Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Institut für
Technische Informatik und
Kommunikationsnetze

Philipp Mao

Boosting the convergence performance of SDX platforms

Semester Thesis SA -2016-69
October 2016 to January 2017

Tutor: Laurent Vanbever
Supervisor: Rüdiger Birkner, Thomas Holterbach

Abstract

Internet outages in BGP are critical and lead to long convergence times .
Industrial-Scale Software defined Internet exchange Points in short iSDX, suffer like any other BGP speaking router from this problem. This is made even worse because of the additional overhead induced by the iSDX's participant policies. Existing fast reroute frameworks like Swift can help shorten convergence times. In this work we implement Swift into the iSDX. Swift can be easily implemented into the iSDX due their similar architecture.
Without a lot of additional overhead, the convergence time of the iSDX is improved by up to a factor 60. But as both Swift and the iSDX use the destination mac address as a data plane tag the amount of bits available to encode information for the iSDX and Swift is halved. This reduces the iSDX's ability to scale with higher number of participants.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The border gateway protocol is the glue that holds today's internet together. It allow autonomous networks to exchange information about reachability of prefixes without revealing their own network infrastructure.
Remote disruptions in BGP can lead to convergence times of multiple minutes. This long convergence time is caused by the way BGP updates are propagated through the internet. The convergence time is lower bounded by the time it takes for all BGP withdrawal updates to be received.
Software defined networking allows network operators to have more control over the network routing. It is also a technology that can help solve some of the internet's problems including long BGP convergence times.
In this work I will try to improve the convergence time of the industrial scale internet exchange point (iSDX) using the Swift framework, both rely on SDN switches to enable their functionality.

Chapter 2

Background

2.1 iSDX

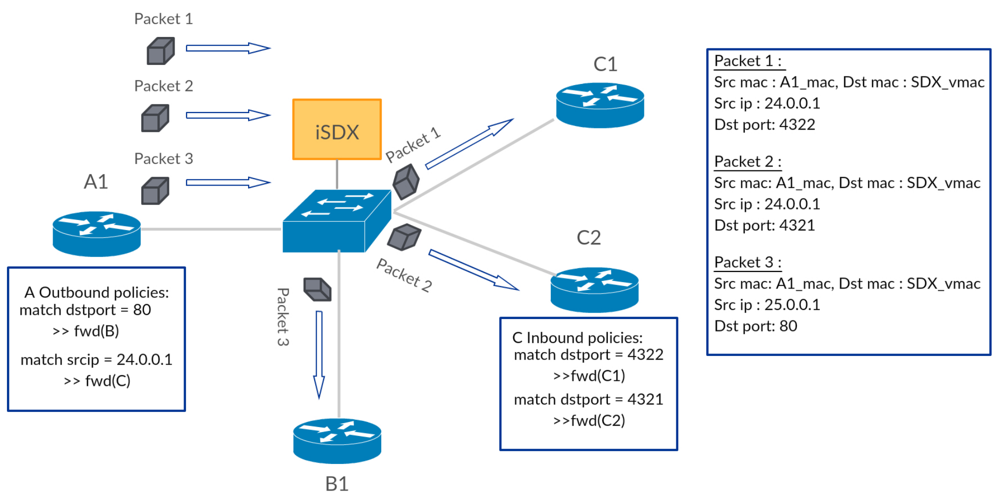
The iSDX is a framework for deploying software defined networking at internet exchange points. An internet exchange point is a physical location where multiple autonomous systems meet to exchange traffic and BGP routes.
Deploying the iSDX at an internet exchange point gives participants more control over routing decisions. This is done by giving participants the opportunity to define policies. These policies allow participants to influence traffic coming from and going in the internet exchange point without obscure BGP message manipulation. The iSDX uses an SDN switch connected to the iSDX and all the participants to implement the policies as flowrules.

2.1.1 policies

Policies match on a field of the packet header and then forward the packet to a participant.

outbound policies: Outbound policies let participants direct packets going from themselves to the iSDX. They allow the participants to choose the participant the packet gets sent to.

inbound policies: Inbound policies allow participants to direct packets coming from the iSDX. In effect they allow the participant to choose to which of his own routers packets from the iSDX get sent to.



A Outbound policies:
match dstport = 80
>> fwd(B)
match srcip = 24.0.0.1
>> fwd(C)

C Inbound policies:
match dstport = 4322
>>fwd(C1)
match dstport = 4321
>>fwd(C2)

Packet 1 :
Src mac : A1_mac, Dst mac : SDX_vmac
Src ip : 24.0.0.1
Dst port: 4322

Packet 2 :
Src mac: A1_mac, Dst mac : SDX_vmac
Src ip : 24.0.0.1
Dst port: 4321

Packet 3 :
Src mac: A1_mac, Dst mac : SDX_vmac
Src ip : 25.0.0.1
Dst port: 80

Figure 2.1: Example of outbound and inbound policies with an iSDX connected to three participants

2.1.2 virtual next-hop, virtual mac address

In order to not violate BGP advertisements (send packets to participants that did not advertise the prefix of the packet) the iSDX uses the destination mac address as a data plane tag.
In the destination mac address the iSDX encodes the participants advertising the prefix of the packet and the BGP best next hop participant for this prefix. This mac address corresponds to a virtual next hop, which is assigned to every prefix.
The iSDX sends BGP updates to the participants, with the next-hop of the update set to the virtual next-hop of the prefix. The virtual mac address is communicated to the participants via ARP.

iSDX vmac: | participants advertising the prefix | BGP best next-hop participant |

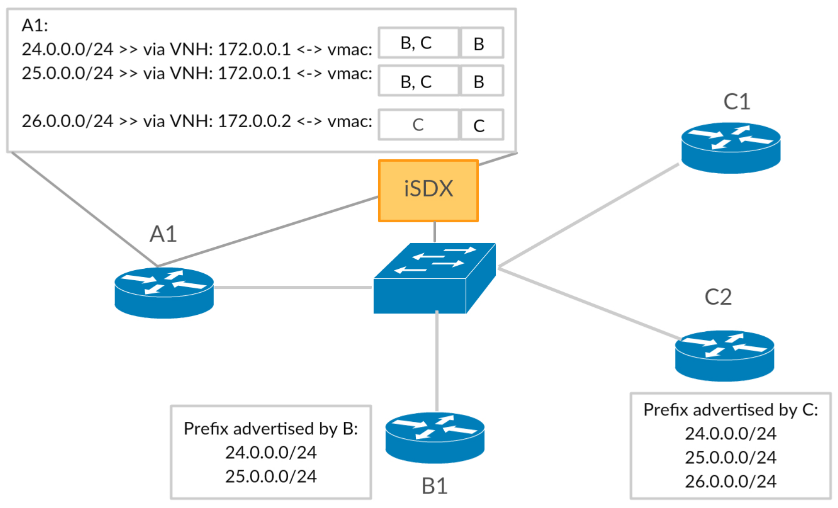

A1:
24.0.0.0/24 >> via VNH: 172.0.0.1 <-> vmac:
B, C
B
25.0.0.0/24 >> via VNH: 172.0.0.1 <-> vmac:
B, C
B
26.0.0.0/24 >> via VNH: 172.0.0.2 <-> vmac:
C
C
iSDX
C1
A1
C2
Prefix advertised by B:
24.0.0.0/24
25.0.0.0/24
B1
Prefix advertised by C:
24.0.0.0/24
25.0.0.0/24
26.0.0.0/24

Figure 2.2: Example of the iSDX vmac with an iSDX connected to three participants

2.1.3 iSDX architecture

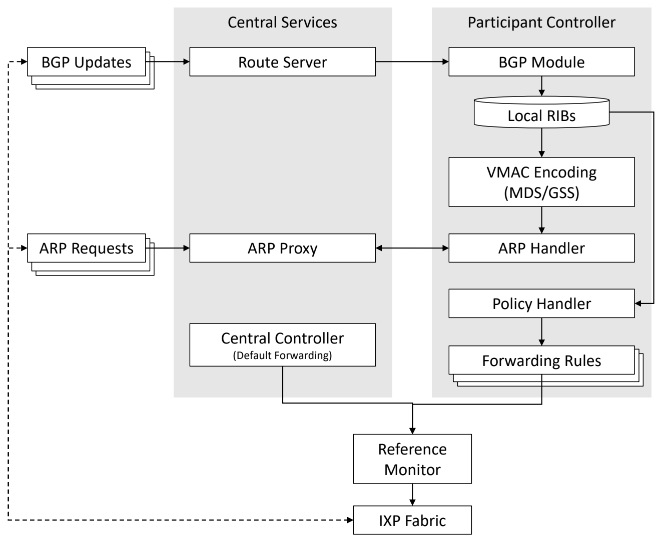


Figure 2.3: iSDX architecture

The iSDX architecture has two main Parts. The Central Services and the Participant Controller. Every participant has its own participant controller.
Central Services forwards BGP updates and ARP queries to the corresponding participant controller. It also initializes all the static flow rules.
The participant controller receives BGP updates from the Route Server. It then actualizes it's RIB checks if the virtual mac address was changed by the received BGP update, if so sends a gratuitous ARP to the ARP proxy. The participant controller also manages each participant's inbound and outbound policies.

2.2 Swift

Swift is a prediction and fast-reroute framework to improve the convergence time of a BGP speaking router upon remote failure. Swifts prediction relies on the fact that the cause of a burst of withdrawals can be determined before receiving all the withdrawals.
Swift uses a SDN switch connected to the swifted router, it's neighbors and to the Swift controller.

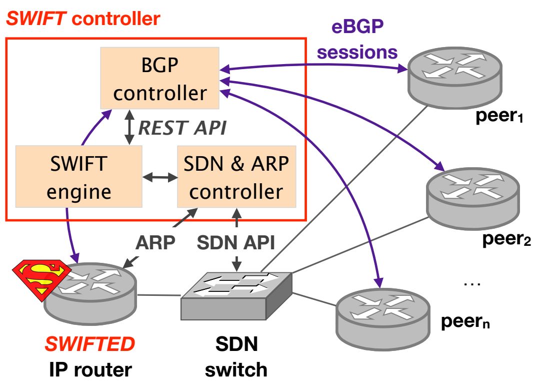


Figure 2.4: Example topology of a swifted router

The Swift Controller has two main modules: the burst prediction algorithm and the encoding of routing information.

2.2.1 encoding of routing information

Swift similarly to the iSDX uses virtual next-hops and the destination mac address to encode information about the packet's prefix.
For every prefix Swift encodes the AS-path up to a certain depth and the backup next-hops for each AS-link on that AS-path. This encoding is then mapped to a virtual next-hop. When the swifted router wants to send a packet to any prefix it will use the virtual next-hop assigned by Swift. The virtual next-hop directly maps to the destination mac address.

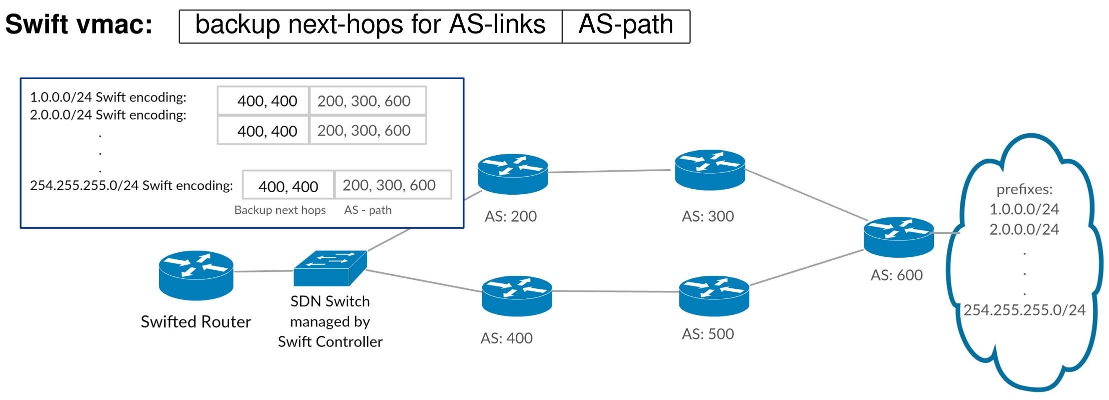

Swift vmac:
backup next-hops for AS-links
AS-path
1.0.0.0/24 Swift encoding: 400, 400 200, 300, 600
2.0.0.0/24 Swift encoding: 400, 400 200, 300, 600
254.255.255.0/24 Swift encoding: 400, 400 200, 300, 600
Backup next hops
AS - path
AS: 200
AS: 300
AS: 600
AS: 400
AS: 500
Swifted Router
SDN Switch managed by Swift Controller
prefixes: 1.0.0.0/24 2.0.0.0/24 254.255.255.0/24

Figure 2.5: Example of the Swift vmac

2.2.2 burst prediction algorithm

The burst prediction algorithm takes BGP updates. It uses the updates to build a AS-topology. Once enough withdrawals arrive in a time frame short enough to trigger a burst, the burst prediction algorithm uses the withdrawals and the AS-topology to predict the failed AS-link.
Upon predicting a failed link the Swift Controller pushes flow rules int the SDN switch matching on the failed AS-link and on the corresponding backup next-hop.
Every packet that traverses the failed link (has the failed AS-link in it's AS-path encoding) will get rerouted to the backup neighbor.

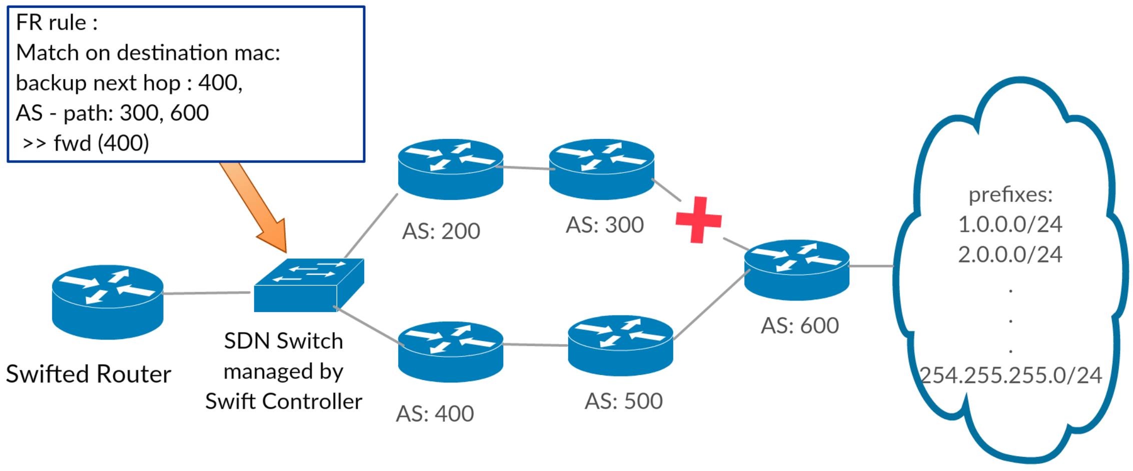


Figure 2.6: Example of a fast-reroute after BPA predicts the AS link 300 600 to be down

By predicting the failed link and pushing fast-reroute rules instead of waiting for all withdrawals to arrive, Swift reduces the convergence time of the swifted router significantly.

2.3 Motivation

The iSDX takes a long time to converge upon remote failure. The main problem is that the iSDX needs to do some additional computing a traditional BGP-router does not need to do.
When receiving a withdrawal the iSDX participant controller updates it's RIB, checks if the policy flow rules have changed and if the virtual next hop/vmac has changed. This additional computation adds a significant overhead to the convergence time of the iSDX.
Implementing Swift into the iSDX promises to significantly reduce the convergence time. Convergence time being the time until packets get sent to the correct participant and not into a black hole.
The main reason why implementing Swift in the iSDX is reasonable is their similar architecture. Both the iSDX and Swift use a SDN switch to program flow rules to steer traffic. They both are connected to BGP speaking routers and receive BGP updates from them. Swift and iSDX both use the destination mac address to encode information about a prefix and use the next-hop to map this vmac to a prefix. In addition the swift framework allows multiple swifted routers to be connected to the SDN switch, which in the iSDX's case means that all the participants will benefit from Swift.

Chapter 3

iSDX with Swift

The main design prinicple when implementing Swift into the iSDX is to change as little as possible in both the iSDX and Swift. Utilizing their similar architecture to keep the full functionality of both the iSDX and Swift.

3.1 Architecture

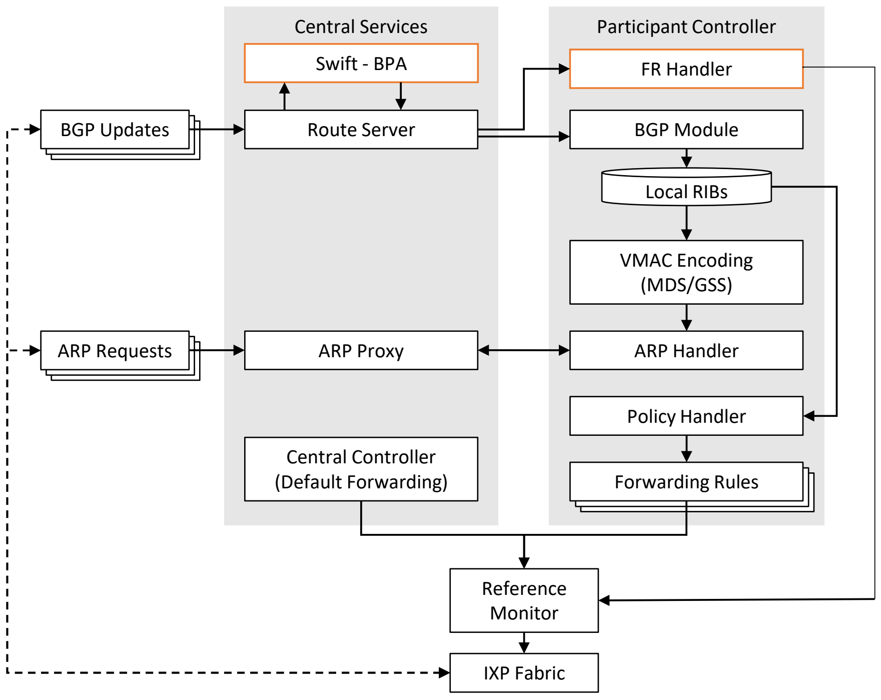


Figure 3.1: iSDX architecture with Swift

Figure 3.1 shows the iSDX architecture with Swift. The orange modules are new modules added to the iSDX to implement Swift. The iSDX receives two additional modules the Swift-BPA module in the central services and the FR handler in the participant controller. With these two modules the iSDX will now detect bursts of withdrawals, predict the failed AS-link and push fast-reroute rules into the IXP fabric.
In the following sections I will go over the functionality of the new modules and other changes to the iSDX in more detail.

3.2 Swift-BPA

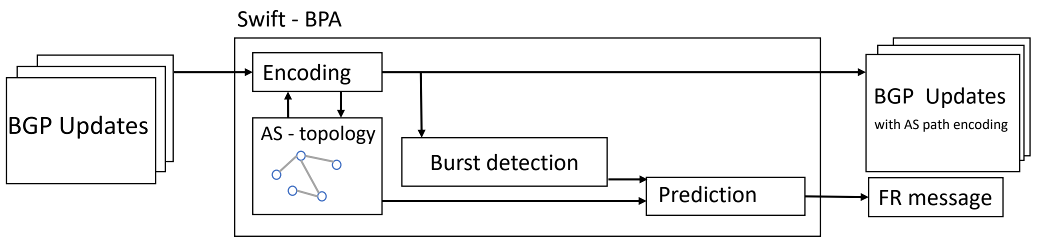


Figure 3.2: pipeline of the Swift-BPA module

The Swift-BPA module implements Swifts main functionality, encoding and prediction.
The Swift-BPA module receives BGP updates from the route server. It then adds the AS-path encoding to the BGP update and sends the modified BGP update back to the route server.
The Swift-BPA then checks if the received BGP updates are enough trigger a burst. If so it starts the prediction process. At the end of the prediction the Swift-BPA sends a fast-reroute message to the route server. The fast reroute message informs the participant controllers about the predicted AS-link.

3.3 FR-handler

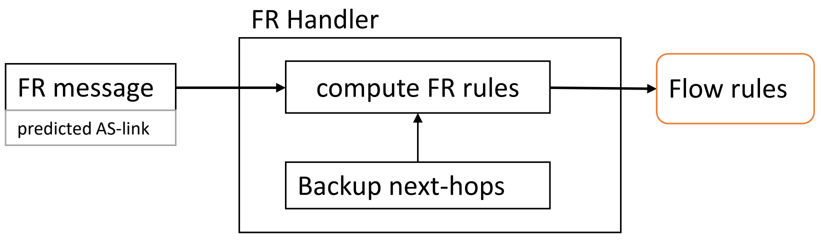


Figure 3.3: pipeline of the fast reroute handler

The FR-handler implements the handling of FR messages and pushes fast reroute rules into the SDN switch.
The FR handler receives FR messages from the route server. The FR handler then computes the fast reroute rules using the backup next-hops and the predicted AS-link. The fast reroute rules get sent to the reference monitor as flow rule messages.

3.4 Changes to the iSDX

There are a few changes to the iSDX's modules.

route server: The route server now has two modules the listener and sender.
The listener receives BGP updates and forwards them to the Swift-BPA.
The sender receives modified BGP updates and FR messages from the Swift-BPA and forwards them to the participant controllers. The sender processes FR messages with higher priority than modified BGP updates, this way the FR messages reach the FR handler as quickly as possible.

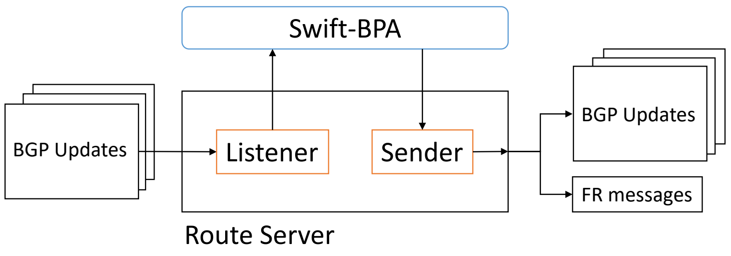


Figure 3.4: pipeline of the modified route server

local RIB: The local RIB now stores the AS-path encoding. The AS-path encoding is then used in the vmac encoding.

VMAC Encoding: The vmac encoding now computes the backup next-hops for the AS-path of the BGP update. There is one backup next-hop for every AS-link on the AS-path, packets can be sent to the backup next-hop in case this AS-link is predicted to be down.
The vmac encoding now builds the vmac using both iSDX and Swift encoding.
More in section 3.5

3.5 vmac partitioning

Both Swift and the iSDX use the destination mac address to encode information about the prefix of the packet. The destination mac address has to be shared between the iSDX and the Swift encoding. This is because the iSDX and Swift encode different information about the prefix. The iSDX encodes the participants advertising the prefix and the BGP best next-hop. Swift encodes the backup next-hops and the AS-path.
The encoded AS-path starts with the second AS on the AS-path. This is because the first AS is already encoded as the BGP best next-hop. (in the iSDX part of the vmac)

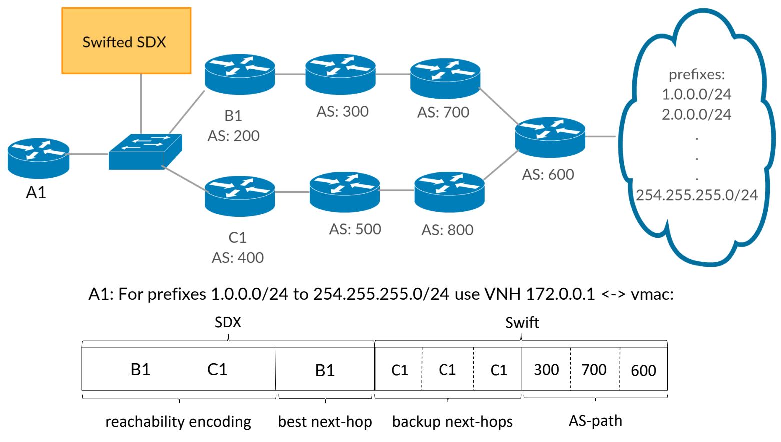


Figure 3.5: example of the vmac in the iSDX with Swift

Chapter 4

Evaluation

In this chapter I will evaluate the convergence time of the iSDX without and with Swift. I will then examine the vmac partitioning and the number of flow rules required for a fast reroute.

4.1 Test Setup

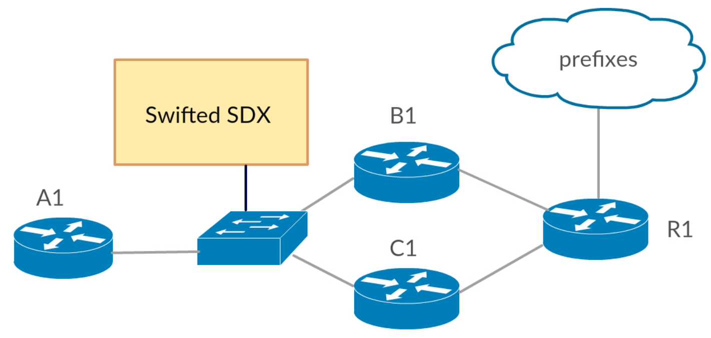


Figure 4.1: Experiment Setup

The test setup has an iSDX with or without Swift connected to three participants. Participants B1 and C1 are connected to the rest of the internet via R1 and advertise up to 500000 prefixes to A1. Participant A1 prefers routes from B1. Remote failure is simulated by setting the link B1 R1 down. If this link is down A1 needs to updates his RIB, check if flow rules have changed and update the virtual next-hop/vmac for every withdrawn prefix.
The experiment setup is run in mininet.

4.2 Convergence time without Swift

Convergence time is measured as the time between the first withdraw arriving in the route server and the participant controller finishing to process the last withdraw. To measure the convergence time the built in iSDX log server is used.
This convergence time does not take into account the hold timer or the time the participant router takes to process the withdrawals. But since these things are not under the control of the iSDX they are ignored in this evaluation.

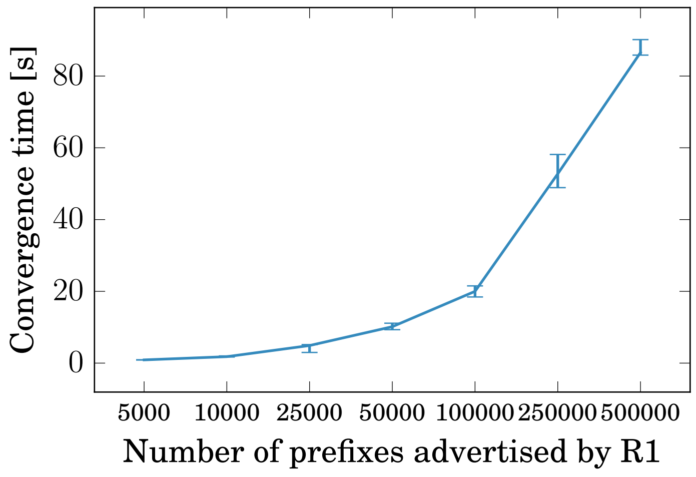


Figure 4.2: Convergence time of the iSDX without Swift

The convergence time increases linearly with the number of prefixes advertised by R1.
At 500’000 prefixes the iSDX takes about 90 seconds to converge. During these 90 seconds A1 sends packet to B1, which then get dropped by B1.

4.3 Convergence time with Swift

Convergence time is measured as the time between the first withdraw arriving in the route server and the participant controllers FR handler finishing to push the Fast-reroute rules.

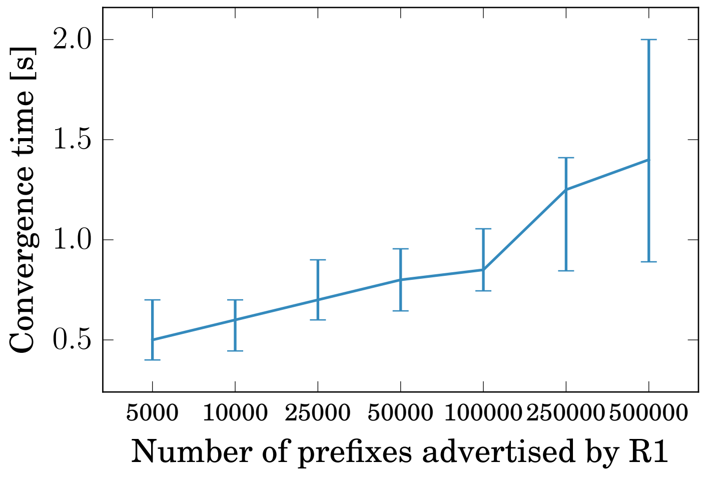


Figure 4.3: Convergence time of the iSDX without Swift

The convergence time increases slightly with higher number of prefixes. At 500’000 prefixes the iSDX takes about 1.5 seconds to push the FR rules. After these 1.5 seconds packets sent from A1 get redirected to C1 and reach their destination.
For 500’000 prefixes the convergence time is reduced by a factor of 60.

4.4 vmac evaluation

Since both the iSDX and Swift use the destination mac address to encode different information, the amount of bits available to the iSDX and Swift is reduced. In this section the vmac partitioning in its current state will be analyzed.

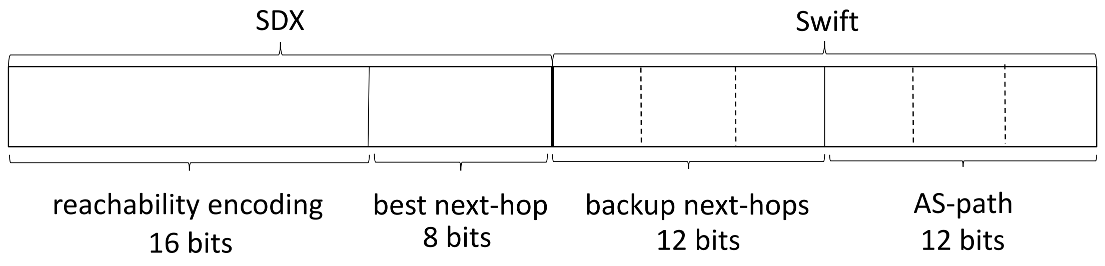


Figure 4.4: Vmac of the iSDX with Swift

In its current state 24 bits are allocated to the iSDX. 16 bits for the reachability encoding and 8 bits for the BGP best next-hop.
The amount of bits allocated for the best next-hop limits the number of participants the iSDX with Swift can have. The number of participants is limited to 2^8 = 256.
24 bits are allocated to Swift. 12 bits are used to encode backup next-hops and 12 bits are used for the AS-path encoding.
With 12 bits used for the AS-path encoding the encoding has a coverage performance of about 85%. (see Swift paper Figure 9)
12 bits for 3 backup next-hops means 4 bits for each next-hop. This means that every participant can have 16 backup next-hops at most. This is not a lot and after 16 backup next-hops have been assigned some prefixes will end up with no backup next-hop. On the other hand it also limits the number of fast reroute rules.

4.5 number of flow rules

After a fast reroute message has been received the FR handler pushes reroute rules into the IXP fabric. For every backup next-hop that the participant has stored a rule is pushed. This means the maximum number of rules pushed by a participant after a fast reroute is 16.
FR messages get sent to every participant so the maximum number of flow rules for all the participants is 256*16 = 4096. This amount of flow rules is not substantial enough to have a significant impact on the iSDX. With the number of participants limited to 256 and participants having a reasonbale number policies, the flow rule limit for current SDN switches should not be reached. (iSDX paper figure 3 (a) amsix paper figure 9)

Chapter 5

Conclusion

In this project Swift was implemented into the iSDX without significantly changing either iSDX or Swift.
The convergence time of the iSDX was significantly reduced without without too many additional flow rules. Their similar architecture makes integrating one into the other easy but it also severely constrains the iSDX's ability to scale with a higher number of participants. This is due to the bottleneck created by the limited size of the destination mac address. With the current design up to 256 participants can be supported. At the point of this work only 1.8% of all IXP's have more than 256 participants. (www.pch.net/ixp/dir)
If the swifted iSDX should be deployed at an IXP with more participants, more bits need to be allocated to the iSDX part of the vmac. This in turn impacts the performance of Swift, leading to traffic being unnecessarily redirected or failed links not being correctly detected. One might look into implementing a more lightweight fast reroute framework that does not need to encode information on the destination mac address.

Appendix A

Title

A.1 Section 1

```
All is presented exactely the way you write it.

Page boundaries are not checked.......................................................
```

Appendix B

Title

Appendix C

Title

Here may come your thesis schedule (the original plan and ev. the actual outcome).

Appendix D

Title

Here comes the original task description that you agreed on with your supervisor/tutor.