ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

TIK 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 (BGP) is the glue that holds the Internet together. It allows 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 we 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.
In the following, We will first provide some background on BGP, iSDX and Swift in 2. Then we describe the motivation behind implementing Swift in the iSDX in 4. In 5 we evaluate the system both in terms of convergence time and introduced overhead and discuss the resutlts in 6.

Chapter 2

Background

In this chapter, we give a brief overview of BGP in 2.1 and explain the problem of long convergence time upon remote failure in BGP. We highlight the architecture and key insights of both iSDX and Swift in 2.2 and 2.3, respectively. The architecture of both frameworks are very similar, they are based on a SDN switch, an SDN controller and a route server. In the next chapter, we explain how these similarities can be put to use in the implementation.

2.1 BGP

The Border Gateway Protocol allows autonomous systems to exchange routing information with each other. The routing information is communicated on a per prefix basis using BGP updates. BGP updates contain the prefix and the AS-path the packet will traverse, if packets are sent via this route. There are two types of updates, announcements and withdraws. Announcements inform that the prefix can be reached via this route and withdraws inform that the prefix can not be reached anymore via this route.
Upon receiving a BGP update routers compute the best route to the prefix of the update and then send updates to their peers. Routers only send announcements with their own best route to the prefix and if the routers do not know a route to the prefix they send withdraws.
Convergence time in BGP upon remote failure is slow. This is because of the way updates propagate trough the network. Changes are only passed on once routers have finished computing the best route to the prefix. Only once all the changes have been received, can routers make sure packets do no get sent into a black hole or loop anymore.

2.2 iSDX

The iSDX is an internet exchange point enhanced with a SDN switch and SDN controller.
An internet exchange point is a physical location where multiple autonomous systems meet to exchange traffic and BGP routes. In a traditional exchange point participants can only use a single route per prefix even though they often have multiple routes available. The iSDX gives the participants additional more fine-grained control over routing decisions. Participants can define policies to make use of this.
In this section we will first show the iSDX architecture in 2.2.1, explain the policies in 2.2.2 and explain how the next-hop and destination mac address is used in 2.2.3.

2.2.1 iSDX Architecture

The iSDX architecture has two main Parts. The Central Services and the Participant Controller. Central Services forwards BGP updates and ARP queries to the corresponding participant controller. It also initializes all the static flow rules.
Every participant has its own participant controller. The participant controller receives and processes BGP updates from the Route Server. Every participant controller has a local routing

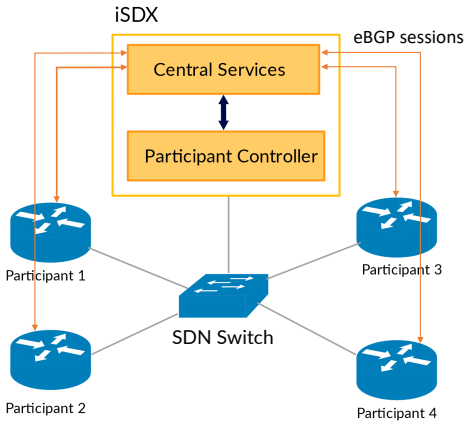

Figure 2.1: iSDX architecture

information base keeping track of all the received routes, currently used routes and routes advertised to other participants.It takes care of the Virtual next-hop and VMAC assignment. It also updates policy flow rules and manages ARP requests and gratuitous ARP. Before BGP updates get sent to the participants border router they are processed by the participant controller.

2.2.2 Policies

Policies match on a field of the packet header and then forward the packet to a participant. Policies are implemented as flow rules that the participants controller programs into the SDN switch.

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. An expamle of outbound policies is shown in Figure 2.2, where participant A has defined two outbound policies directing traffic to either one of the other two participants.

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. An example of inbound policies is shown in Figure 2.2, where participant C has defined two inbound policies directing traffic to either one of it's routers.

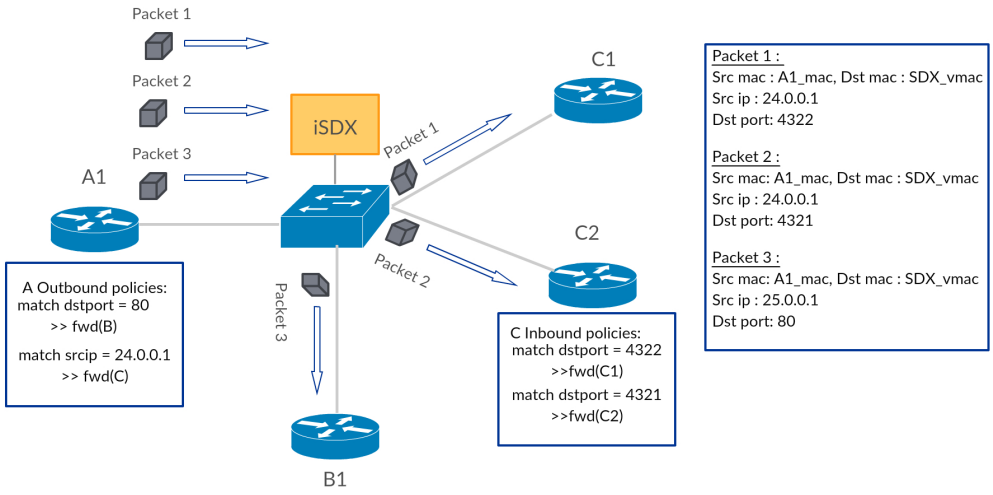

Figure 2.2: Example of outbound and inbound policies with an iSDX connected to three participants

2.2.3 Virtual Next-Hop, Virtual MAC Address

Participants are not restricted when defining outbound policies. Some policies might direct packets to participants that did not advertise the prefix or simply do not know a route to the prefix. This is a problem because outbound policies can end up violating BGP advertisements. The iSDX solves this problem by attaching additional information to each packet. This additional information is embedded into the destination mac address, transforming the destination mac addresses of packets traversing the SDN switch into a Virtual Mac Address.
In the Virtual Mac Address the participant controller encodes the participants advertising the prefix of the packet and the BGP best next hop participant for this prefix. The first part is used every time a outbound policy is applied. The outbound policy checks if the participant the outbound policy is sending packets to has advertised the prefix. The second part is used if the packet does not match any outbound policy. Default rules in the SDN switch match on the best next-hop participant and send the packet to this participant.
The Virtual Mac Address corresponds to a Virtual Next Hop, which is assigned to every prefix. The participant controller sends BGP updates to it's border routers, 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.
Figure 2.3 shows the Virtual Next-Hops and Virtual MAC Addresses used by participant A.

iSDX VMAC:
(RB: replace with a figure like the one later in the report)

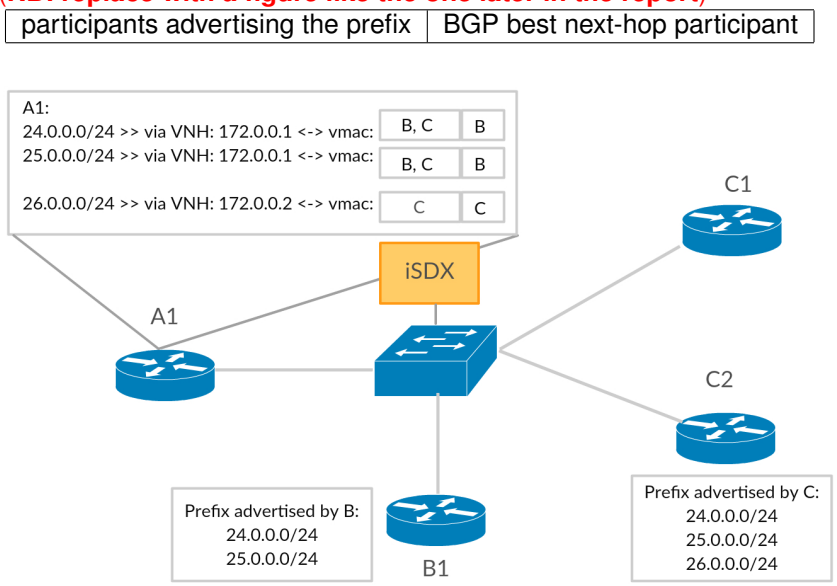

Figure 2.3: Example of the iSDX vmac with an iSDX connected to three participants

2.3 Swift

(RB: make sure that the order of the iSDX and Swift text is aligned)

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 predicted before receiving all the withdrawals.
In this section we will show the Swift architecture in 2.3.1, explain the encoding in 2.3.2 and how Swift reduces the convergence time of the swifted router in 2.3.3.

2.3.1 Architecture

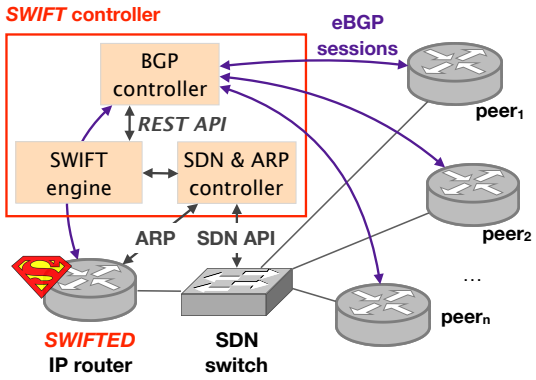

Figure 2.4: Example topology of a swifted router

Swift uses a SDN switch connected to the swifted router, its neighbors and to the Swift controller. The Swift Controller has three main parts, the BGP controller, Swift engine and the SDN & ARP controller. The BGP controller receives BGP updates from the peers of the swifted router. The BGP controller forwards the updates to the Swift engine. The Swift engine has two main modules: the burst prediction algorithm and the encoding of routing information. These two modules are the two main features of Swift. The SDN & ARP controller programs flow rules into the SDN switch and manages ARP requests.

2.3.2 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. Backup next-hops are neigbhors of the swifted router which also advertise the prefix and their advertised route does not traverse the specific AS-link. 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.
Figure 2.3 shows the swifted router and the VMACs used for the prefixes advertised by the neigbhors.

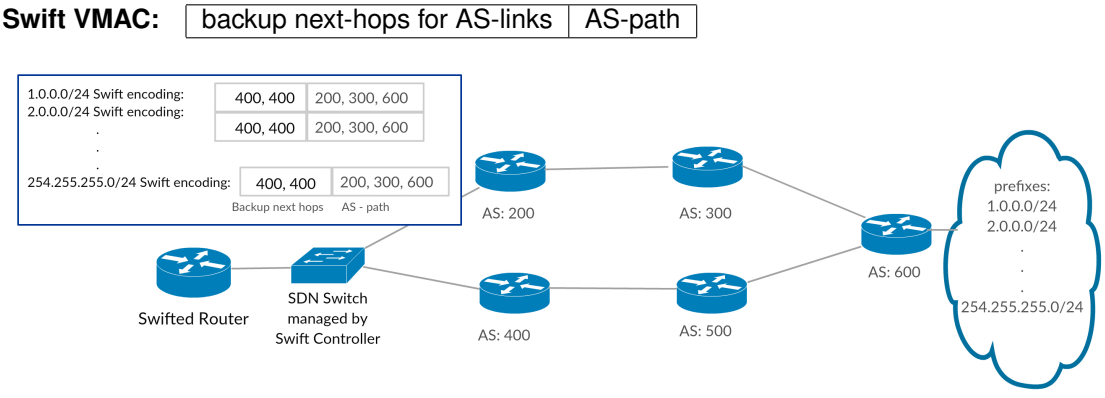

Figure 2.5: Example of the Swift vmac

2.3.3 Burst Prediction Algorithm

(RB: refer to the figure)

The burst prediction algorithm takes BGP updates. It uses the updates to build an 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 Fast Reroute flow rules into the SDN switch matching on the failed AS-link and on the corresponding backup next-hop. In Figure 2.3 the burst prediciton module has predicted the AS-link 300 600 to be down. So it pushes rules matching on the AS-path 300 600 and the backup next-hop in this case 400.
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.

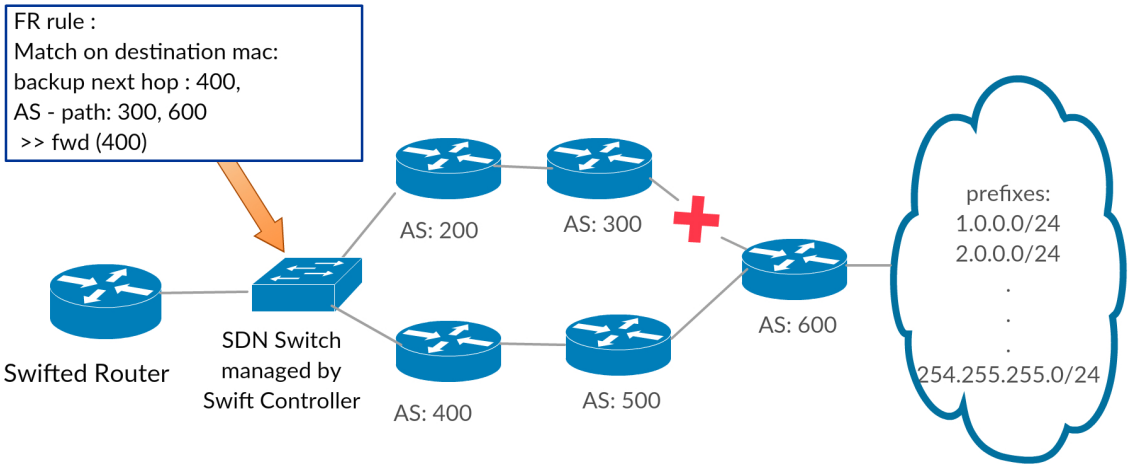

Figure 2.6: Example of a fast-reroute after BPA predicts the AS link 300 600 to be down

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.

Chapter 3

Motivation

All routers using BGP suffer from long convergence times after remote failure. This is mainly due to the propagation delay of the updates. At an iSDX many BGP routers are present and the propagation delay is made even worse because of the additional computation the participant controller needs to do to process a BGP update. Only once the participant controller has finished processing the update can the update be sent to the border router of its participant.
When receiving a BGP update the iSDXs route server first forwards the update to the participant controllers. The 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 time until the participants border router receives the BGP update. Which in turn impacts the convergence time of the routers connected to 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 idea being that Swift can push fast-reroute flow rules into the SDN switch and redirect packets to a backup participants.
The main reason why implementing Swift in the iSDX is reasonable is their architecture. The architecture of the iSDX is very similar to what Swift requires. 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.
The main challenge when implementing Swift into iSDX is to change as little as possible in both systems. Utilising their similar architecture to keep the full functionality of both iSDX and Swift. At the same time also making sure that Swift does not add too much overhead to the processing of a BGP update.
In the next chapter we explain how Swift was implemented in the iSDX.

Chapter 4

Implementation

In this chapter we show how Swift was implemented into iSDX. We give an overview of the modified iSDX architecture in 4.1. We explain the two new modules that were added to the iSDX in 4.2 and in 4.3 respectively. We explain what was changed in the default iSDX modules in 4.4 and explain the VMAC partitioning in 4.5
In the next chapter we present the results of the tests done to measure the convergence performance of the iSDX.

4.1 Architecture

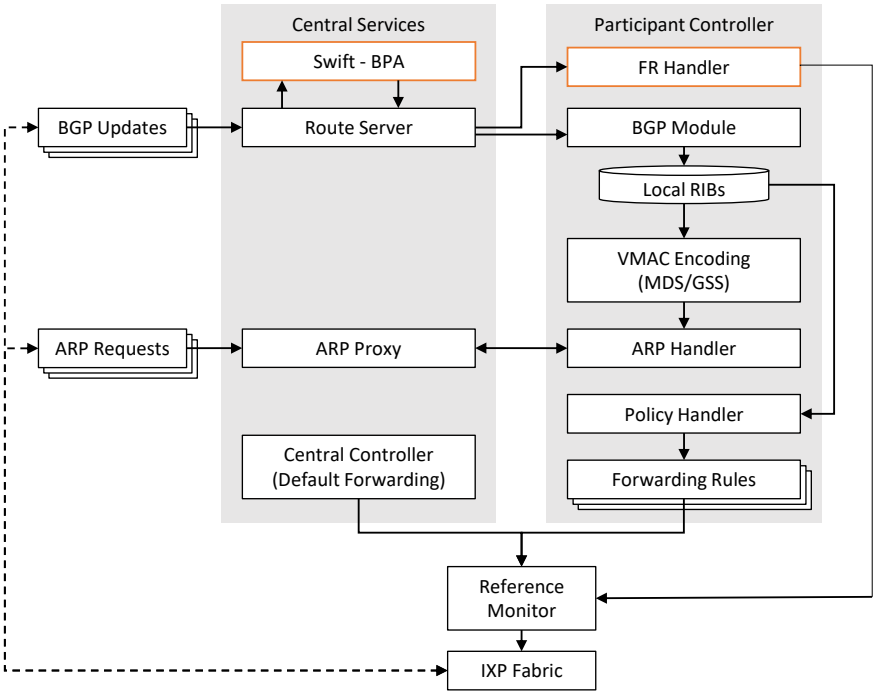

Figure 4.1: iSDX architecture with Swift

Figure 4.1 shows the iSDX [4] architecture with Swift. The orange modules represent the new modules we had to add 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.

4.2 Swift-BPA

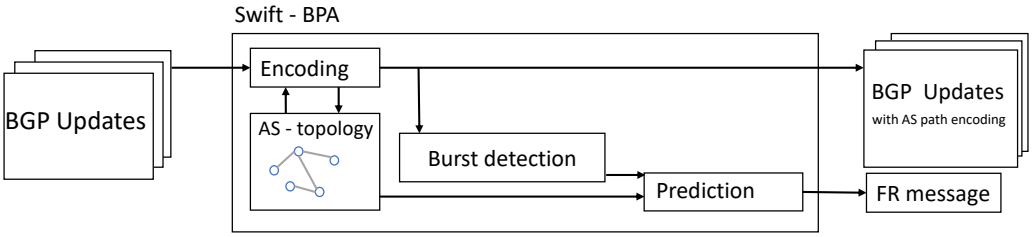

Figure 4.2: pipeline of the Swift-BPA module

The Swift-BPA module implements Swifts main functionality, encoding and prediction. It is part of the central services and exchanges BGP updates and FR messages with the route server.
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 to 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 AS-link that is predicted to be down. The route server forwards the FR message to the participant controllers.
Similiar to the participant controller every participant has a Swift-BPA process running. The Swift-BPA only receives BGP updates from his own routers. This way the Swift engine can be implemented without any modifications.

4.3 FR-Handler

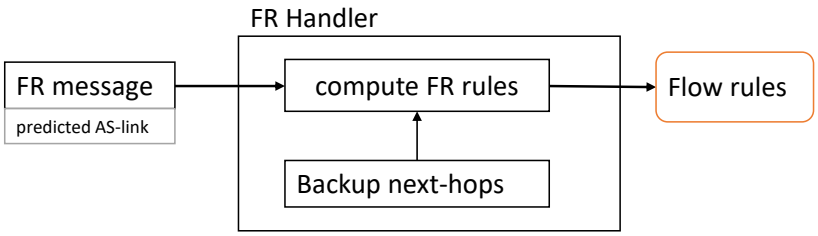

Figure 4.3: pipeline of the fast reroute handler

The FR-handler implements the pushing of fast reroute rules.
Once the Swift-BPA predicts a failed AS-link it sends FR messages to all the participant controllers. The FR handler receives FR messages and computes the fast reroute rules using the backup next-hops and the predicted AS-link. Every participant controller computes their own backup next-hops.
The fast reroute rules get sent to the reference monitor as flow rule messages. The reference monitor then programs the flow rules into the SDN switch. Just like in Swift the fast reroute rules match on the failed AS-link and on the backup next-hop.

4.4 Changes to the iSDX

There are a few changes to the iSDX's modules, mainly the route server, the local RIB of the participant controller and the VMAC encoding.

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 handlers as quickly as possible.

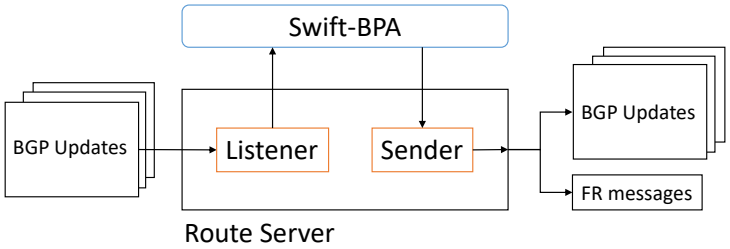

Figure 4.4: pipeline of the modified route server

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 4.5

4.5 VMAC Partitioning

Both Swift and the iSDX use the destination mac address as a VMAC to encode information about the prefix of the packet. The VMAC 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 AS-path and the backup next-hops for each link on 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)

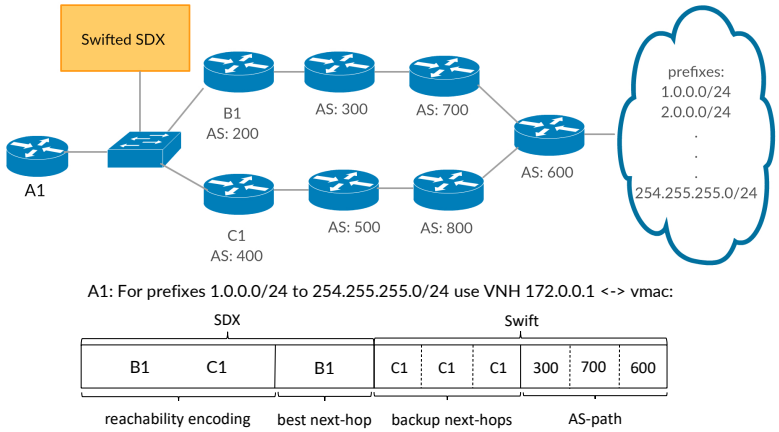

Figure 4.5: example of the vmac in the iSDX with Swift

Chapter 5

Evaluation

In this chapter we present the results of the tests we ran to evaluate the convergence performance of the iSDX. We explain the test setup in 5.1. We present the convergence time of iSDX without Swift in 5.2. We present the convergence performance of the iSDX with Swift in 5.3 and examine the Swift overhead in 5.4
In the next chapter we will discuss the results, examine the VMAC partitioning and the fast-reroute rules.

5.1 Test Setup

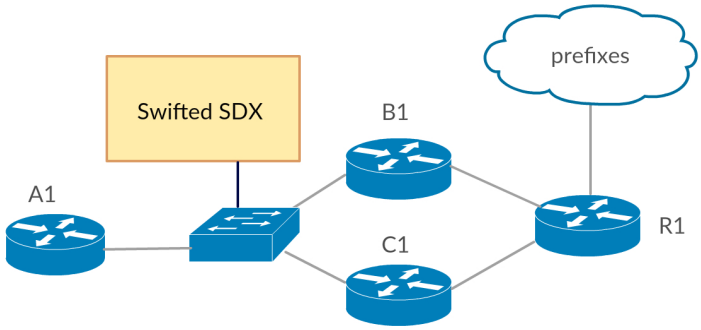

Figure 5.1: Test Setup

Figure 5.1 shows the test 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 on a server with following specs: 1x4 Cores Xeon CPU E5620 @2.4 GHz, 36G RAM, linux version 4.4.0-38-generic. The experiment is run in Mininet [2]. The routers A1, B1, C1 and R1 are quagga [3] routers. The perl script bgpsimple [1]is used to inject an arbitrary number of routes into R1.

5.2 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 we use the built in iSDX log server.
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.

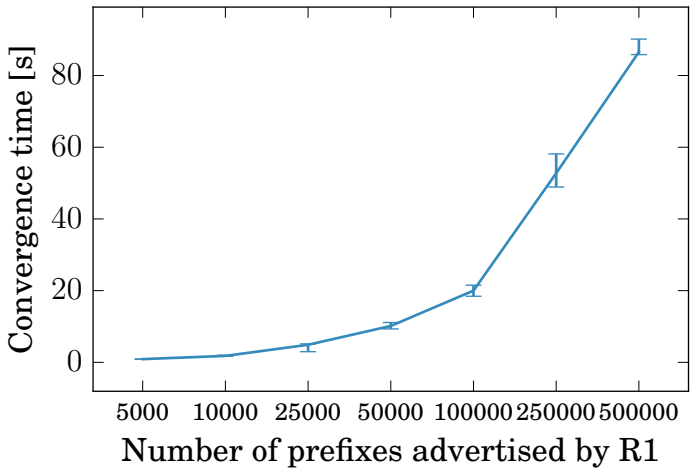

Figure 5.2: Convergence time of the iSDX without Swift

Figure 5.2 shows the convergence time depending on the number of prefixes injected by bgp-simple. 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.

5.3 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.

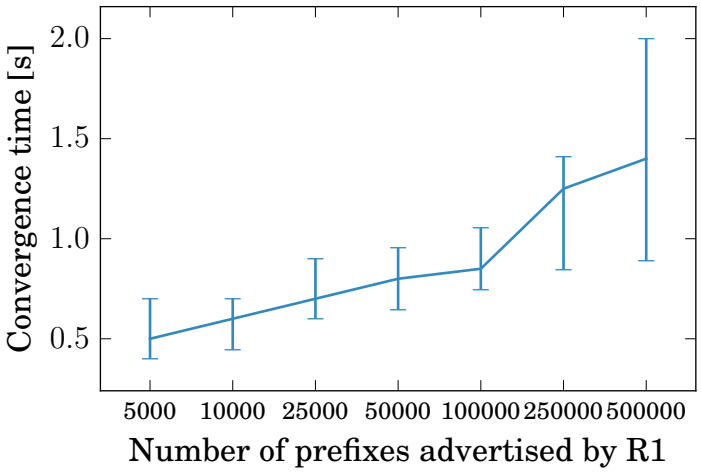

Figure 5.3: Convergence time of the iSDX without Swift

Figure 5.3 shows the convergence time depending on the number of prefixes injected by bgpsimple. 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.

5.4 Swift overhead

Chapter 6

Discussion

In this chapter we will discuss the convergence times of iSDX with and without Swift in 6.1. We will evaluate the VMAC partitioning in 6.2 and examine the fast reroute flow rules in 6.3

6.1 Convergence time

(**RB: discuss results and overhead and swift performance**)

6.2 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. The current VMAC partitioning is simply the first intuition on how the VMAC can be partitioned. The partitioning can be easily changed by configuring the iSDXs parameters. There may still be some optimization. But we realize that optimizing the VMAC partitioning would outgrow the time frame of this project. In this section we evaluate the VMAC partitioning in its current state.

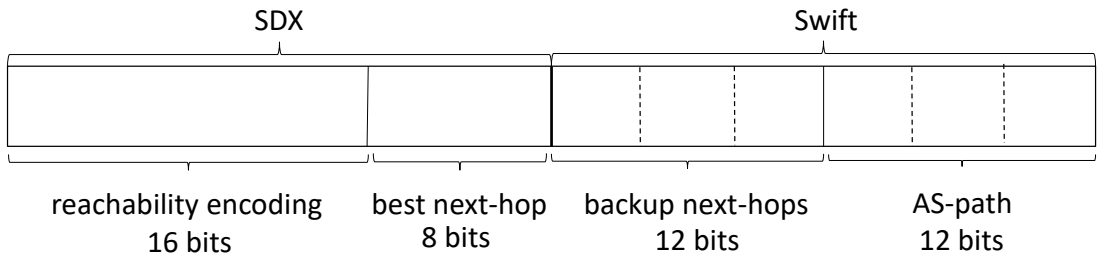

Figure 6.1: Vmac of the iSDX with Swift

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.

6.3 Fast Reroute Flow Rules

In this section we will first examine the number of Fast reroute Flow Rules required for a Fast-Reroute in 6.3.1. We will also examine the priority of the Fast-Reroute Flow Rules compared to Outbound Policies in 6.3.2

6.3.1 Number of Flow Rules

After a Fast Reroute message has been received the FR handler pushes Fast 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 that is peering with the participant whose Swift-BPA triggered the Fast Reroute. This means the maximum number of flow rules pushed after a Fast Reroute is triggered 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)
Usually this upper bound of flow rules should not be reached. Upon a Fast-Reroute not all participants will be peering with the participant that triggered the Fast Reroute. Also the participants may not have reached the maximum number of Flow Rules yet.

6.3.2 Outbound Policies and Fast Reroute Rules

Fast Reroute rules and Outbound Policies are in conflict.
If Outbound policies have a higher priority than Fast Reroute rules, packets that match the policy may be rerouted to a backup next-hop or they may be rerouted to a participant whose BGP route traverses the failed AS-link.
If Fast Reroute rules have a higher priority then the participants Outbound policies will be ignored.
Due to the limited number of bits available, encoding the AS-path and backup next-hops of the participants routes in the reachability encoding is not feasible. In the current iSDX with Swift Fast Reroute rules have a higher priority than Outbound policies. Future work may allow participants to define which Outbound policies they want overridden by Swift and which ones not. But this goes beyond the scope of this project.

Chapter 7

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 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.

Bibliography

[1] Bgpsimple simple bgp peering and route injection script. https://github.com/xdel/bgpsimple. Accessed: 2016-11-02.

[2] Mininet an instant virtual network on your laptop. http://mininet.org/. Accessed: 2017-01-06.

[3] Quagga routing software suite. http://www.nongnu.org/quagga/. Accessed: 2016-10-28.

[4] N. Feamster, J. Rexford, S. Shenker, R. Clark, R. Hutchins, D. Levin, and J. Bailey. SDX: A Software Defined Internet Exchange. *Open Networking Summit*, page 1, 2013.