required to perform policy compression and the effect of frequent BGP updates on the flows in memory require further investigation to verify whether these aspects actually scale to the size of the largest IXPs.

# Acknowledgements

We would like to thank Ariën Vijn and Joris Claassen for guiding us throughout the course of the project and for proofreading our initial proposal and the final report. Their rapid response times regarding technical questions and resource requests allowed for a successful research project. Furthermore we would like to thank Eric Nghia Nguyen Duy for providing us with Routing Information Base dumps of AMS-IX's route servers and Yannick San-A-Jong for handing over additional information regarding peering statistics of participants on the platform. Lastly we thank all AMS-IX employees for their continuous hospitality and genuine interest in our research project.

# References

[1] B. Schlinker, K. Zarifis, I. Cunha, *et al.*, "Peering: An as for us", in *Proceedings of the 13th ACM Workshop on Hot Topics in Networks*, ACM, 2014, p. 18 (Cited on pages 2, 3).

[2] P. Marques, R. Raszuk, D. McPherson, *et al.*, "Dissemination of flow specification rules", *Arbor*, 2009 (Cited on pages 2, 3, 5).

[3] N. Feamster, J. Rexford, S. Shenker, *et al.*, "Sdx: A software defined internet exchange", *Open Networking Summit*, 2013 (Cited on page 2).

[4] A. Gupta, R. MacDavid, R. Birkner, *et al.*, "An industrial-scale software defined internet exchange point", in *13th USENIX Symposium on Networked Systems Design and Implementation (NSDI 16)*, 2016, pp. 1–14 (Cited on pages 2–4, 6, 9).

[5] N. Feamster, H. Balakrishnan, and J. Rexford, "Some foundational problems in interdomain routing", in *Proceedings of Third Workshop on Hot Topics in Networks (HotNets-III)*, Citeseer, 2004, pp. 41–46 (Cited on page 3).

[6] A. A. Stewart and M. F. Antoszkiewicz, "Bgp route analysis and management systems", *ArXiv preprint arXiv:0908.0175*, 2009 (Cited on page 3).

[7] V. Kotronis, X. Dimitropoulos, and B. Ager, "Outsourcing the routing control logic: Better internet routing based on sdn principles", in *Proceedings of the 11th ACM Workshop on Hot Topics in Networks*, ACM, 2012, pp. 55–60 (Cited on page 3).

[8] N. Feamster, H. Balakrishnan, J. Rexford, *et al.*, "The case for separating routing from routers", in *Proceedings of the ACM SIGCOMM workshop on Future directions in network architecture*, ACM, 2004, pp. 5–12 (Cited on page 3).

[9] A. Basu, C.-H. L. Ong, A. Rasala, *et al.*, "Route oscillations in i-bgp with route reflection", in *ACM SIGCOMM Computer Communication Review*, ACM, vol. 32, 2002, pp. 235–247 (Cited on page 3).

[10] T. G. Griffin, F. B. Shepherd, and G. Wilfong, "The stable paths problem and interdomain routing", *IEEE/ACM Transactions on Networking (ToN)*, vol. 10, no. 2, pp. 232–243, 2002 (Cited on page 3).

[11] R. Teixeira, A. Shaikh, T. Griffin, *et al.*, "Dynamics of hot-potato routing in ip networks", *ACM SIGMETRICS Performance Evaluation Review*, vol. 32, no. 1, pp. 307–319, 2004 (Cited on page 3).

[12] N. Feamster and H. Balakrishnan, "Verifying the correctness of wide-area internet routing", 2004 (Cited on page 3).

[13] N. Feamster, J. Winick, and J. Rexford, "A model of bgp routing for network engineering", in *ACM SIGMETRICS Performance Evaluation Review*, ACM, vol. 32, 2004, pp. 331–342 (Cited on page 3).

[14] G. Goodell, W. Aiello, T. Griffin, *et al.*, "Working around bgp: An incremental approach to improving security and accuracy in interdomain routing.", in *NDSS*, 2003 (Cited on page 3).

[15] T. G. Griffin and G. Wilfong, "A safe path vector protocol", in *INFOCOM 2000. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, IEEE, vol. 2, 2000, pp. 490–499 (Cited on page 3).

[16] R. Mahajan, D. Wetherall, and T. Anderson, "Interdomain routing with negotiation", Tech. Rep. CSE-04-06-02, University of Washington, Tech. Rep., 2004 (Cited on page 3).

[17] J. Mambretti, J. Chen, and F. Yeh, "Software-defined network exchanges (sdxs): Architecture, services, capabilities, and foundation technologies", in *Teletraffic Congress (ITC), 2014 26th International*, IEEE, 2014, pp. 1–6 (Cited on page 3).

[18] C. E. Rothenberg, R. Chua, J. Bailey, *et al.*, "When open source meets network control planes", *Computer*, vol. 47, no. 11, pp. 46–54, 2014 (Cited on page 3).

[19] A. Gupta, N. Feamster, and L. Vanbever, "Authorizing Network Control at Software Defined Internet Exchange Points", 2016 (Cited on page 3).

[20] J. Chung, J. Cox, J. Ibarra, *et al.*, "Atlanticwave-sdx: An international sdx to support science data applications", 2015 (Cited on page 3).

[21] J. P. Stringer, Q. Fu, C. Lorier, *et al.*, "Cardigan: Deploying a distributed routing fabric", in *Proceedings of the second ACM SIGCOMM workshop on Hot topics in software defined networking*, ACM, 2013, pp. 169–170 (Cited on page 3).

[22] R. Govindan, C. Alaettinoğlu, K. Varadhan, *et al.*, "Route servers for inter-domain routing", *Computer Networks and ISDN systems*, vol. 30, no. 12, pp. 1157–1174, 1998 (Cited on page 3).

[23] P. Richter, G. Smaragdakis, A. Feldmann, *et al.*, "Peering at peerings: On the role of ixp route servers", in *Proceedings of the 2014 Conference on Internet Measurement Conference*, ACM, 2014, pp. 31–44 (Cited on pages 3, 4).

[24] A. Gupta, L. Vanbever, M. Shahbaz, *et al.*, "Sdx: A software defined internet exchange", *ACM SIGCOMM Computer Communication Review*, vol. 44, no. 4, pp. 551–562, 2015 (Cited on page 3).

[25] R. Lapeyrade, M. Bruyère, and P. Owezarski, "Openflow-based Migration and Management of the TouIX IXP", in *IFIP/IEEE International Workshop on Management of the Future Internet (MANFI'2016)*, 2016 (Cited on page 3).

[26] J. Bailey, D. Pemberton, A. Linton, *et al.*, "Enforcing rpki-based routing policy on the data plane at an Internet Exchange", in *Proceedings of the third workshop on Hot topics in software defined networking*, ACM, 2014, pp. 211–212 (Cited on page 3).

[27] S. H. Yeganeh, A. Tootoonchian, and Y. Ganjali, "On scalability of software-defined networking", *Communications magazine, IEEE*, vol. 51, no. 2, pp. 136–141, 2013 (Cited on page 3).

[28] W. Braun and M. Menth, "Wildcard compression of inter-domain routing tables for openflow-based software-defined networking", in *Software Defined Networks (EWSDN), 2014 Third European Workshop on*, IEEE, 2014, pp. 25–30 (Cited on page 3).

[29] M. Canini, P. Kuznetsov, D. Levin, *et al.*, "A distributed and robust sdn control plane for transactional network updates", in *Computer Communications (INFOCOM), 2015 IEEE Conference on*, IEEE, 2015, pp. 190–198 (Cited on page 3).

[30] T. Bu, L. Gao, and D. Towsley, "On characterizing bgp routing table growth", *Computer Networks*, vol. 45, no. 1, pp. 45–54, 2004 (Cited on page 3).

[31] J. C. Cardona Restrepo and R. Stanojevic, "A history of an internet exchange point", *ACM SIGCOMM Computer Communication Review*, vol. 42, no. 2, pp. 58–64, 2012 (Cited on page 4).

[32] L. Serodio, *Traffic diversion techniques for ddos mitigation using bgp flowspec*, 2013 (Cited on page 5).

[33] Cisco. (2016). Cisco asr 9000 series aggregation services router routing configuration guide, [Online]. Available: `http://www.cisco.com/c/en/us/td/docs/routers/asr9000/software/asr9k_r5-2/routing/configuration/guide/b_routing_cg52xasr9k/b_routing_cg52xasr9k_chapter_011.html` (visited on 07/08/2016) (Cited on page 5).

[34] RIPE NCC, *Ripe routing information service raw data*, `https://www.ripe.net/analyse/internet-measurements/routing-information-service-ris/ris-raw-data`, 2016 (Cited on page 8).

[35] B. Owens, *Openflow switching performance: Not all tcam is created equal - packet pushers -*, `https://packetpushers.net/openflow-switching-performance-not-all-tcam-is-created-equal/`, 2013. (visited on 06/01/2016) (Cited on page 9).

[36] H. Pan, H. Guan, J. Liu, *et al.*, "The flowadapter: Enable flexible multi-table processing on legacy hardware", in *Proceedings of the second ACM SIGCOMM workshop on Hot topics in software defined networking*, ACM, 2013, pp. 85–90 (Cited on page 11).

[37] C. Monsanto, J. Reich, N. Foster, *et al.*, "Composing software defined networks", in *Presented as part of the 10th USENIX Symposium on Networked Systems Design and Implementation (NSDI 13)*, 2013, pp. 1–13 (Cited on page 11).

[38] Open Networking Foundation, *OpenFlow Switch Specification Version 1.3.0*, ser. ONF TS-006. 2012, pp. 10–11. (visited on 07/05/2016) (Cited on page 12).

[39] Brocade, *Brocade mlx series routers - data sheet*, `https://www.brocade.com/content/dam/common/documents/content-types/datasheet/brocade-mlx-series-ds.pdf`, 2013. (visited on 07/08/2016) (Cited on page 15).

[40] M. Canini, S. Uhlig, M. Gusat, *et al.*, *Endeavour: Towards a exible software-de ned network ecosystem*, `https://github.com/h2020-endeavour/endeavour/`. (visited on 07/07/2016) (Cited on pages 15, 16).

[41] P. Bosshart, D. Daly, G. Gibb, *et al.*, "P4: Programming protocol-independent packet processors", *ACM SIGCOMM Computer Communication Review*, vol. 44, no. 3, pp. 87–95, 2014 (Cited on page 16).

# A Experimentation workflow

**Source Code:** `https://github.com/jeroen92/iSDX`
For each experiment the maximum amount policies and the percentage of participants to set up a flow to can be set using two shell variables. This experiment creates random flows in the fabric according to the specified settings for 100 to 800 participants, each round incrementing the amount of participants with 100. In the end, the amount of generated flows for each round are collected and outputted in CSV format to the console. The experiment was ran for each combination of mesh (10, 30, 50) and maximum policies (4, 8, 16).

```
maxpolicies=4
mesh=10
for i in $(seq 1 8); do ./routeToJson.py "$i"00 /home/vagrant/iSDX /home/vagrant "$mesh"
    ↪ "$maxpolicies" >> ./s2-"$mesh"-"$maxpolicies"; done; sudo rm /home/vagrant/iSDX/
    ↪ examples/test-mtsim/policies/p*; cat ./s2-"$mesh"-"$maxpolicies" | grep -oP '(?<=L
    ↪ : )[\d]+' | awk 'BEGIN{nrs=""}{ nrs=nrs$0",}END{ print nrs}'; cat ./s2-"$mesh"-"
    ↪ $maxpolicies" | grep -oP '(?<=M: )[\d]+' | awk 'BEGIN{nrs=""}{ nrs=nrs$0",}END{
    ↪ print nrs}'
```

**Listing 2:** Generating flow rules for the given configuration and extracting the amount of total flows

The following code was used to generate the same set of flows as the above snippet, but also allowed to connect to the OpenvSwitch in order to query the flows that were pushed into the fabric.

```
python /home/vagrant/iSDX/scripts/routeToJson.py 100 /home/vagrant/iSDX /home/vagrant 10
    ↪ 4
```

**Listing 3:** Generating flow rules for the given configuration and pushing them into the OpenFlow fabric