Eidgenössische
Technische Hochschule
Zürich

Ecole polytechnique fédérale de Zurich
Politecnico federale di Zurigo
Swiss Federal Institute of Technology Zurich

Department of Information Technology
and Electrical Engineering

Networked Systems Group
Prof. Dr. Laurent Vanbever
Spring Term 2015

Semester Thesis

Improving the Scalability of Software-Defined Internet Exchange Points

Author:
Rüdiger Birkner

Advisors:
Prof. Dr. Laurent Vanbever
Vasileios Kotronis

Abstract

Current Internet exchange points (IXPs) offer only limited control over how the traffic is forwarded. The decision on where the traffic is delivered to is solely made based on the destination IP prefix. The Software-Defined Internet exchange point (SDX) enhances existing IXPs by providing a fine-grained control over the forwarding of the outgoing and incoming traffic. The participants are presented with their own virtual SDX switch on which custom flow rules can be installed.

However, with the improved control comes limited scalability. The main issue of the first version of the SDX lies within the close coupling of BGP route information and SDN state, which leads to a complex and frequent policy compilation process. We address this issue by encoding the reachability of the prefixes into the destination MAC address of the packets. The notion of the VNH/VMAC changes from a simple tag, to a carrier of the reachability information of the respective prefixes. This approach reduces the dependence of the SDN state on the BGP route information to a minimum while providing all the functionalities of the first version. As a result, the policies only have to be recomputed rarely and the compilation becomes a very simple process.

Acknowledgments

I would like to thank my advisors Prof. Dr. Laurent Vanbever and Vasileios Kotronis for the good support. Further thanks go to the Networked Systems Group (NSG) at the Swiss Federal Institute of Technology (ETH) Zürich for the opportunity to work on this interesting topic.

Zürich, May 29, 2015 Rüdiger Birkner

Contents

Abbreviations

API application programming interface

ARP address resolution protocol

AS autonomous system

BGP Border Gateway Protocol

BIRD BIRD Internet routing daemon

ETH Eidgenössische Technische Hochschule (Swiss Federal Institute of Technology)

FEC forwarding equivalence class

IP Internet Protocol

IXP Internet exchange point

JSON JavaScript Object Notation

NSG Networked Systems Group

OF OpenFlow

REST representational state transfer

RIB routing information base

RIS Routing Information Service

RS route server

SDN Software-Defined Networking

SDX Software-Defined Internet exchange point

TCP Transmission Control Protocol

UDP User Datagram Protocol

VNH virtual next hop

VMAC virtual MAC address

Chapter 1

Introduction

Internet exchange points (IXPs) are infrastructures that interconnect different networks to exchange traffic amongst them. The participants are interconnected through a shared layer-two network and exchange their routing information via BGP-speaking border routers. To reduce the number of BGP sessions required, a BGP route server (RS) is operating at the IXP. Each participant connects to that RS and advertises its routes to it. The best path for each destination is selected by the RS on behalf of each participant based on the available routes and the local preferences.

BGP allows to select a single best path per destination prefix. However, the best path for web traffic, might not be the best path for video streaming traffic. In addition, there exist several techniques to influence the selection of the best path both for the outgoing traffic of a participant (local preference) and the incoming traffic (AS Path Prepending, MED). Nevertheless, those techniques only allow to influence the decisions. The final decision is always made by the RS. Hence, the decision might be suboptimal for the participant.

In 2014, Gupta et al. [4] have introduced a Software-Defined Internet exchange point (SDX): An enhanced IXP that takes advantage of the features of Software-Defined Networking (SDN). The traditional switch is replaced by a SDN switch and a controller is added to the exchange point. Through the combination of an existing IXP architecture and upcoming SDN technology a more fine-grained control over how traffic can be directed is feasible. It is now possible, instead of only influencing the selection of the best path on a per-prefix basis, to directly select the best path on a per-flow basis. A flow can be any traffic with some shared characteristics (e.g. the same TCP destination port and destination IP address).

The SDX is a first prototype that successfully shows the benefits of enhancing an IXP with SDN. Nevertheless, the first version of the SDX had several limitations hindering the deployment in production IXPs. AMS-IX one of the largest IXPs in Europe, for example, has over 700 participants [2]. The SDX, however, can only support a few

participants in parallel. When starting the project the main goal was to address this issue and enable the route server to cope with several hundred of participants.

The final policy of the controller is composed of all the policies of the participants and the BGP route information. This composition process is computationally intensive and limits the performance of the SDX. We realized that the policy composition is a further and even more severe limitation. Hence, the focus of the thesis shifted from improving the route server to finding ways to avoid the close coupling of the BGP route information and SDN state through the policy compilation.

The following chapter describes the SDX as proposed by Gupta et al., In the third chapter, we describe our first approach at improving the RS. This chapter served as a stepping stone through which the real limitations of the SDX had been identified and lead to the fourth chapter that describes the proposal of our new SDX architecture.

Chapter 2

Previous Work

2.1 Idea

This thesis builds on the SDX introduced by Gupta et. al. The SDX is an enhanced IXP using an SDN switch and a controller. Like a traditional IXP, it offers to interconnect autonomous systems (ASes) and operates a BGP route server. Furthermore, it allows each participant to specify policies on how the traffic has to be delivered. Each participant is given very fine-grained control over where its traffic is forwarded to. The combination of BGP route information and SDN techniques allows the participant to not only use the best path of a prefix, but to use any available path for that prefix.

The following paragraphs describe the most important aspects of the SDX.

2.1.1 Participant Policies

Through a simple abstraction built on Pyretic [7], each participant can express how both its incoming and outgoing traffic is handled. Each participant has the illusion of having the full control over its own virtual switch due to the abstraction. In the end, the SDX controller combines all participant policies into a single, coherent one.

Having a virtual switch per participant ensures that each participant can only specify how its outgoing and incoming traffic is handled. It is not possible to interfere with the traffic of another participant.

The policies are enforced by the controller by taking the available BGP routes into consideration. Even if a participant's policy tells the SDX to forward certain parts of the traffic to one participant, it is only forwarded there if that participant advertised a route for this traffic. Hence, no traffic is lost or dropped. To achieve this behavior, the SDX augments the participants' policies with the available BGP routes.

All the default BGP routes are kept as fallback routes. All the traffic that is not covered by a policy is forwarded on the best path as with traditional BGP.

2.1.2 Virtual Next Hops

Recall that the participants' policies are augmented with the BGP routing information. By simply augmenting the policies with the available routes, the number of flow rules explodes. Therefore, the concept of virtual next hops (VNHs) and virtual next MAC addresses (VMACs) is introduced.

The data-plane state is being reduced by combining IP prefixes with the same forwarding behavior throughout the SDX fabric into so-called forwarding equivalence classes (FECs). When augmenting the policies, the controller does not have to take every single available route into account, but only has to check the FEC. Since an FEC might consists of many non-adjacent prefixes, it is hard for the controller to somehow group them. Therefore, all the flows belonging to the same FEC are tagged by using the respective VMAC.

The tagging of the traffic is being offloaded to the participants' border routers. Each border router already maintains a forwarding table with an entry for each destination. The forwarding tables are populated through the BGP advertisements received from the route server. Hence, by replacing the actual next hop with the VNH, the appropriate tag is added to the flows.

When a border router needs to send traffic to a certain destination, it will check the next hop and consequently request the MAC address through an ARP request. The ARP request is handled by the controller and the corresponding VMAC is sent in the reply. Therefore, all the flows going through the SDN switch are tagged with the VMAC in the destination MAC address field. The switch immediately knows to which FEC the flow belongs to.

2.1.3 Incremental Deployment

The beauty of the SDX lies in the possibility of an incremental deployment and the backwards compatibility. The SDX offers the same capabilities and services to its participants as a regular IXP does. Participants that do not want to take advantage of the extended capabilities, do not have to and can just use the IXP as they did before.

2.2 Implementation

The SDX consists of two tightly coupled parts as shown in Figure 2.1: the RS and the policy composer. Both parts are written in Python. The SDX controller runs on top

of Pyretic which in turn runs on POX [1]. It only supports OpenFlow (OF) version 1.0 [6].

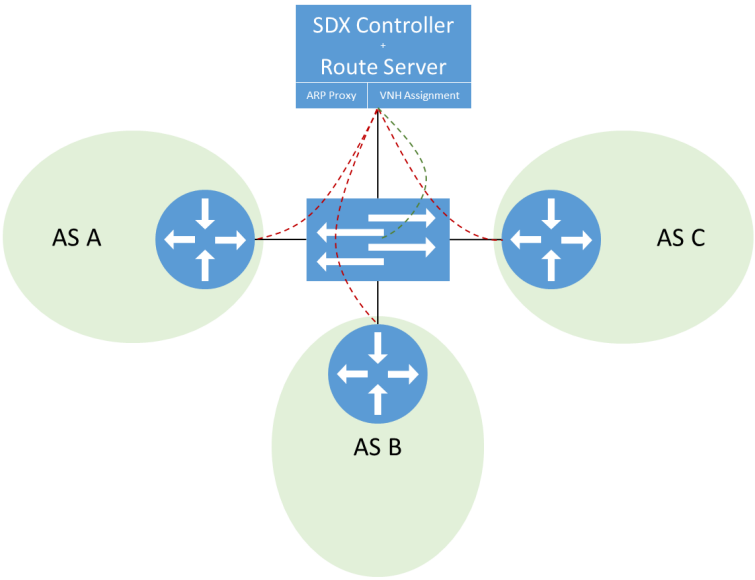

Figure 2.1: The SDX controller integrates a route server.

The RS relies on ExaBGP [5], which is a handy tool to inject BGP messages and transform them into text/JSON. It establishes and maintains a BGP session with each of the participants and takes care of the correct advertisement of the routes.

The policy composer takes the routes learned through the RS and the policies specified by the participants and computes the SDX policy. The computation of the SDX policy includes the assignment of the VNHs and the composition of the final Pyretic policy. The VNHs are advertised back to the participants through the RS as next hops for the routes.

2.3 Issues

2.3.1 Scalability of the Route Server

Currently, the SDX relies on a RS consisting of ExaBGP as its BGP speaker and SQLite databases as routing information bases (RIBs). This solution is not optimized and especially ExaBGP is unable to cope with several hundred BGP sessions at the same time efficiently, since it has not been built to act as a RS.

2.3.2 Coupling of SDN State and BGP Route Information

The final Pyretic policy is composed of the participant policies and the BGP route information. This close coupling of the BGP information and the policies leads to the fact that the Pyretic policy has to be recomputed whenever the reachability of a single prefix changes (e.g. a participant withdraws or announces a route).

2.3.3 Redundant Route Advertisements

Not only the policy has to be recomputed with every change in the RIB, but also the VNHs might change and therefore, the routes have again to be advertised to the participants. However, establishing and maintaining many BGP sessions with many participants is asking too much of the current RS. Also, some VNHs change, others do not. The current setup advertises each route to every participant even if the VNH did not change. It is not a matter of correctness, but of the unnecessary burden that is placed on the RS.

2.3.4 Correctness of BGP Advertisements

A traditional RS collects all the route advertisements of its peers, computes the best paths on behalf of each of its participants and then advertises them back to its peers. A BGP route advertisement includes the prefix of the route, the next hop through which the prefix can be reached, the set of ASes that have to be traversed to reach the prefix (AS Path) and a few more items. It is highly critical that always the correct AS path is advertised due to the fact that the AS path is used to detect forwarding loops.

However, in contrast to a traditional RS, where a single route is advertised for each prefix, we use the BGP advertisement in the SDX for something it was not meant for: We advertise not one, but multiple routes in a single announcement. Therefore, we have to take extra care of the advertised AS path. In the current setup, only the AS path of the best route is advertised while the other AS paths are just omitted. This may lead to forwarding loops in case the participant advertises the received route to other ASes.

2.3.5 Policy Composition

In addition, the policy composition is computationally intensive. First, the participants policies have to be augmented with the BGP route information. Then, the FECs have to be computed and the VNHs are assigned. All the outbound policies have to be combined with all the inbound policies. Lastly, the Pyretic policy is compiled into flow rules.

Chapter 3

Improved SDX

3.1 Idea

By offloading the RS capabilities to an external production grade RS, we hoped to improve the scalability and performance of the SDX. The core was not modified: the VNH assignment and the policy composition have been taken from the first version of the SDX.

In Figure 3.1, the setup of the improved SDX is shown. Next to the SDX controller, a RS is running. The route server establishes and maintains the BGP sessions with all border routers. All the routing information is passed on to the SDX controller. Based on this information the VNHs are assigned, the policies enforced and the ARP requests answered as it was done before.

3.2 Implementation

The RS and the SDX controller are two separate entities. The routing information acquired by the RS has to be sent to the controller and the new route advertisements containing the VNHs have to be sent back to the RS. This communication is based on a BGP session between the two parts.

The improved version uses BIRD Internet routing daemon (BIRD) as a RS. BIRD [3] is a production grade RS that is also used at several IXPs including AMS-IX, DE-CIX and LINX.

In order to account for the different policies and preferences in terms of best path selection a participant may have, a lot of information has to be exchanged. To correctly compose the policies, the controller needs to know to pieces of information from the route server for each participant:

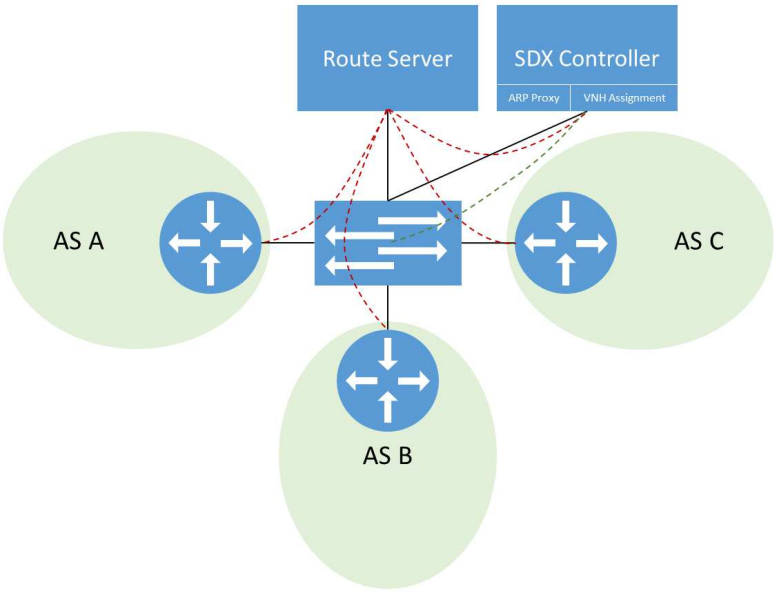

Figure 3.1: The route server is separated from the SDX controller.

All the routes that have been advertised to that participant

The best path for each prefix

When the routes received by the RS form a full routing table ($\tilde{5}$00'000 routes), a lot of data has to be passed on from the RS to the SDX. Since we are using BGP for the communication between the two, only one route can be transmitted per prefix. To send several advertisements per prefix, we decided to use the BGP extension add-paths [10] which allows to send several routes for a single prefix.

To get all the required information, quite a complex structure of routing tables is necessary. Figure 3.2 shows the setup when only a single participant is connected. With each additional participant, we have to add two new tables (TXXXM and TXXXB) and three pipes to connect them.

Also when the controller receives the routes from the RS, it somehow has to identify to which participant the route belongs to. Once the controller knows that, it still has to identify whether that route is just a route that has been advertised to the participant or whether it is a best path for a prefix. To achieve this, community based tagging is used.

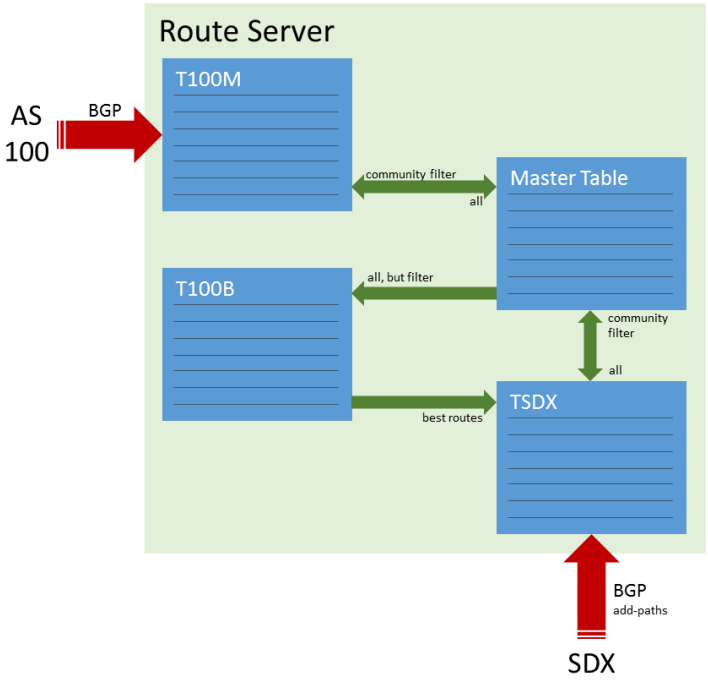

Figure 3.2: Route server configuration.

3.3 Improvements

Besides having a production grade RS, the number of redundant BGP advertisements is reduced. After a change in the overall Pyretic policy, the routes are only advertised to the respective participant if the route changed. This simple change reduces the number of route advertisements drastically.

In addition, instead of advertising the AS path of the best path, an AS set [8] is used to combine all AS paths of all advertised routes to avoid forwarding loops.

3.4 Issues

The current setup builds on the first prototype and therefore manages only to patch some of its limitations, but does not manage to do away with them. The work on the "improved" SDX even revealed new issues:

3.4.1 Information Exchange

While reducing the number of BGP sessions at the SDX controller compared to the first prototype, the number of route advertisements received and sent have not been reduced. Still the same information is necessary and has to be exchanged. Instead of receiving this information through many sessions, it is all received through a single session.

In addition, every route advertisements is being sent twice: Once from the participant to the RS and a second time from the RS to the SDX core.

3.4.2 Lack of Customization

In addition, the current setups lacks customizability. It is not possible for a participant to specify policies for custom best path selection, nor to specify with which other participants the routes should be exchanged. Both features increase the complexity of the setup.

3.4.3 Memory

The number of routing tables increases linearly in the number of participants. A lot of information is duplicated or copied and slightly modified.

Chapter 4

Supercharged SDX

While addressing one limitation, the RS, the improved version of the SDX drew our attention to other issues. The close link of the RS and the SDN controller for augmenting the policies provided by the participants with the routes advertised via BGP is one of the main limitations of the first version of the SDX. The augmentation leads to huge *dynamic* SDN policies containing millions of clauses that are hard to compile and recompile efficiently.

We realized that we have to break this link and thought of ways to decouple the RS from the controller. Ideally, breaking this link brings the following improvements with it:

Policy Compilation
The Pyretic policy is solely compiled on the basis of the participant policies not the BGP prefixes. Even when the reachability of a prefix changes, the policy does not have to be recomputed.

Information Exchange
No or very little information has to be sent from the route server to the controller and vice versa.

While investigating other architectures of the SDX, we carefully tried to retain its most important features of the first version:

Participant Policies
Each participant is able to control on a per-flow basis where its traffic is sent to.

Incremental Deployment
Peers at an SDX are not required to participate and can simply use the IXP as any other traditional IXP.

4.1 Idea

In the following paragraphs, we first explore the minimal requirements of an SDX setup and then propose one approach that addresses the mentioned limitations.

4.1.1 Participant's Policies

The SDX allows a participant to specify two kinds of policies: inbound and outbound policies. While inbound policies do not depend on the knowledge of the advertised routes, outbound policies do.

A possible outbound policy of a participant A is for example:

```
match(dstport=80) >> fwd(C)
```

In this case, all the traffic from participant A destined to TCP port 80 should be forwarded to participant C. However, this should only be done, for those flows which belong to an IP prefix that participant C has advertised. All other flows should be directed to the participant offering the best path for the respective flows instead.

4.1.2 Minimal Information Required by the SDN Controller

In order to implement outbound policies correctly, the SDN controller needs to know two things:

1. **Per-Prefix Reachability**
   The controller needs to know through which participants a prefix is reachable.
2. **Per-Prefix Best-Path**
   The controller needs to know which participant offers the BGP best path to direct default traffic to it.

This information is already available to the RS and has to be passed on to the SDN controller. Currently, this information is exchanged using plain BGP routes which are then used to augment the SDN policy, leading to potentially millions of clauses. Our goal is to minimize the information exchanged and remove the need to augment the policy with any IP prefix information.

To do so, we propose to offload the communication between the RS and the controller by using the VNH/VMAC address as a carrier of the information. In this new scheme, the VNH will carry both the reachability and the best path information. The notion of the VNH/VMAC changes from the first SDX version, where it identified the FEC, to this proposal where it contains both the reachability of the prefix and its best path. Unlike the original SDX, it means that the RS will be responsible for computing the VNH/VMAC, not the SDN controller.

At a high-level, our scheme works like this: Dedicated MAC addresses (provisioned using the VNH) are used to represent all prefixes reachable via X where X is a SDX participant. Dedicated MAC addresses are also used to represent All prefixes for which X is the best. Once the SDN controller knows the dedicated MAC addresses, it can use them to augment the participant's policy to achieve the desired behavior:

```
(match(dstport=80) & match(dstmac="reachable through C")) >> fwd(C)
```

We stress that these MAC addresses are static and can be defined once and for all. As prefixes come and go, the RS simply updates the VNH and corresponding VMAC to reflect the change in reachability. Most of the time upon updates, the SDN policy does not even need to be changed! For instance, if C can reach `p1`, `p2`, `p3`, the corresponding VNH will be mapped to the VMAC reachable through C. If C cannot reach `p1` anymore, the VNH changes, and the corresponding traffic will not match the policy anymore.

4.1.3 Matching on the MAC Address

Strawman:
The easiest way to encode the reachability into the VMAC, is to dedicate one bit to each participant. If that bit is set, the prefix is reachable through that participant and if it is not set, the prefix cannot be reached through the participant. Then, the controller would just have to match on the bit of the participant to see whether that flow can be directed to that participant or not. Observe that this forwarding is achievable as OF version 1.3 [6] now supports arbitrary bit-mask matching on MAC addresses.

Let's Aggregate to Make it Work:
A MAC address has only 48 bits and the IXP might have participants in the order of few hundreds. By an empirical analysis of the routing tables provided by the Routing Information Service (RIS) of RIPE NCC (cf. Appendix A), we observed that the cardinality of the set of participants advertising the same prefix *is not larger than 20*. While it is not possible to dedicate one bit to each of the participants, it is certainly possible to dedicate one bit to each of the participants of one set.

When dedicating one bit to each participant of a set advertising the same prefix, the controller only needs to know the participants in the set and their positions and can then consequently match on a single bit to check whether that flow can be redirected to a certain participant or not. Hence, the RS has to notify the controller about each set and its members.

4.1.4 Supersets

Each prefix is advertised by a set of participants. By simply using all those sets as proposed above, a lot of information needs to be passed on from the RS to the controller. The controller needs to know each set and all the participants within. However, the

number of the sets can be reduced simply by removing all the sets that are subsets of others. In addition, combining some of the remaining sets to so-called supersets reduces the number of sets even further. The maximum possible cardinality of the supersets is limited by the length of the MAC address. The cardinality of the union of several basic sets is not much larger than the cardinality of the largest basic set, if the intersections of the sets are large.

Consider the following example: prefix `p1` is advertised by participants A, B, C, D, E and prefix `p2` is advertised by participants B, C, D, E, F. By combining the sets of both prefixes, we get a superset which only contains one more element A, B, C, D, E, F and both sets can be built using that superset.

Therefore, it is possible to combine all basic sets to get a minimal number of supersets of fixed size that cover all basic sets. As a consequence, only the supersets and their members have to be communicated to the controller.

For an example of the Subset/Superset at play, see Section 4.1.9.

4.1.5 VMAC Encoding

We split the 48 bits of the MAC address into two parts as shown in Figure 4.1: one to encode the reachability of a prefix (superset, see Section 4.1.4) and the other one to encode which participant is best.

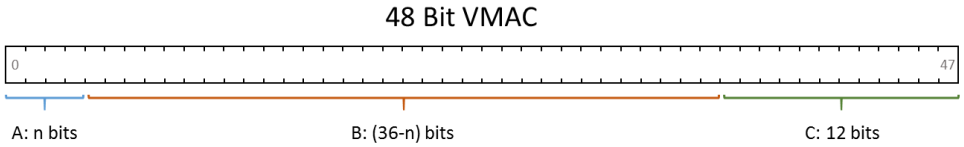

Figure 4.1: VMAC encoding.

1. **Reachability of a Prefix**
   The SDX controller needs to know which set of participants has advertised the prefix. In other words, it has to know through which set of participants a certain prefix is reachable. We have seen that the maximum cardinality of such a set is 20. There are 36 bits dedicated to represent such a set. The first n bits (Part A) identify the so-called superset. The remaining (36-n) bits represent each participant within that superset. The supersets are being formed by forming the union of several of the sets that advertise the same prefix until the cardinality of the superset reaches 36-n. This allows to represent each set that has advertised a prefix by using one of the supersets.

2. **Best Route for a Prefix**
   Each participant is assigned a unique identifier between 0 and 4095. Part C of

the VMAC is used to signal the best route by just containing the identifier of the respective participant.

4.1.6 Information Exchange

Using the proposed encoding, only two pieces of information have to be passed from the RS on to the controller to provide full functionality.

1. **The Supersets and their Members**
   The controller needs to know the identifier of each superset and its members.

2. **The Participant Identifiers**
   To identify the best route for each flow, the controller needs to know the identifier of each participant.

Unlike the original SDX, this information is *mostly static*:

> Route withdrawals do not affect the supersets. Route announcements only affect the supersets, if a participant starts announcing a route and the resulting set of participants announcing this route is not a subset of any of the supersets.
>
> The supersets change only when a new participant appears or leaves (daily or weekly frequency at best). Through smart generation of the supersets, the changes can be minimized.

4.1.7 Selective Peering

Since the VMAC carries all the reachability information and the VMAC is customized for each participant, it is simple to account for participant's policies. Not only can the participants specify to whom their routes are advertised, but also through the VMACs it is ensured that no participant can maliciously send its traffic through a participant that does not want that traffic.

The following example illustrates this: We have an SDX with five participants A, B, C, D, E. Prefix `p1` is announced by C, D, E. C and D both want to advertise their routes to A, but not to B. E in contrast advertises its routes to B, but not to A. As a result, the VMAC for `p1` that A will receive, has both the bit of C and D set. Hence, it is possible for A to forward the traffic for `p1` to C or D, but not to E. B receives a VMAC for `p1` where only the bit of E is set. B can only forward its traffic for `p1` to E, even though `p1` is also reachable through C and D.

4.1.8 Modified SDX Setup

The modified SDX consists of two separate parts as show in Figure 4.2: the RS and the SDX controller. Unlike the original SDX, the VNH assignment and address resolution protocol (ARP) proxy is part of the RS, not the controller.

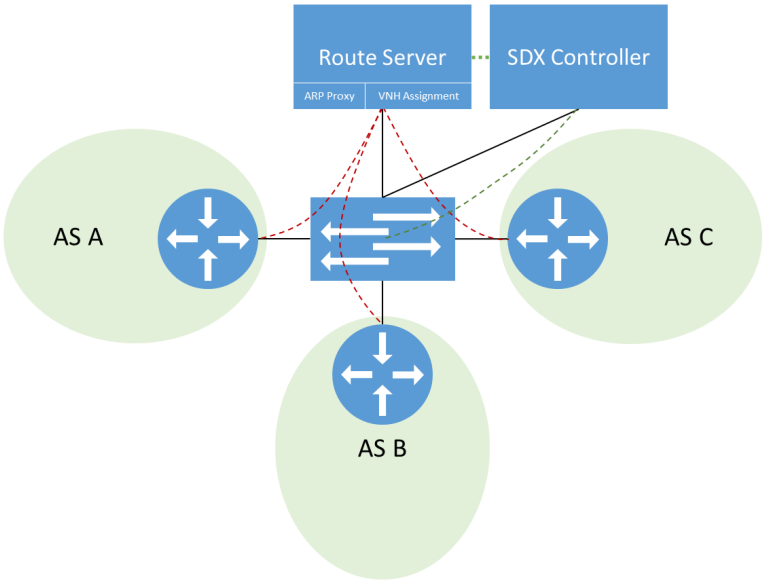

Figure 4.2: The route server is separate from the SDX controller and additionally assigns the VNHs.

Route Server:
As before, all the participants establish a BGP session to the RS. In the RS, the supersets and consequently the VNHs/VMACs are being computed. An ARP proxy replies to all the ARP requests by the participants by sending the corresponding VMAC.

Controller:
The controller parses the participant policies and generates the Pyretic policy using the pre-provisioned VMACs.

4.1.9 Example

We have an SDX with 5 participants (A, B, C, D, E). The participants advertise the following prefixes:

Table 4.1: Advertised Routes

Participant	Prefix
A	-
B	p1*, p2*
C	p1, p3*, p4
D	p1, p4
E	p2, p4*

The asterisk indicates the best path for that prefix.

Participant A specified the following outbound policy:

```
match(dstport=80) >> fwd(D)
```

For the sake of simplicity, we assume that the MAC addresses have a length of 9 bits. 3 bits are dedicated to the best route. Of the remaining 6 bits, 3 bits are used to identify the superset and 3 bits are used for the participants within that superset.

The sets of participants advertising the same prefixes are:

```
p1: (B, C, D)
p2: (B, E)
p3: (C)
p4: (C, D, E)
```

This leads to three supersets of size 3:

```
0: (B, C, D)
1: (B, E)
2: (C, D, E)
```

The identifiers of the participants A, B, C, D, E are 0, 1, 2, 3, 4, respectively.

This leads to the following VMACs for the four prefixes:

```
p1: 000 111 001
p2: 001 110 001
p3: 001 010 010
p4: 010 111 100
```

The first three bits identify the superset, the following three bits represent the participants within that superset and the last three bits identify the participant offering the best route for that prefix.

In order to implement participant A's policy, we have to check whether the destination of the flow is reachable through D. We just have to match on the specific bits. However, since there are two supersets that contain D, we have to match on both supersets:

```
if_((match(dstmac=000XX1XXX) + match(dstmac=010X1XXXX) >>
match(dstport=80)), fwd(D), fwd_to_best_path())
```

The Xs in the match statement represent "don't care" bits

In more detail, the following rules result from the given setup:

```
1: (match(dstport=80) + match(dstmac=000XX1XXX)) >> fwd(D)
2: (match(dstport=80) + match(dstmac=010X1XXXX)) >> fwd(D)
3: match(dstmac=XXXXXX000) >> fwd(A)
4: match(dstmac=XXXXXX001) >> fwd(B)
5: match(dstmac=XXXXXX010) >> fwd(C)
6: match(dstmac=XXXXXX011) >> fwd(D)
7: match(dstmac=XXXXXX100) >> fwd(E)
```

Rules 1 and 2 are due to participant A's policy. Rules 3 to 7 are installed to ensure BGP best path routing. We can see that a participant flow rule may lead to several actual flow rules. This is due to the supersets. If one of the participants (in this case D) appears in several supersets, we have to have a match on every superset in which D appears. To ensure BGP best path routing for all the flows that are not handled by one of the participant's policies, we only need to add one flow rule per participant.

4.2 Implementation

The supercharged SDX is written in Python and consists of two main parts: an extended RS and the SDX controller. The RS has been extended by the superset computation, VNH assignment and an ARP proxy. The SDX controller is an app built on the Ryu controller [9]. It parses the participant policies and enforces them.

4.2.1 Route Server

BGP Speaker
As in the previous versions, we are using ExaBGP to establish the BGP sessions and exchange the BGP route information.

In the global settings, each participant can specify to which other participants it would like to advertise its routes. The RS takes this into account and sends custom route advertisements to each participant.

It is especially important to send the correct AS set with each route advertisement. When sending a route advertisement to a participant, the RS consults the RIB to get all the different routes that have been advertised to this participant and combines all AS paths into an AS set.

To reduce the number of advertisements that are being sent out by the SDX whenever the reachability of a prefix changes, it is checked for which participants the advertised route changed. Consequently, the advertisements are sent only to those participants.

ARP Proxy
For simplicity reasons, we assign each prefix a single VNH. Once an ARP request is received by the ARP proxy, it is determined which participant sent it and then the VMAC is computed accordingly. If two participants send an ARP request for the same prefix, they will get a different, custom VMAC based on their peerings and preferences.

In order to further reduce the number of VNHs, prefixes can be grouped by the sets of participants advertising them.

When the reachability of a prefix changes, not only the route but also the VMAC changes. Since the VNH is the same for all participants and does not change, we need to send a gratuitous ARP reply to the participants to ensure that the VMAC is adjusted.

Superset Computation
In the previous chapter, we showed how the supersets are computed in a static setting (e.g. when all routes have already been advertised). However, in reality the supersets have to be computed dynamically: routes are announced and withdrawn and hence, the optimal supersets might change. We have chosen a pragmatic approach:

Whenever a new route is advertised by one of the participants, the set of participants advertising exactly that prefix is computed. If this set is already a subset of one of the supersets, we can stop here. Otherwise, we check to which superset we have to add the least members such that the set is a subset of the superset while staying within the limit imposed by the size of the MAC address. In case, it is not possible to extend an existing superset, a new superset is added.

This approach allows us to only slightly change the supersets and therefore, we do not have to change any of the existing flow rules installed by the controller. However, the computed supersets might not be optimal.

Therefore, it is possible to specify a threshold such that if the number of supersets exceeds this threshold, the supersets are completely recomputed. At that point the vast majority of the routes has been advertised and we can assume that this is the static case. Hence, we are able to compute the optimal number of supersets. Nevertheless, all the outbound flow rules have to be deleted and newly installed according to the new supersets.

4.2.2 Controller

Since our approach requires bitmask matching on MAC addresses, we have to use OF version 1.3 and above. Pyretic runs on POX, which only supports OF version 1.0. As a consequence, we decided to move from Pyretic to Ryu. Besides the bitmask matching,

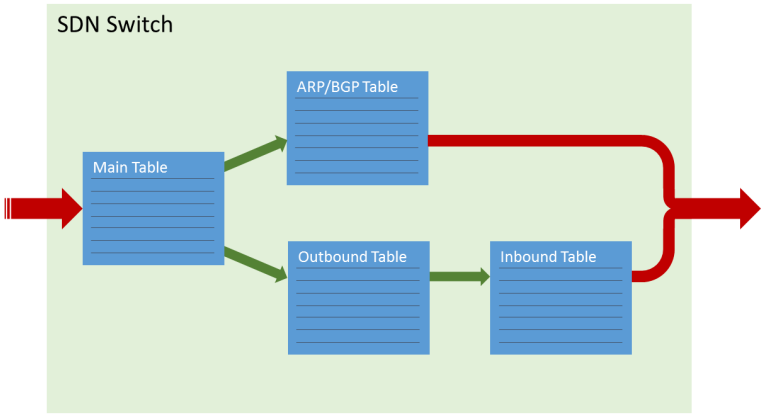

Figure 4.3: Flow tables.

we are also using multiple pipelined flow tables, which has the favorable side-effect of eliminating the need for the composition of the inbound and outbound policies.

Flow Tables
In total four flow tables are installed in the switch. The longest pipeline is of length three.

Main Table
All incoming traffic first goes through the main table. All the ARP and BGP traffic is directly sent to the ARP/BGP table. All the other traffic is tagged with the originating participant. The tagging is necessary in order to reduce the number of flow rules in the following stages, since each participant might be connected to the SDX through several ports. Afterwards the traffic is sent to the outbound table.

Outbound Table
The name of this table is misleading. It is called "outbound" not because the traffic leaves the switch through this table, but because it contains the flow rules that stem from the participants' outbound policies.

For each outbound policy of a participant (e.g. match(dstport=80) $\gg$ fwd(B)), one flow rule per superset containing B is installed. To install these flow rules, we need to know the supersets. This is the only information that the controller has to be provisioned with beforehand. Whenever, the supersets change, the outbound policy flow rules have to be adjusted.

In addition, to the policy flow rules, there is one rule per participant that matches on the best path section within the VMAC. By this mean, it is ensured that all the flows that are not affected by one of the outbound policies, are forwarded on the BGP best path.

Inbound Table

In the inbound table all the flow rules can be installed at start up, since they do not depend on the BGP route information.

ARP/BGP Table
For the correct operation of the SDX, it is important that all the BGP and ARP traffic can pass through the switch easily. Therefore, a L2-Learning switch is running on this table. In addition, all the ARP requests targeting a VNH are directly sent to the route server.

When the reachability of a prefix changes, the VMAC changes and a gratuitous ARP reply is sent out. A gratuitous ARP must be transmitted as a local broadcast packet. However, the VMAC is participant specific and it is not possible to broadcast the same gratuitous ARP reply to all participants. Therefore, we have a flow rule for each participant that matches on a special destination MAC address indicating the destination participant. It then rewrites it to the broadcast address and forwards it only on those ports connected to that participant.

REST API
The only information that has to be exchanged, are the supersets (the identifier and its members) which are sent from the RS to the controller. A representational state transfer (REST) application programming interface (API) has been implemented to allow the RS to update the supersets.

4.3 Improvements

4.3.1 Speedier Policy Compilation

Since the reachability of a prefix is encoded into the VMAC, we eliminated the need to augment the participants' policies with the BGP route information. In addition, due to the use of multiple flow tables within the switch, the in- and outbound policies of the participants do not have to be composed anymore. This reduces the complexity of the policy compilation into flow rules.

4.3.2 Reduced Flow Rule Churn

Thanks to the multiple flow tables, the in- and outbound policies do not have to be composed anymore. As pointed out before, only the outbound policies depend on the reachability information acquired by the RS. Hence, only those policies change over the time.

All the flow rules can proactively be installed in the switch and only change, if the policies of the participants change. And even if the policy of a single participant changes, only those flow rules linked to its policy have to change.

4.3.3 Correct BGP Advertisements

Since we combine all AS paths of the available routes into one AS set, we ensure that the BGP advertisements are correct and no forwarding loops occur.

4.3.4 One to One - Policy to Flow Rule Mapping

Since a switch can implement only a limited number of flow rules, an IXP operator might be interested in basing its pricing on the number of flow rules that are being used by a participant. By abandoning Pyretic, we increase the visibility of the policies within the flow tables. It is possible to immediately determine from which participant's policy a flow rule stems.

4.3.5 Selective Route Advertisements

By using the proposed encoding, it is simple to provide each participant with the option to selectively announce certain routes only to a subset of the participants at the SDX. If a participant does not want to advertise a prefix to another participant, the respective bit in the VMAC just has to be set to zero.

Chapter 5

Conclusion & Outlook

In the course of this project, we identified that the dependency of the policy composition on the BGP route information is a major limitation of the SDX.

We proposed a new approach by changing the notion of the VNH/VMAC. Instead of using the VMAC as a simple tag, we encode both the reachability and the best path of a prefix into the VMAC. With this encoding, the information that has to be exchanged between the RS and the controller before operation can be greatly reduced and the majority of the information is carried by the flows themselves.

In addition, the policy compilation got simpler and does not have to be performed upon each and every change in the RIB.

One of the biggest limitations of our proposal is the MAC address size of 48 bits. Currently, our proposed encoding only works if there are less than 36 participants at the IXP that advertise the same prefix. The analysis (cf. Appendix A) shows that the cardinality of the sets is at most 20. However, the analysed routing tables only represent a subset of the real routing table at the respective IXPs. A smarter encoding has to be found to alleviate this limitation as well.

The presented idea is a good basis for the next-generation SDX. There are still many extensions and improvements possible:

By having a closer link between the controller and the RS, we can on the one hand have more accurate BGP advertisements and on the other hand, improve the VMAC scalability. In our proposal, we combine all the AS paths of the routes that are available to a participant to an AS set. The policies of a participant will most likely not involve all possible routes. Therefore, by knowing what policies a participant specified, we can reduce the number of combined AS paths and have a more accurate AS set.

While an outbound flow rule is being replaced or changed, the switch falls back to forward the traffic on the best path. Some participants do not care about this, others only want to use a specific path and in no case another one. Therefore, we propose to

allow the participant to specify what should happen in case their policy is not or cannot be applied.

The policies currently allow only to match on certain characteristics of a flow and then forward it to a specific participant. A participant might want to specify not just the best participant to forward the flow to, but to specify a list of participants in order of their preference.

A limitation that we pointed out in the beginning is the scalability of the RS: Even in our approach, the RS is still based on ExaBGP which is not built to operate as a RS. Therefore, it is necessary to find an alternative for a production grade SDX.

Appendix A

Analysis of BGP Routing Tables

Table A.1: Analysis of Three Routing Tables of Different IXPs

IXP	AMS-IX	DE-CIX	LINX	Combined*
Date	2015/03/22	2015/04/01	2014/12/05	2015/04/20
# Advertised Prefixes	564379	50363	539718	610613
# Connected ASes	65	39	52	243
# Sets advertising a Prefix	423	190	399	4618
average Cardinality	7.9	14.7	10.2	50
maximum Cardinality	15	20	15	82
# Sets (after removing all subsets)	69	21	50	331
# Supersets of Cardinality 30	4	2	3	not possible

*Since only a few of the actual IXP participants provide their routing information, the data sets do not represent a full IXP routing table. Therefore, we tried to generate a larger BGP routing table by combining all available BGP routing tables. The combined BGP routing table contains the available routing tables of the following IXPs:

AMS-IX

CIXP

DE-CIX

JPIX

LINX

MIX

MSK-IX

NETNOD

NYIIX

PAIX

PTTMetro-SP

VIX

List of Figures

List of Tables

Bibliography

[1] (2015, May) POX. [Online]. Available: http://www.noxrepo.org/pox/about-pox/

[2] AMS-IX. (2015, May) Network (as number) statistics. [Online]. Available: https://ams-ix.net/connected_parties

[3] O. Filip, M. Mares, and O. Zajicek. (2015, May) Bird internet routing daemon. [Online]. Available: http://bird.network.cz/

[4] A. Gupta, L. Vanbever, M. Shahbaz, S. P. Donovan, B. Schlinker, N. Feamster, J. Rexford, S. Shenker, R. Clark, and E. Katz-Bassett, "Sdx: A software defined internet exchange," *SIGCOMM Comput. Commun. Rev.*, vol. 44, no. 4, pp. 551–562, Aug. 2014. [Online]. Available: http://doi.acm.org/10.1145/2740070.2626300

[5] T. Mangin. (2015, May) Exabgp - the bgp swiss army knife of networking. [Online]. Available: https://github.com/Exa-Networks/exabgp/tree/3.4

[6] Open Networking Foundation. (2015, May) OpenFlow switch specification. [Online]. Available: https://www.opennetworking.org/sdn-resources/technical-library#tech-spec

[7] J. Reich, C. Monsanto, N. Foster, J. Rexford, and D. Walker, "Modular SDN Programming with Pyretic," *USENIX ;login*, vol. 38, no. 5, pp. 128–134, Oct. 2013.

[8] Y. Rekhter, T. Li, and S. Hares, "A border gateway protocol 4 (bgp-4)," Internet Requests for Comments, RFC Editor, RFC 4271, January 2006. [Online]. Available: http://www.rfc-editor.org/rfc/rfc4271.txt

[9] Ryu SDN Framework Community. (2015, May) Ryu sdn framework. [Online]. Available: http://osrg.github.io/ryu/

[10] D. Walton, A. Retana, E. Chen, and J. Scudder, "Advertisement of multiple paths in bgp," Working Draft, IETF Secretariat, Internet-Draft draft-ietf-idr-add-paths-10, October 2014. [Online]. Available: http://www.ietf.org/internet-drafts/draft-ietf-idr-add-paths-10.txt