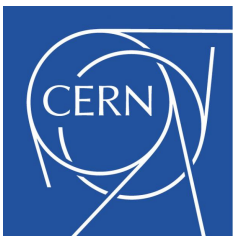
Evaluation of an IP Fabric network architecture for CERN's data center

Carles Garcia Cabot
Bachelor's Degree in Informatics Engineering – Information Technologies



European Organization for Nuclear Research (CERN)
Supervised by Carles Kishimoto Bisbe

Department of Computer Architecture
Facultat d'Informàtica de Barcelona (FIB)
Universitat Politècnica de Catalunya (UPC) – BarcelonaTech
Supervised by José M. Barceló Ordinas

25/04/2018

CERN-THESIS-2018-035
25/04/2018

Abstract

CERN has a large-scale data center with over 11 500 servers used to analyze massive amounts of data acquired from the physics experiments and to provide IT services to workers. Its current network architecture is based on the classic three-tier design and it uses both IPv4 and IPv6. Between the access and aggregation layers the traffic is switched in Layer 2, while between aggregation and core it is routed using dual-stack OSPF. A new architecture is needed to increase redundancy and to provide virtual machine mobility and traffic isolation.

The state-of-the-art architecture IP Fabric with EVPN is evaluated as a possible solution. The evaluation comprises a study of different features and options, including BGP table scalability and autonomous system number distributions. The proposed solution contains eBGP as the routing protocol, a route control policy, fast convergence mechanisms and an EVPN overlay with iBGP routing and VXLAN encapsulation. The solution is tested in the lab with the network equipment currently used in the data center. The results are satisfactory, however the equipment is found to lack the necessary resources to implement this architecture at a large scale.

Resumen

CERN dispone de un centro de datos de grande escala con más de 11 500 servidores usados para analizar masivas cantidades de datos adquiridos a partir de los experimentos y para proveer servicios IT a los trabajadores. La arquitectura de red actual está basada en el diseño 3-tier. Entre las capas de acceso y agregación el tráfico es conmutado en capa 2, mientras que entre agregación y núcleo está enrutado usando OSPF dual-stack. Una nueva arquitectura es necesaria para incrementar la redundancia y para proveer movilidad de máquinas virtuales y aislamiento de tráfico.

La arquitectura IP Fabric con EVPN es evaluada como una posible solución. La evaluación consiste de un estudio de diferentes características y opciones, incluyendo la escalabilidad de la tabla BGP y distribuciones de números de sistema autónomo. La solución propuesta contiene eBGP como el protocolo de routing, una política de control de rutas, mecanismos de convergencia rápida y un overlay EVPN con routing iBGP y encapsulación VXLAN. La solución es testeada en el laboratorio con el equipamiento de redes actualmente usado en el centro de datos. Los resultados son satisfactorios, sin embargo se ha detectado que el equipamiento no tiene los recursos necesarios para implementar esta arquitectura a gran escala.

Resum

CERN disposa d'un centre de dades de gran escala amb més de 11 500 servidors utilitzats per analitzar massives quantitats de dades adquirides a partir dels experiments i per proveir serveis IT als treballadors. L'arquitectura de xarxa actual està basada en el disseny 3-tier. Entre les capes d'accés i d'agregació el tràfic és commutat en capa 2, mentre que entre agregació i nucli està enrutat utilitzant OSPF dual-stack. Una nova arquitectura és necessària per incrementar la redundància i per proveir mobilitat de màquines virtuals i aïllament de tràfic.

L'arquitectura IP Fabric amb EVPN és evaluada com una possible solució. L'avaluació consisteix d'un estudi de diferents característiques i opcions, incloent l'escalabilitat de la taula BGP i distribucions de números de sistema autònom. La solució proposada conté eBGP com el protocol de routing, una política de control de rutes, mecanismes de convergència ràpida i un overlay EVPN amb routing iBGP i encapsulació VXLAN. La solució és testejada en el laboratori amb l'equipament de xarxes actualment utilitzat en el centre de dades. Els resultats son satisfactoris, tot i això s'ha detectat que l'equipament no té els recursos necessaris per implementar aquesta arquitectura a gran escala.

Acknowledgements

I would like to thank Carles Kishimoto and the whole CE section for their support and providing the opportunity to work and make an impact at CERN. I would also like to thank José M. Barceló for his help and valuable advice.

Contents

List of Figures

List of Tables

Acronyms

ARP Address Resolution Protocol

ASN Autonomous System Number

BFD Bidirectional Forwarding Detection

BGP Border Gateway Protocol

CERN European Organization for Nuclear Research

DC Data Center

ECMP Equal Cost Multi Path

EGP Exterior Gateway Protocol

EVPN Ethernet Virtual Private Network

IGP Interior Gateway Protocol

LACP Link Aggregation Control Protocol

LAG Link Aggregation Group

LHC Large Hadron Collider

MAC Media Access Control

MPLS MultiProtocol Label Switching

MRAI Minimum Route Advertisement Interval

OSPF Open Shortest Path First

P2P Point to Point

RFC Request For Comments

RFD Route Flap Damping

SS Super Spine

STP Spanning Tree Protocol

TCP Transport Control Protocol

UDP User Datagram Protocol

VGA Virtual Gateway Address

VLAN Virtual Local Area Network

VM Virtual Machine

VNI VXLAN Network Identifier

VPLS Virtual Private LAN Service

VTEP Virtual Tunnel End Point

VXLAN Virtual eXtensible Local Area Network

Part I

The project

1 Introduction

1.1 CERN

The European Organization for Nuclear Research (CERN) is the largest particle physics laboratory in the world. Its goal is to study the basic constituents of matter – the fundamental particles. The particles are made to collide together at close to the speed of light using the world's largest and most complex scientific instruments. The process gives the physicists clues about how the particles interact, and provides insights into the fundamental laws of nature.

Founded in 1954, the laboratory is located near the Franco-Swiss border in Geneva. It was one of Europe's first joint ventures and now has 22 member states. There are over 2250 staff members employed by CERN but there can be up to 13 000 people on site at any one time. This includes users, students, fellows, sub-contractors and visiting scientists from around the world.

The instruments used at CERN are purpose-built particle accelerators and detectors. Accelerators boost beams of particles to high energies before the beams are made to collide with each other or with stationary targets. Detectors observe and record the results of these collisions. The Large Hadron Collider (LHC) is the world's largest and most powerful particle accelerator and it first started up on September 2008.

The LHC consists of a 27-kilometer ring of superconducting magnets located 100 meters underground. The particles are made to collide approximately 1 billion times per second at close to the speed of light, generating about one petabyte of data per second. However, such quantities of data are impossible for current computing systems to record and they are hence filtered by the experiments, keeping only the most interesting ones. The filtered data are then aggregated in the CERN Data Center (DC), where initial data reconstruction is performed, and where a copy is archived to long-term tape storage.

Even after the drastic data reduction performed by the experiments, the DC processes on average one petabyte of data per day [1]. Therefore, the DC requires a high-bandwidth network with a carefully design architecture.

1.2 The data center

The DC is the heart of CERN's entire scientific, administrative and computing infrastructure. All services, including email, scientific data management and videoconferencing use equipment based in the DC. It houses over 11 500 servers (174 000 processor cores and 61 900 disks) and 230PB of permanent storage in tapes [2][3]. A high-performance Ethernet

network comprising of over 400 switches is deployed to interconnect all these machines. The total power requirement is 3.5MW.

Today's DC network is based on the classic 3-Tier design model which consists of the core, aggregation and access layers. Between the core and aggregation layers the traffic is routed using Open Shortest Path First (OSPF), while between the access and aggregation layers it is switched in Layer 2. All access switches are connected to the same aggregation router with one or multiple uplinks.

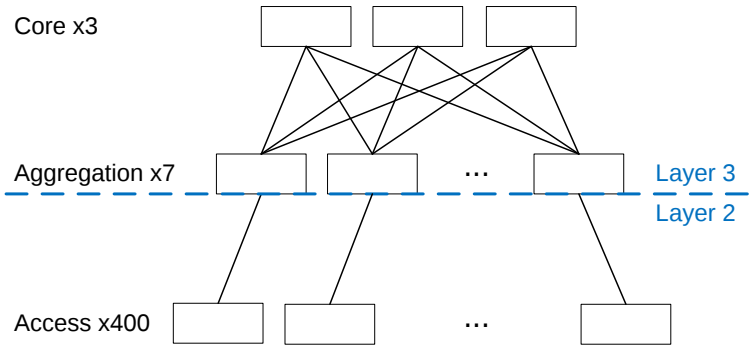


Figure 1.1: Schema of the current architecture

The fact that each switch is connected to a single router has not been a problem thanks to the high reliability of the current hardware. It only becomes a problem when there is a need to do maintenance to an aggregation router (e.g. a firmware upgrade), which results in a 5 minute disconnection of all 50 access switches connected to it, and therefore hundreds of servers.

In addition, introducing a new manufacturer with less reliable hardware could have a big impact in the DC, making network redundancy an important factor to improve.

1.3 Context

I joined CERN in February 2017 as a technical student in the Information Technology (IT) department. As a member of the Communication Engineering section (CE) of the Communication Systems group (CS), I've been working on improving the DC network. The topic of this thesis, *Evaluation of an IP Fabric network architecture for CERN's data center*, has been my main project. I've done most of the research and development and have received support and contributions from the engineers in the section.

The IT department provides the computer services required for the fulfillment of CERN's mission in an efficient and effective manner. This includes data processing, storage, networks and support for the whole laboratory and its users. It also provides a ground for advanced research and development of new IT technologies with partners from other research institutions and industry [4].

The Communication Systems group is responsible for all telecommunication services. These include the campus network for office and WiFi connectivity, a technical network to support accelerator operations, a high-performance data center network to support physics computing, external high-bandwidth connections to computing facilities around the world, fixed and mobile telephony services, and radio services for the emergency teams.

The Communication Engineering section consists of 12 network and computer engineers responsible for provisioning network equipment, operating services such as WiFi, DNS and DHCP, and designing and configuring the network architectures.

2 Background

2.1 Layer 2 architectures

The most common classic architecture is the 3-tier switched design which consists of 3 layers (or stages): access, aggregation (or distribution) and core. Each higher layer, from the server towards the WAN, has higher port density and bandwidth capacity where the core functions as the ”trunk” of the tree-based design.

Traditionally, Spanning Tree Protocol (STP) is used to achieve a redundant loop-free topology. However, it doesn't scale well in large broadcast domains and the redundancy it provides is passive, which means the backup links are only used when the main fails. Although the protocol has been enhanced over the years, the convergence and stability is still not enough.

Recently, new Layer 2 protocols that provide active redundancy have appeared with the goal of replacing STP. These are TRILL and related proprietary protocols. However, they have limited number of implementations and equipment that supports it, so its applicability is limited and the cost of such designs is expensive. [5].

An alternative to the L2 design is a hybrid L2/L3, like CERN currently has. This means that traffic between the aggregation and the core layers is routed in L3 with OSPF. This was done in order to have active redundancy, therefore increasing the core bandwidth and avoiding wasting expensive high-bandwidth transceivers and ports.

The simplest complete solution to the redundancy problem is to extend Layer 3 down to the access layer, thus converting all switches to routers. This was not possible until recently, when ToR switches that support routing have become more common and cheaper.

2.2 State-of-the-art

IP Fabric is the current state-of-the-art network architecture for medium and large-scale data centers, as proven by the implementation of tech giants (e.g. Microsoft [6], Facebook [7]) and the solutions offered by most of the largest communications hardware companies (e.g. Cisco [8], Juniper [9], Brocade [10]). However, they rarely explain the inner workings or discuss design decisions, and sometimes offer a vendor-agnostic solution that doesn't fit our needs. The network engineering group in the organization considers that a proper evaluation is needed in order to understand better the technology and optimize it for our needs and use cases. The best existing resources for the state-of-the-art are:

- RFC7938 - Use of BGP for Routing in Large-Scale Data Centers [5]: it's quite detailed but the physical architecture presented is different to what CERN's currently has. Makes many assertions without providing proof or comparing solutions.

- Brocade Network Virtualization in IP Fabric with BGP EVPN [11]: not very detailed but provides an example configuration for their devices, albeit incomplete.

The IP Fabric architecture, also known as Leaf-Spine, typically consists in a 3-stage Clos network [12] where all ports are Layer 3. The access switches (Leaves) act as routers and are connected to all aggregation routers (Spines). The Leaves then load balance the traffic among all available links using Equal Cost Multi Path (ECMP) routing. Compared to the current situation, a failing Spine would have no impact but reduced capacity.

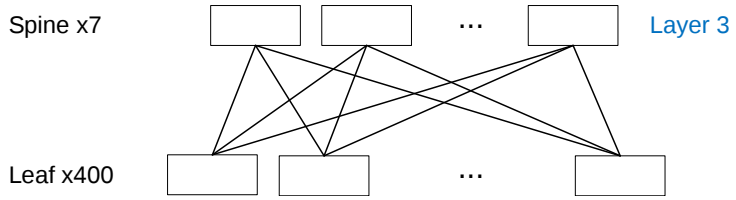


Figure 2.1: 3-stage IP Fabric schema

Connecting 400 Leaves to all Spines is not viable since Spine routers would need a huge density and the cost of thousands of links would be very high. As a result, the proposed architecture will be a 5-stage Clos network, introducing the Super Spine (SS) routers.

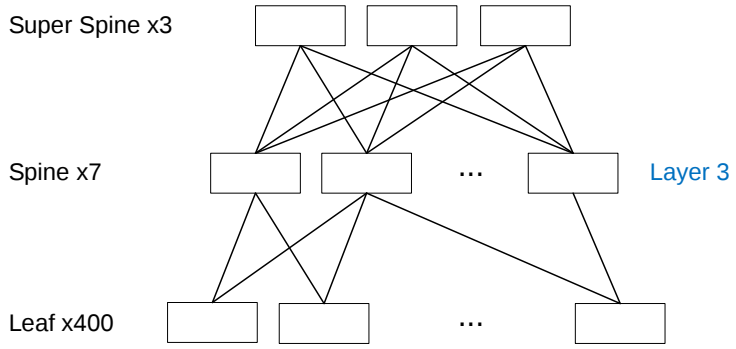


Figure 2.2: 5-stage IP Fabric schema

The network looks similar to the current. It will be necessary to add at least an uplink to each Leaf towards a different Spine for redundancy and to enable the routing protocol.

In addition, there could be a separation of storage and computing servers by creating clusters (or pods). A cluster is a set of Leaves connected to the same Spines. Having a cluster for Virtual Machines (VM) would allow to deploy new technologies like Ethernet Virtual Private Network (EVPN) for VM mobility without impacting the whole data center.

To sum up, the main benefits of an IP Fabric are:

- Increased redundancy, minimizing the impact of hardware failure and maintenance stops.
- Traffic isolation between computing domains and services.
- Ability to deploy EVPN and provide VM mobility

Drawbacks:

- Economical: The cost of the switches as they will need to support routing, and more ports and cables will be used.

2.3 Technologies

This project will require the knowledge of different networking protocols, the main described below:

- Border Gateway Protocol (BGP): an Exterior Gateway protocol (EGP) that is used to exchange routing information among routers in different Autonomous Systems (AS). It's a path vector routing protocol and has longer convergence time than link-state protocols but excellent scalability. The routing information includes the complete route to each destination. The protocol uses the network reachability information to construct a graph of AS connectivity, which enables to remove routing loops and enforce policy decisions at the AS level [13]. Although BGP is known as the de facto interdomain routing protocol in the Internet, it has been recently introduced in the data center.
- Open Shortest Path First (OSPF): an Interior Gateway Protocol (IGP) commonly used in large enterprise networks. OSPF is a link-state routing protocol providing fast convergence and good scalability. Each router maintains a database describing the Autonomous System's topology [14].
- Equal Cost Multi Path routing (ECMP): it is a balancing method of routing when the routing table contains multiple next-hop addresses for the same destination with equal cost. If there is an ECMP set for the active route, a hash algorithm is used to choose one of the next-hop addresses in the set to install in the forwarding table [15].
- Link Aggregation Control Protocol (LACP): enables aggregating multiple links between Ethernet physical interfaces to create a single logical link called Link Aggregation Group (LAG). The LAG balances traffic across the member links and effectively increases the bandwidth. Another advantage of link aggregation is increased availability, because the LAG is composed of multiple member links. If one member fails, the LAG continues to carry traffic over the remaining links [16].
- Bidirectional Forwarding Detection (BFD): provides fast link failure detection for many media types and routing protocols [17].

3 Objective

The objective of this project is to evaluate a next-generation network architecture, the IP Fabric, for CERN's DC. This architecture would allow to increase network redundancy and provide traffic isolation and virtual machine mobility. These features are needed by data center administrators and users. The evaluation will comprise an in-depth analysis of different variations and their impact.

3.1 Scope

The IP Fabric will be designed based on current and future needs. Different designs will be contrasted qualitatively (networking protocols, complexity of operation, reliability...) and quantitatively (hardware resources needed, path redundancy, convergence time...). The design selected as most fitting will be tested in a lab with physical routers and tester devices. To automatize the tests, Python scripts will be written to generate the configuration of the routers. Based on theoretical and practical results, the necessary resources to implement the architecture in the data center will be estimated.

In summary, this project will require the understanding of advanced networking concepts, the knowledge of the operation of a data center and programming skills to achieve automation.

3.2 Possible obstacles

The lab tests will be carried out with 7 routers. It's possible that the results won't apply exactly to the real DC, which would have hundreds of routers. This is difficult to predict and won't be known until the architecture is deployed. The deployment should be done gradually to ensure impact is minimal in case of failure.

Related to this, another possible obstacle is that the router models used for the testing might be different than those in the DC when the architecture is deployed. This is because newer routers may be acquired next year. If that happened, the tests should be redone to ensure the deployment and design is correct.

In addition, some of the routers to be used for the tests are loaned by a vendor for CERN to test. Therefore, the tests will have to be done in a short timespan so the units can be returned by the deadline.

3.3 Stakeholders

The target audience of the project are network engineers that need an evaluation in depth of this new architecture. This document will be helpful to many institutions and companies interested in implementing these technologies. In particular, the engineers in the CE section at CERN need this research to decide if the implementation is feasible considering the available resources.

CERN has over 10 000 users and workers that use the data center daily for their research and work. They would all benefit from this upgrade thanks to the increase of network redundancy and the reduced impact of network failures.

4 Plan

4.1 Methodology

The first phase of the project will be reading all relevant and available documentation needed to learn and understand the networking concepts required. In the second phase, different designs and variations will be compared and discussed with other network engineers to decide which is best. The third phase will consist in checking that the chosen design is theoretically sound, and corrections will be made if necessary. Lastly, once the model is very clear, it will be implemented in the lab and tested in conditions as similar as possible to the real DC. If big problems or limitations are found, a different design will be tested.

All the work, including the documentation written and obtained, the router configurations and outputs, and the Python scripts will be saved on a local and remote Git repository. The material such as networking devices, cables, optical fibers and transceivers is provided by CERN.

The development will be followed closely by the engineering team, which will participate in the discussions, and the tutor, who will be informed regularly of the progress and will help to achieve the objectives.

4.2 Tasks and schedule

The estimated duration of the project is approximately 4 months. It starts the 12 September 2017 and finishes before the 22 January 2018, when it will be presented. There will be a break of 2 weeks between the 22 December and the 7 January for holidays. It must be noted that the schedule and tasks could be modified in the future if any problems or inconveniences were found.

The plan of the project can be divided in the following tasks, whose time to complete has been estimated:

Understanding current architecture design and limitations

The current architecture of CERN's data center needs to be understood, as well as its needs and limitations. Despite having previous knowledge of how a data center works, it's necessary to learn how it operates in practice and which are the peculiarities of this one. In addition, the software tools and procedures used by the engineers will have to be studied. From the start, a PC with Ubuntu 16 and Windows 10 will be used.

Project planning

Project planning is important since it is being developed in a company. The resources needed will be monitored and not used exclusively for this project, so a clear schedule and planning of the workloads needs to be estimated.

Learning advanced networking concepts

To understand network architectures for large-scale data centers, advanced concepts that are not taught in university are needed. This will be helpful also for the next task, since understanding technical documents is vital for the project. The knowledge will be acquired through public available documents, such as company manuals and technical standard specifications, and through the mentoring of peers. The networking concepts in question are:

- Routing protocols in depth: Open Shortest Path First (OSPF), Border Gateway Protocol (BGP)
- Internet Protocol version 6 (IPv6)
- Control protocols: Link Aggregation Control Protocol (LACP), Bidirectional Forwarding Detection (BFD)
- Overlay protocols: Virtual eXtensible Local Area Network (VXLAN), Ethernet Virtual Private Network (EVPN)

Reading available documentation and papers about IP Fabric

Due to the novelty of this technology, the available literature is not extensive. However, searching and understanding it will take a long time.

Evaluate IP Fabric designs

Different designs will be contrasted qualitatively (networking protocols, complexity of operation, reliability...) and quantitatively (hardware resources needed, path redundancy, convergence time...). Discussions with other network engineers will take place in order to decide which is best.

Study and adapt the selected design

This task will consist in checking that the chosen design is theoretically sound, and corrections will be made if necessary. It will be adapted to our current needs and limitations, based on what is learned in the previous research. Things that will have to be considered are:

- Are our current routers and switches compatible with the protocols and functions this design requires?
- If not, can the design still be implemented while ignoring the missing features?
- If yes, do they have the necessary hardware resources to implement the design in such scale?
- If the hardware resources are limited, which changes need to be made?
- Which hardware not currently owned by the company could make the design feasible?

Test the selected design in the lab

Once the model is very clear, it will be implemented in the lab and tested in conditions as similar as possible to the real DC. If big problems or limitations arise, it will be necessary to go back to the previous task.

In order to do this, the available hardware will be installed and configured. The necessary tools will be learned too. To automate the testing, Python scripts will be written to generate the configuration of the routers. Based on theoretical and practical results, the necessary resources to implement the architecture in the DC will be calculated.

Specifically, the resources needed for this task are: 7 routers, a tester device and a python IDE. Apart from a Network Engineer, a Software Engineer role will act.

Write documentation

After all the tests are finished and consistent and validated results are obtained, a formal document will be written. It must be noted that documenting the process will be necessary for all tasks.

To write the documentation, LaTeX and Microsoft Visio will be used.

Task	Estimated time (hours)
Understanding current architecture	50
Project planning	25
Learning advanced networking concepts	35
Reading documentation about IP Fabric	50
Evaluate IP Fabric designs	100
Study and adapt the selected design	120
Test the selected design in the lab	110
Write documentation	50
Total	540

Table 4.1: Estimated time for each task

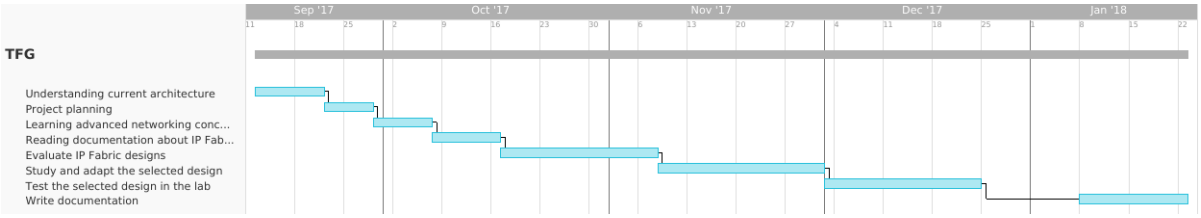


Figure 4.1: Gantt chart

4.3 Resources

Hardware

Note that the network devices below are called switches by the vendors, but could also be called routers as they have routing capabilities. In this document both terms will be used interchangeably.

- PC (Intel i7, 8GB RAM, 256GB SSD)
- 2 Brocade SLX 9540-48S switches [18]: a Top-of-Rack (ToR) 1U switch
- 4 Brocade ICX7750-48F switches [19]: a ToR 1U switch
- 1 Brocade MLXe-16 switch [20]: an aggregation or core modular chassis switch
- 1 Spirent HyperMetrics SPT-11U tester [21]: a network tester that can generate traffic and emulate routers

Figure 4.2: Brocade SLX 9540-48S

Figure 4.3: Brocade ICX7750-48F

Figure 4.4: Brocade MLXe-16

Software

This project will use Free and Open Source Software (FOSS) except when a proprietary solution is needed, like in the case of Windows 10 and Microsoft Visio, one of the best diagramming tools, which will be used to draw the networks.

- Ubuntu 16
- Windows 10
- LaTeX document preparation system
- Spyder 3 Python IDE

- Spirent TestCenter test software
- Microsoft Visio diagramming software

4.4 Action plan

The plan is to complete the tasks sequentially as specified. The writing of the documentation could be made at the same time as most tasks. As seen in the Gantt chart, all tasks except the writing of documentation should be completed before Christmas holidays.

Again, the goal of the project is to evaluate if the IP Fabric architecture is viable for CERN's data center. It is expected to find a solution that meets the requirements, but it is not guaranteed. If such solution is not found, the lab testing task would be used to compare different designs.

An unlikely but possible obstacle could be the unavailability of lab resources. This is improbable since they have been reserved, but if such case happened, the number and length of tests should be reduced.

If a switch had a problem and stopped working, the tests would be adapted to the situation. Fortunately, a single switch less wouldn't be very disruptive.

Apart from these 3 unlikely obstacles, the execution of the tasks is straightforward. The project will be developed during 19 weeks, or 17 if holidays are discounted. This means each week will require 32 hours of work at most, which is viable.

4.5 Project budget

In this section the budget is presented, taking into account the total price of the products and resources needed, and the depreciation when relevant. The total duration of the project is 118 days (from 12 September to 22 January, minus 14 days for holidays). This number will be used to calculate depreciation.

Hardware budget

An important part of the budget will be dedicated to hardware because high-end data center switches, necessary for the tests in the lab, are very expensive.

Product	Price per unit (€)	Units	Price (€)	Useful life (years)	Depreciation (€)
PC	800	1	800	5	51.73
Brocade SLX	35 000	2	70 000	5	4 526.03
Brocade ICX	15 000	4	60 000	5	3 879.45
Brocade MLX	45 000	1	45 000	5	2 909.59
Spirent	20 000	1	20 000	5	1 293.15
Total			195 800		12 659.95

Table 4.2: Hardware budget

Sources for the prices: [22], [23], [24], [25]

Software budget

Most programs used are FOSS. A license for Windows 10 and Microsoft Visio will have to be acquired.

Product	Price per unit (€)	Units	Price (€)
Ubuntu 16	0	1	0
Windows 10	200	1	200
LaTeX document preparation system	0	1	0
Spyder 3 Python IDE	0	1	0
Spirent TestCenter test software	0	1	0
Microsoft Visio diagramming software	590	1	590
Total			790

Table 4.3: Software budget

Human resources budget

This project requires a Network engineer [26] and a Software engineer [27]:

Role	Hourly wage (€)	Work hours	Salary (€)
Network engineer	40	500	20 000
Software developer	45	40	1 800
Total			21 800

Table 4.4: Human resources budget

Indirect costs

The necessary hardware requires a significant amount of power. The electricity cost in Switzerland is approximately 0.25€/kWh [28]. Source for the power consumption values: [29],[30],[31].

Product	Power (W)	Units	Total power (W)	Hours	Energy (kWh)	Cost (€)
PC	100	1	100	540	54	13.50
Brocade SLX	166	2	332	110	36.5	9.10
Brocade ICX	250	4	1000	110	110	27.50
Brocade MLX	320	1	320	110	35.2	8.80
Spirent	690	1	690	110	75.9	19.00
Total			2442		311.6	77.90

Table 4.5: Indirect costs

Unexpected costs

There aren't any unexpected costs that can be foreseen. Nonetheless, a contingency reserve equal to the 3% of the total budget will be added.

Total budget

Concept	Cost (€)
Hardware budget	12 659.95
Software budget	790.00
Human resources budget	21 800.00
Indirect costs	77.90
Subtotal	35 327.85
Contingency reserve (3%)	1 059.84
Total	36 387.69

Table 4.6: Total budget

The budget for each task is shown below. Each budget includes the relevant number of Network engineer hours paid, plus the cost of the resources specified in the last column. Indirect costs and the contingency reserve are omitted.

Task	Budget (€)	Justification
Understanding current architecture	2 251.73	PC, Windows 10
Project planning	1 000.00	
Learning advanced networking concepts	1 400.00	
Reading documentation about IP Fabric	2 000.00	
Evaluate IP Fabric designs	4 000.00	
Study and adapt the selected design	4 800.00	
Test the selected design in the lab	17 208.22	Switches, Software engineer wage
Write documentation	2 590.00	Microsoft Visio

Table 4.7: Budget for each task

Budget monitoring

The specified budget is expected to be stable and big deviations are not foreseen. The budget that will require the most monitoring will be that of Human Resources because the time estimations of each task could be slightly wrong and the project consists of two different engineer roles.

In principle, it's not expected to work more than the allocated hours. Because many of the tasks will consist in comparing different solutions, if time runs short, a few will be abandoned and the attention will be centered around the best. In the exceptional case that more paid hours were needed, the costs would be deducted from the contingency reserve.

Problems with the software and hardware won't cause a deviation in the budget. Commercial software includes support, so the company could be contacted if necessary. If a switch had a problem and stopped working, the tests would be adapted to the situation, which wouldn't be very disruptive. Two major hardware failures would be disruptive but it's extremely improbable, so it's not taken into account.

In order to control the budget, at the end of each task it will be updated. If the working hours have exceeded the number allocated, the corresponding amount will be deducted from another task. Otherwise, the surplus will be reallocated.

5 Sustainability and social commitment

5.1 Environmental dimension

This project has a moderate environmental impact due to the power consumption of the switches during the tests, which is unavoidable. On the other hand, this amount is negligible when taking into account that the DC has over 400 switches operating 24h a day. The useful life of the implemented architecture won't have an extra environmental impact. In principle, the hardware used will be the same which is used today or similar, and as a consequence the power consumption won't increase. In summary, this part is rated with a 7 out of 10, because the total energy consumption during the project will be 311.6kWh; a 20 out of 20 useful life and a 0 risk.

5.2 Economic dimension

The economic impact is significant due to the high costs of the switches. However, they have a 5 year useful life, so the depreciation is small. After the project, they will be used for everyday operations in the data center. The costs of the resources couldn't be lowered because the tests should be carried out with the hardware already used in the data center, in order to obtain reliable results. This part will be rated with a 6/10 and, because the hardware resources will be reused, a 20/20 and a 0 risk.

5.3 Social dimension

The social impact can be considered high. First, making this project will allow the author to learn how a large-scale data center works in detail, which is a very unique skill.

Second, the documentation of this project will be useful for the engineering team to decide if the studied architecture should be implemented. If done, during its lifetime the IP Fabric architecture could reduce server downtime by making the network more robust. Therefore, users of services in the data center will be more productive. The impact is even more important when considering that CERN is an international particle physics research institution, whose work serves society. For these reasons, the social dimension is rated with a 10/10 and a 20/20. The only risk is that implementing the architecture can be operationally difficult, and some services may need to be stopped during the upgrade. The risk is rated a -5.

5.4 Sustainability matrix

	PPP	Useful life	Risks
Environmental	7	20	0
Economic	6	20	0
Social	10	20	-5
Subtotal	23	60	-5
Total		78	

Table 5.1: Sustainability matrix

Part II

The evaluation

6 IP Fabric features and options

We start the evaluation by presenting the most important features of the IP Fabric and discussing different options. For an introduction to the architecture, see section 2.2. This chapter concerns the underlay of the fabric – the physical infrastructure; for a discussion of the overlay see chapter 9.

6.1 Routing protocol

For mid-scale data centers, OSPF and Intermediate System to Intermediate System (IS-IS) can be used. For large-scale data centers like CERN's, BGP should be used due to its scalability and the traffic engineering capabilities it provides. Although BGP is known as the de facto interdomain routing protocol in the Internet, it has been recently introduced in the data center.

Both internal BGP (iBGP) and external BGP (eBGP) can be used, exclusively or combined. The use of one or the other impacts the complexity of the required configuration and the routes available in the Fabric.

6.2 Route control

To maximize redundancy, all routes that have a physical path should be available, however, this has a cost and may not be viable. In theory, convergence time increases with the number of routers that need to update their BGP table due to a failure [5, p. 24]. Moreover, the number of existing paths is very large due to the mesh between Leaves and Spines, and the mesh between Spines and SS. Thus, convergence time and BGP table sizes should be estimated.

Many paths in the Fabric are suboptimal and might not be necessary or ever used, so they shouldn't be advertised. BGP peers advertise only the best route to a destination according to the selection algorithm [32]. Below we write in order the preferences of the algorithm, which might be slightly different depending on the implementation:

1. Highest LOCAL_PREF (Local Preference)
2. Shortest AS_PATH
3. Lowest origin value
4. Lowest MED (Multi-Exit Discriminator)
5. Prefer eBGP over iBGP

6. Lowest IGP cost to the next-hop
7. Lowest router ID
8. Lowest peer address

There are 3 types of suboptimal paths that are abundant and worth analyzing, as shown in figure 6.1:

- From a Spine to a Leaf using another Leaf (the Red paths): suppose Leaf H selects route D-I to reach a host connected to Leaf I as best. Then, it will advertise route H-D-I to Spine E, which will add it to the BGP table.
- From a SS to a Spine using another Spine (the Orange paths): suppose Spine D selects route A-F-J to reach a host connected to Leaf J as best. Then, it will advertise route D-A-F-J to SS B, which will add it to the BGP table.
- From a Leaf to a Leaf of the same cluster using a SS (the Yellow paths): suppose SS C selects route G-K to reach a host connected to Leaf K as best. Then, it will advertise route C-G-K to Spine F, which will add it to the BGP table.

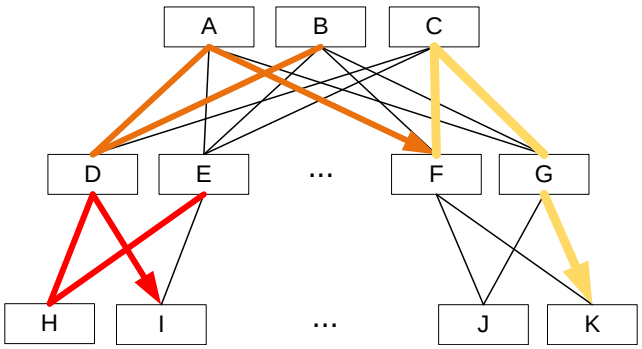


Figure 6.1: Suboptimal paths

In section 7 we will analyze quantitatively the consequences of having these three types of routes.

At the moment, we can state that Orange paths are necessary in the current physical architecture because Super Spines summarize data center networks to Core routers (these are the core of the global CERN network). Core routers don't have knowledge of the available DC routes and they send traffic to A, B and C. In case of two link failures (for example B-F and B-G), traffic from outside the DC could be lost if orange paths weren't allowed (for example traffic in B towards a host connected to J).

If all paths are available, it might be preferred that traffic only goes up and down one time. For example, yellow and red paths have the same cost in number of hops. To prefer yellow paths, SS should send routes to Spines with a lower MED than Leaves do.

In any case, it may be a good idea to set all route advertisements from all routers with a MED 150 (around the middle of the range), so in the future individual routes can be preferred or penalized.

Default routes

Default routes should be sent by SS so Leaves know the best path to external routes. If they are configured as static in each Leaf, traffic will be lost in some cases. For example, if a Spine loses all uplinks, Leaves will still send traffic to it.

6.3 iBGP

iBGP requires a full mesh of sessions and as this does not scale, Route Reflectors (RR) [33] should be used. The problem with route reflection is that it only reflects the best route. To enable ECMP, we need the AddPath BGP capability [34], which adds additional paths into the advertisements between RR and clients. Unfortunately, it is not widely supported yet.

To use iBGP exclusively, all Spines should be RR of their connected Leaves and SS should be RR of all Spines. This configuration filters Red and Orange paths. In addition, because RRs don't change the next-hop when they reflect routes, next-hop-self should be used in order to avoid relying in an IGP to solve the next-hop recursively.

Alternatively, iBGP and eBGP can be mixed. In this scenario, Spines should be RR of their connected Leaves, all in the same AS. Spines and SS would have eBGP sessions. This configuration filters Red and Yellow paths.

6.4 eBGP

Using eBGP exclusively is the most flexible and straightforward solution, but a distribution of Autonomous System Numbers (ASN) must be chosen. All paths can be allowed or filtered with ASNs or route-maps (router policies).

Because CERN runs a dual-stack network, IPv4 and IPv6 routes can be advertised in the same or different sessions. Having an IPv4 session to send IPv4 routes and an IPv6 session to send IPv6 routes is the simplest option. Sending IPv6 routes to an IPv4 peer requires setting an IPv6 next-hop to them with a route-map. The downside of having two sessions is that it implies more monitoring work. There might be some cases in which alarms will be duplicated.

In conclusion, the best option is a full eBGP IP Fabric with a separate session for each IP version, because it requires less configuration and features than the iBGP alternative. From now on, we will assume we are working with eBGP. In section 8 we discuss the ASN distribution options.

6.5 Clusters

A cluster (or pod) is defined as a set of Leaves connected to the same Spines. They can be used to separate groups of servers that need specific uplink bandwidth or protocols. By separating computing and storage servers in different clusters, some features could be implemented only where needed, like EVPN or 40GbE uplinks.

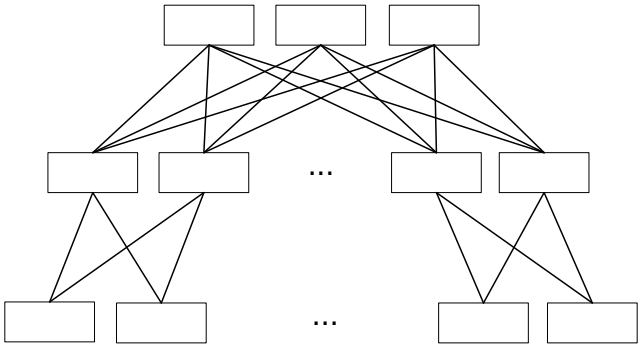


Figure 6.2: IP Fabric with clusters of 2 Spines

Not using clusters makes the physical architecture more flexible, since Leaves can be connected to any Spine, but makes the number of paths available and the routes used more unpredictable if there's not a good and consistent organization. For this reason, using clusters may be a better solution even if they aren't homogeneous.

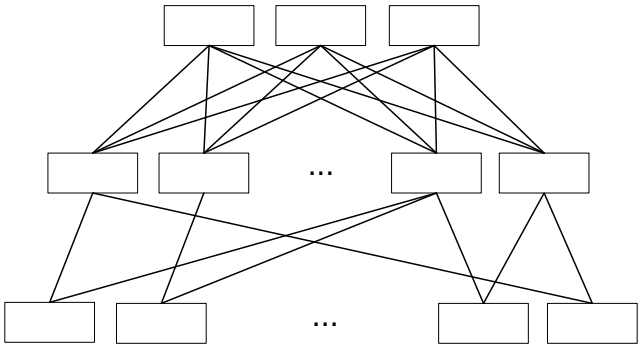


Figure 6.3: IP Fabric without clusters

6.6 Load balancing

One of the main features of the IP Fabric is that all uplinks can be active if using load-balancing. Leaves can load-balance to their Spines, Spines to all Super Spines, and these to every group of Spines. To achieve this, Equal Cost Multi Path (ECMP) is used, which allows to load-balance between multiple paths of the same cost by adding multiple entries towards the same destination in the routing table.

For example, let's take a look at the fragment of a routing table shown below. The entry 220 is duplicated, because to reach 10.93.10.25 there are two gateways: 10.90.128.21 and 10.90.128.29. This is in the routing table of an ECMP-enabled Leaf connected to two Spines.

```
220     10.93.10.25/32     10.90.128.21     ve 287    20/0    Be    1d1h
        10.93.10.25/32     10.90.128.29     ve 289    20/0    Be    1d1h
221     10.93.10.26/32     10.90.128.21     ve 287    20/0    Be    1d1h
        10.93.10.26/32     10.90.128.29     ve 289    20/0    Be    1d1h
222     10.93.10.27/32     10.90.128.21     ve 287    20/0    Be    1d1h
        10.93.10.27/32     10.90.128.29     ve 289    20/0    Be    1d1h
```

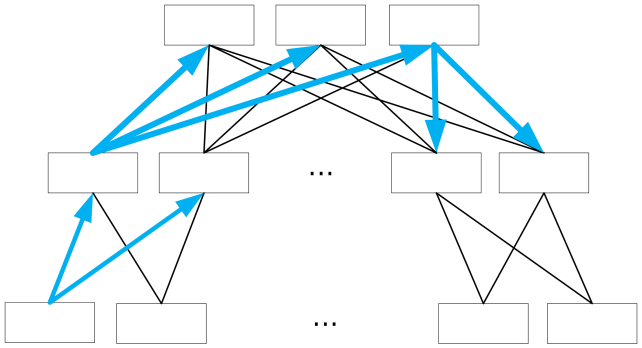


Figure 6.4: ECMP

6.7 Point-to-Point networks

For Point-to-Point (P2P) links in IPv4, netmasks /31 can be used to save addresses [35]. Public addresses are needed to allow Path MTU Discovery [36], which might be required by external sites in order to avoid packet fragmentation.

In many cases, more than one uplink from a Leaf to a Spine will be needed to fulfill bandwidth requirements. This can be done by setting a P2P subnet for each uplink or by setting a LAG. The former multiplies the number of IPs and learned routes and requires configuring extra BGP sessions. The latter will require the router to do load-balancing in Layer 2 (between members of a LAG) and Layer 3 (between LAGs going to different Spines).

Assuming 400 Leaves, there will at least 800 P2P prefixes in IPv4 and also in IPv6. Options:

- Do not advertise them.
- Advertise them in BGP.
  - Summarize them on every device. Requires a good address allocation scheme.
  - Don't summarize them.

- Advertise them in OSPF. The advantage is that if something goes wrong with BGP, there will still be access to the routers. The disadvantage is that we have to configure another protocol. We have the option to advertise IPv4 in OSPFv2 and IPv6 in OSPFv3 or both in OSPFv3, if the router supports it [37].

6.8 Maintenance

When putting a Spine or SS on maintenance, it might be wanted to avoid transit through it. There are two easy ways of doing it without taking down BGP sessions:

- Withdraw all routes to peers. Can be done by applying a route-map that denies all outbound routes. It's simple and easy to automate.

```
route-map ON_MAINTENANCE deny 10

address-family ipv4 unicast
neighbor leaf-group route-map out ON_MAINTENANCE
neighbor ss-group route-map out ON_MAINTENANCE
address-family ipv6 unicast
neighbor leaf-group-v6 route-map out ON_MAINTENANCE
neighbor ss-group-v6 route-map out ON_MAINTENANCE
```

- Use AS-prepending, so peers never select its paths as best. Must be adapted to each ASN and neighbors will still see the routes in their BGP tables.

```
route-map PREPEND permit 10
match ip address ANY_IP
set as-path prepend  65002 65002 65002 65002
```

6.9 Convergence

BGP is by default a protocol with slow convergence, due to the predefined timers in most implementations (e.g. Keepalive=60s, Hold-down=180s). To minimize the convergence time, link failure should be detected immediately in order to start the update process – we cannot depend on timers. At the same time, a system to avoid the effects of route flapping is needed.

Link failure

The fast-fallover router feature should be used to take down the BGP session immediately when the link to a peer goes down.

For rare situations in which there's a link failure but it is not detected (the link appears up), Bidirectional Forwarding Detection (BFD) [38] or Link Aggregation Control Protocol

(LACP) could also be used. BFD provides faster detection (e.g. latency of 50ms), which might be useful for applications like VoIP. If the fabric already has LAGs, LACP could be used everywhere (including single links) for consistency. BFD for LAGs [39] is quite recent and not widely supported.

Apart from link failure, there can be other situations in which a BGP session goes down, like misconfiguration. For this reason, timers are still relevant but should be decreased accordingly.

Route flapping

BGP has two mechanisms to control the frequency of route advertisement: MinRouteAdvertisementInterval (MRAI) and Route Flap Damping (RFD) [40]. MRAI deals with very short bursts on the order of a few to 30 seconds. RFD deals with longer bursts, minutes to hours.

MRAI is an internal timer that determines the minimum amount of time that must elapse between advertisements or withdrawals of routes towards a particular destination to a peer. It cannot be modified in most routers.

RFD is a technique for BGP that penalizes unstable routes by restricting their advertisement. According to RFC 7196 [41], RFD may penalize well-behaved prefixes' normal convergence process in sites with a topological richness, because it amplifies the number of update messages exchanged. Therefore, RFD should be tuned accordingly.

Notice that RFD must be enabled in both IP address families.

6.10 Summary

We conclude that external BGP is the most adequate protocol to use for CERN's DC. We have identified necessary and suboptimal paths, the benefits of organizing the routers in clusters and the usefulness of ECMP (one of the most important IP Fabric features). We have discussed the options for managing Point-to-Point links, putting a router on maintenance and minimizing convergence time.

7 BGP table scalability

Regarding BGP scalability, there are two factors that must be understood: the number of routes installed and the number of sessions per router. This depends on the configuration of the fabric and should be calculated in order to ensure the routers have the necessary resources.

Because we are assuming eBGP, the number of sessions per router equals the number of routers connected to it. It is easy to verify if the device can handle it by reading its technical sheet.

In this chapter we find the expressions to calculate the number of paths installed in the BGP table in the best and worst cases for Leaves, Spines and Super Spines. The mathematical expressions shown in this section assume that the Fabric is working in normal conditions, that there's only one logical link between two routers and that P2P and loopback networks are not advertised in BGP (the only advertisements come from Leaves, which advertise networks for servers). It is guaranteed that the table size will never be larger than in the worst cases, but might be smaller than in the best cases if some links or routers fail.

7.1 Leaves

A Leave has a path to each network originally advertised by itself and a path to each other network received from its connected Spines. The BGP table size for any Leaf T_L can be expressed as:

$$T_L = N_0 + (N - N_0) \cdot S \tag{7.1}$$

$N_0 =$ number of networks advertised by the Leaf
$N \ =$ total number of networks in the fabric
$S \ =$ number of connected Spines

7.2 Spines

In the best case for a Spine, all SS and Leaves select as best all paths that have the Spine as next-hop. In this case, the Spine receives:

- A route to each network advertised by connected leaves (from them).
- A route to each other network (from each SS).

The BGP table size in the best case for a Spine can be expressed as:

$$N_L + (N - N_L) \cdot SS$$

N_L = total number of networks advertised by connected Leaves
SS = number of Super Spines

In the worst case, SS and Leaves don't select as best any path that has the Spine as next-hop. Then, the Spine receives advertisements classified in 4 groups:

- A route to each network advertised by connected leaves (from them).
- A route to each other network (from each SS).
- A route to each network advertised by connected leaves (from each SS). These are Yellow paths in figure 6.1.
- A route to each other network (from each connected leaf). These are Red paths in figure 6.1.

The BGP table size in the worst case for a Spine can be expressed as the formula below, where the terms represent the groups in order:

$$N_L + (N - N_L) \cdot SS + N_L \cdot SS + \sum_{i=1}^{L}(N - N_{Li}) \tag{7.2}$$

L = number of connected Leaves
N_{Li} = number of networks advertised by connected Leaf i

Simplified:

$$N \cdot (L + SS)$$

In summary, the BGP table size for Spines T_S is:

$$N_L + (N - N_L) \cdot SS \quad \leq \quad T_S \quad \leq \quad N \cdot (L + SS) \tag{7.3}$$

It can be seen that the difference between the best and worst case are the Yellow and Red paths (specially the latter).

7.3 Super Spines

In the best case for a Super Spine, all Spines select as best all paths that have the Super Spine as next-hop. In this case, the Super Spine receives:

- A route to each network originated in every cluster (from each Spine of the cluster).

The BGP table size in the best case for a Super Spine can be expressed as:

$$\sum_{i=1}^{C} S_i \cdot N_i$$

N_i = total number of networks advertised by Leaves in cluster i
S_i = number of Spines in cluster i
C = number of clusters

In the worst case, Spines don't select as best any path that has the Super Spine as next-hop. Then, the Super Spine receives:

- A route to each network originated in every cluster (from each Spine of the cluster).
- A route to each network originated in every cluster (from each Spine of the other clusters). These are "Orange" paths in figure 6.1.

The BGP table size in the worst case for a Super Spine can be expressed as:

$$\sum_{i=1}^{C} S_i \cdot N_i + \sum_{i=1}^{C} S_i \cdot (N - N_i) \tag{7.4}$$

Simplified:

$$N \cdot S$$

In summary, the BGP table size for Super Spines T_{SS} is:

$$\sum_{i=1}^{C} S_i \cdot N_i \quad \leq \quad T_{SS} \quad \leq \quad N \cdot S \tag{7.5}$$

It can be seen that the difference between the best and worst case are the Orange paths.

7.4 Example scenario

The following scenario will be used to calculate the BGP table sizes and the impact of Red, Orange and Yellow paths in a data center similar to CERN's.

- 3 Super Spines ($SS = 3$)
- 8 Spines
- 400 Leaves
- 4 clusters of 2 spines and 100 Leaves each ($C = 4, S = 2, L = 100$)
- Each Leaf advertises 8 prefixes ($N_{Li} = 8$)
- In total, the fabric has 3 200 advertised prefixes ($N = 3200$)

According to expression 7.1, each Leaf will have

$$T_L = 8 + (3200 - 8) \cdot 2 = 6392$$

In case the hardware doesn't support this large number of routes in IPv6, the solution would be to use only default routes in the IPv6 session. This would cause suboptimal routing in cases of multiple link failures in the fabric. However, because nowadays most traffic is IPv4, this could be considered an acceptable risk.

According to expression 7.3, each Spine will have:

$$800 + (3200 - 800) \cdot 3 \quad \leq \quad T_S \quad \leq \quad 3200 \cdot (100 + 3)$$

$$8000 \quad \leq \quad T_S \quad \leq \quad 329\,600$$

As seen in expression 7.2, Yellow paths represent $N_L \cdot SS = 2400$ and Red paths represent $\sum_{i=1}^{L}(N - N_{Li}) = 319\,200$.

This means that the Spine routers should support 329 600 routes in IPv4 and IPv6, most of which are suboptimal. Therefore, allowing Red paths may not be viable and should be filtered.

According to expression 7.5, each Super Spine will have:

$$4 \cdot 2 \cdot 800 \quad \leq \quad T_{SS} \quad \leq \quad 3200 \cdot 8$$

$$6400 \quad \leq \quad T_{SS} \quad \leq \quad 25\,600$$

As seen in expression 7.4, Orange paths represent $\sum_{i=1}^{C} S_i \cdot (N - N_i) = 19\,200$, so their impact is significantly lower than Red.

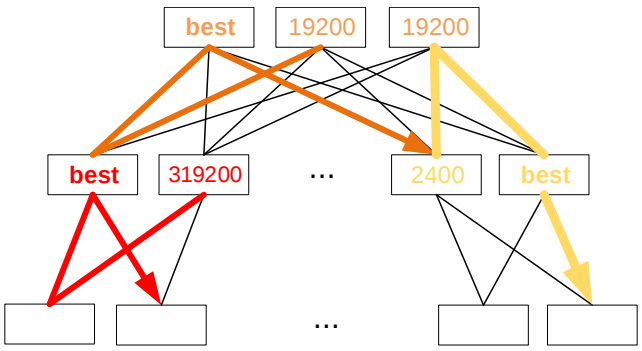


Figure 7.1: Impact of suboptimal routes

7.5 Summary

We have analyzed quantitatively the best-case and worst-case BGP table sizes for each router in the fabric. We have written general mathematical expressions that allow to calculate

these sizes depending on the number of advertised routes, the number of routers in each layer of the fabric and cluster, and how they are connected. These numbers allow to verify if the configuration is viable, and if not justify the filtering of certain groups of routes. In an scenario approximate to CERN's DC, we have found that the number of Red paths is too large and they shouldn't be installed in the BGP tables.

8 ASN distribution policy

When using eBGP, there are many different ASN distributions we can choose. In this chapter we analyze the features of each.

In BGP, there are two ranges of private ASNs [42]. The 16-bit range is small and goes from 64 512 to 65 534 inclusive, for a total of 1 023 numbers. The 32-bit range goes from 4 200 000 000 to 4 294 967 294 inclusive, for a total of 94 967 295 numbers available. One possible downside of 32-bit numbers is that not all routers have complete support. For example, they might not be able to remove these numbers from the AS-path, which will be necessary to do in the border routers so that they can export specific routes outside of CERN.

Because 32-bit numbers are very long, there are two formats to represent them [43]. The asplain format (e.g. 4 259 840 100), is the decimal representation. The asdot+ format (e.g. 65000.100) separates the first and last 16 bits of the number with a dot. This representation is more human friendly, however, if there are existing scripts or configurations that use regular expressions with ASNs, they would need to be adapted to this format.

In addition, assignment policies can be deployed based on rack, room or data center, which might be easier to do with the biggest range.

Thus, there are 3 options:

- Use 16-bit numbers exclusively and use an strategy to assign them efficiently.
- Use 32-bit numbers exclusively.
- Start using 16-bit numbers, and if they run out start using 32-bit numbers.

8.1 ASNs in Leaves

There are 3 possible distributions.

Unique ASN for each Leaf

Routes can be traced by looking at the AS-path, making troubleshooting easy, but requires keeping a database containing all routers and their assigned number.

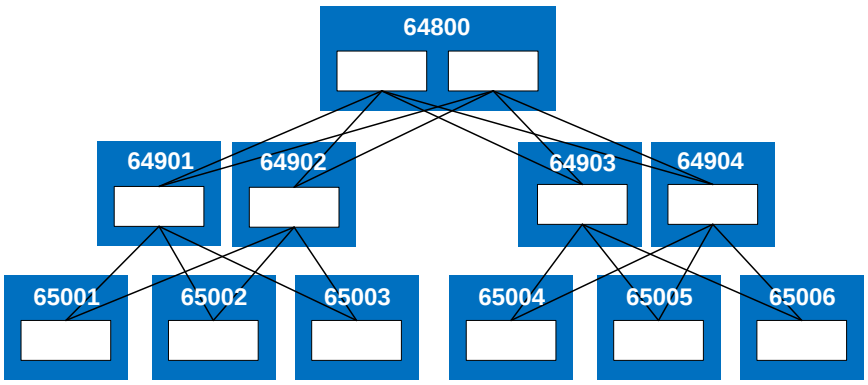


Figure 8.1: eBGP fabric with a unique ASN for each Leaf

Repeated ASNs in each cluster

This solution allows to save ASNs if using 16-bit numbers exclusively, but adds a little more complexity in the database. For this system to work, Leaves must be configured with allowas-in, which allows them to accept paths containing their own ASN.

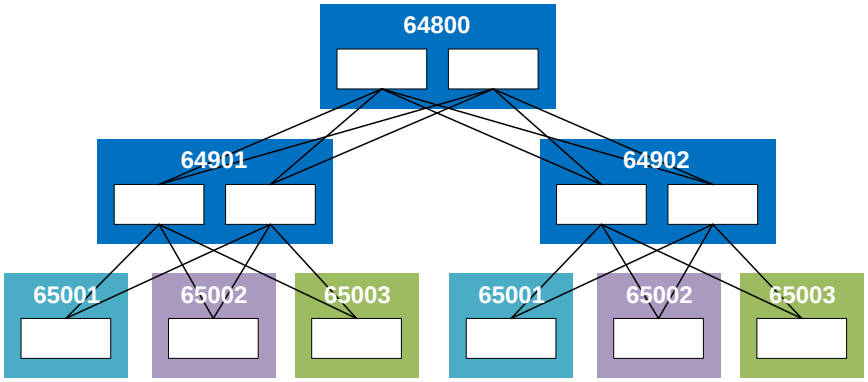


Figure 8.2: eBGP fabric with repeated ASNs in each cluster

Same ASN for all Leaves

To avoid having to manage hundreds of numbers, the same ASN number can be used in all Leaves. This makes automation easier but AS-PATH loses information. Leaves must be configured with allowas-in 2 (2 to allow red paths). Example:

```
neighbor 10.90.128.5 allowas-in 2
neighbor 10.90.128.13 allowas-in 2
```

The allowas-in command must be configured for each connected Spine and in both address-families.

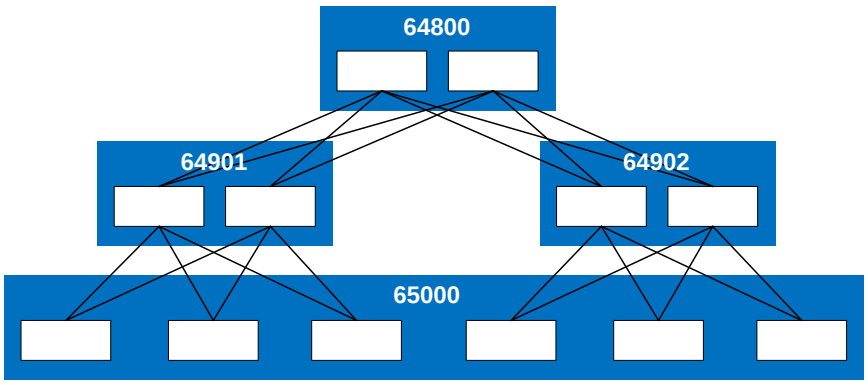


Figure 8.3: eBGP fabric with the same ASN in each Leaf

8.2 ASNs in Spines

At the spine level, the numbers can be distributed in two ways.

Same ASN in Spines of the same cluster

This configuration implies that Red and Yellow routes (see Figure 6.1) are not learned by Spines due to the AS-path check (paths that already contain the ASN are discarded).

Unique ASN for each Spine

This allows to make clusters flexible. For example, a Leaf could be connected to two Spines of different clusters and it still would work. However, to avoid the Red and Yellow paths, route-maps and communities should be used.

To filter Red paths, the following route-map can be applied in Spines towards their Leaves:

```
route-map TO_LEAVES permit 10
set community no-advertise
```

8.3 ASNs in Super Spines

At the Super Spine level, the numbers can be distributed in two ways.

Same ASN in all Super Spines

This configuration implies that Orange routes are not learned by Super Spines due to the AS-path check.

Unique ASN for each Super Spine

In this case, Orange routes can be filtered using route-maps and communities.

The following route-map can be applied in Super Spines towards Spines:

```
route-map TO_SPINES permit 10
set community 65000:555
```

And in Spines towards Super Spines:

```
route-map TO_SUPERSPINES deny 10
match community 65000:555
route-map TO_SUPERSPINES permit 20
```

The community number can be any.

8.4 Summary

Many different ASN distribution policies have been discussed for each layer of the fabric. Some policies require more configuration, extra complexity or explicit filters of unnecessary paths. A combination of policies should be chosen depending on the capabilities of the routers and the preferences of the network engineers.

9 Overlay

After analyzing the underlay of the IP Fabric, we now take a look to the overlay: network virtualization and EVPN.

9.1 Network virtualization

Today's data centers have an increasing demand for computing, storage and network resources from applications. In order to scale, resources are being abstracted from their logical representation, in what is referred to as virtualization.

With server virtualization, each physical server supports multiple Virtual Machines (VMs), each running its own operating system and applications. With network virtualization, the same physical network allows multiple virtual network instances, also called overlays, that isolate traffic from the others. Multi-tenant data centers (e.g. different departments) need to separate resources for security and privacy.

A key feature of virtualization is VM mobility, that is, to migrate from one server to another live while continuing to run. To achieve this, address space separation between tenants must be supported. Also, it must be possible to migrate VMs anywhere in the data center without restricting VM addressing to match the subnet boundaries of the underlying network. For live migration, a VM must retain its IP and MAC address to prevent existing connections (e.g. TCP) from breaking and needing to be restarted [44].

9.2 EVPN

EVPN is a protocol that extends Layer 2 domains over an IP Fabric and supports VM mobility. Unlike other protocols used for similar purposes (e.g. VPLS), it provides separation between the data and control planes. In the control plane, it uses BGP; in the data plane it can use different encapsulation mechanisms, mainly Virtual eXtensible Local Area Network (VXLAN) and Multi-Protocol Label Switching (MPLS).

Data plane

In this document we'll test VXLAN encapsulation, since it is simpler and more common than MPLS in the data center. The VXLAN protocol allows to segment networks similarly to what VLAN does, but it can have up to 16 million administrative domains as opposed to 4000. The key feature of VXLAN is that it provides a mean to extend a Layer 2 network

over Layer 3 network. The endpoints for the tunnels are called Virtual Tunnel EndPoints (VTEP).

The downside of using VXLAN is that it adds and overhead of 50 bytes to each original frame, as shown in figure 9.1.

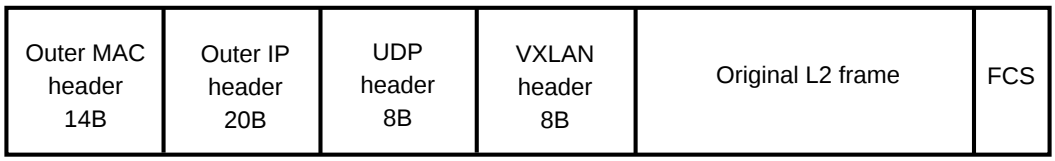


Figure 9.1: VXLAN frame encapsulation

Control plane

EVPN adds a new address family in Multi Protocol BGP. This new address family allows MAC addresses to be treated as routes in the BGP table. The entry can contain just a MAC address or a pair IP address/MAC address (ARP entry). Because the learning happens at the control plane, it can be restricted by applying policies.

Broadcast and multicast traffic can be sent using ingress replication. This is a technique used to avoid flooding frames. Because VTEPs know each other (thanks to BGP), whenever they must flood a frame in a VXLAN segment, they replicates the frame in hardware and send a unicast frame to each of the VTEPs in that segment.

Another feature is ARP/ND suppression. Because Leaves have remote ARP entries from BGP, they can respond to any local ARP queries. This eliminates the need for flooding ARP requests in the network infrastructure.

Unlike in the underlay, iBGP is the best option for an EVPN overlay. This is because iBGP advertisements don't change the next-hop and this is the desired behavior. VTEPs don't have knowledge of the underlay, they just know each other – the other endpoint of the tunnel, which should be the next-hop.

Finally, a choice to be made is the placement of L3 gateways, that is, the gateways to route from one VXLAN to another. Ideally, they will be located in the Leaves, because they also contain the VTEPs and this would allow optimal routing. However, not all ToR switches support it, so they can be placed at the Spine level.

9.3 Summary

In this chapter we introduce the need for network virtualization and its advantages. We discuss EVPN, the state-of-the-art protocol to create overlays in the data center. Unlike in the underlay, iBGP is the best option for an EVPN overlay.

10 Proposed solution

After reviewing the main features and options of the IP Fabric with the engineers in the CE section, we have made the following decisions:

- Routing protocol: eBGP. For simplicity, IPv4 sessions will be used to advertise IPv4 routes and IPv6 sessions for IPv6 routes.
- Route control: Red paths will be filtered. Yellow and Orange paths will be allowed.
- Clusters: yes, heterogeneous (mixing computing and storage servers).
- P2P networks: advertised in OSPFv3.
- Maintenance: a route-map will be used to withdraw all advertisements from the router.
- Convergence: LACP, Fast Fallover and Route Damping will be used.
- ASN distribution: start with 16-bit ASNs. Individual ASN for SS and Spines, same ASN in all Leaves.
- Overlay: EVPN with iBGP and VXLAN.

This design has been selected as the best and preferred for CERN's data center. In principle, the current equipment has all features needed to implement this architecture except for EVPN support. Here and in the following chapters, we test it in the lab.

10.1 Setup

The tests have been carried out with 2 Brocade SLX 9540-48S, 4 Brocade ICX7750-48F, 1 Brocade MLXe-16, and 1 Spirent HyperMetrics SPT-11U tester device.

To do a few basic tests the topology showed in figure 10.3 has been built and configured. The goal of these initial tests is to get acquainted with the configuration and to check that the analyses made in previous chapters are correct.

In each router there is an IPv4 and an IPv6 BGP session for each neighbor. To simulate that each Leaf has 8 servers connected, each advertises 8 loopback addresses that are routed to null. Each Spine sends the routes to the Leaves with the community no-advertise, in order to prevent Red paths.

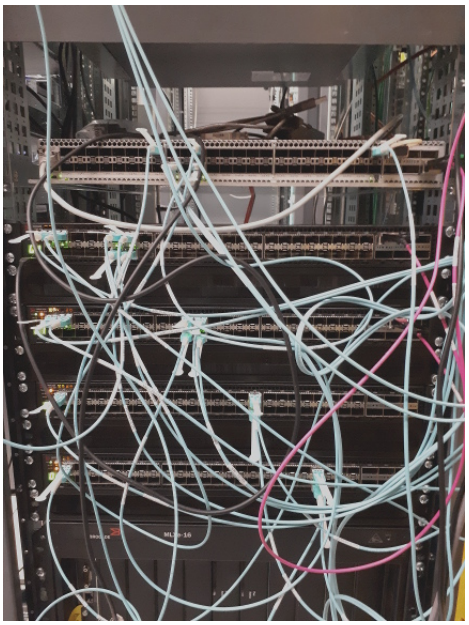


Figure 10.1: Brocade switches during test

Figure 10.2: Spirent during test

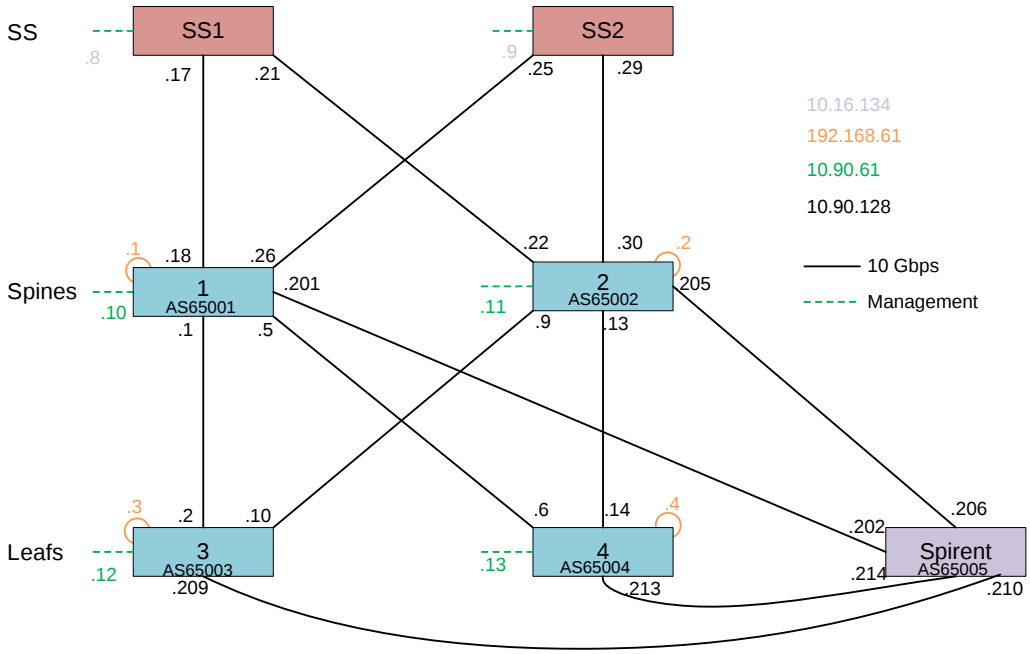


Figure 10.3: Basic BGP tests setup

10.2 Configuration

Here we show the configuration for each device. It includes BGP sessions dual-stack, ECMP, the networks advertised and route-maps to prevent Red paths.

SS configuration

```
router bgp
local-as 65111
capability as4 enable
fast-external-fallover
neighbor fd01:1458:306:7::12 remote-as 65001
neighbor fd01:1458:306:8::16 remote-as 65002
neighbor 10.90.128.18 remote-as 65001
neighbor 10.90.128.22 remote-as 65002

address-family ipv4 unicast
maximum-paths 2
multipath multi-as

address-family ipv6 unicast
maximum-paths 2
multipath multi-as
neighbor fd01:1458:306:8::16 activate
neighbor fd01:1458:306:7::12 activate
```

Spine configuration

```
router bgp
local-as 65001
capability as4 enable
fast-external-fallover
neighbor leaf-group peer-group
neighbor leaf-group-v6 peer-group
neighbor 10.90.128.2 remote-as 65003
neighbor 10.90.128.2 peer-group leaf-group
neighbor 10.90.128.6 remote-as 65004
neighbor 10.90.128.6 peer-group leaf-group
neighbor 10.90.128.17 remote-as 65111
neighbor 10.90.128.25 remote-as 65222
neighbor 10.90.128.202 remote-as 65005
neighbor 10.90.128.202 peer-group leaf-group
neighbor fd01:1458:306:1::2 remote-as 65003
neighbor fd01:1458:306:1::2 peer-group leaf-group-v6
neighbor fd01:1458:306:3::6 remote-as 65004
neighbor fd01:1458:306:3::6 peer-group leaf-group-v6
neighbor fd01:1458:306:7::11 remote-as 65111
neighbor fd01:1458:306:9::19 remote-as 65222
neighbor fd01:1458:306:c8::ca remote-as 65005
neighbor fd01:1458:306:c8::ca peer-group leaf-group-v6

address-family ipv4 unicast
maximum-paths 2
multipath multi-as
neighbor leaf-group route-map out TO_LEAVES
neighbor leaf-group send-community

address-family ipv6 unicast
maximum-paths 2
multipath multi-as
neighbor leaf-group-v6 activate
neighbor leaf-group-v6 route-map out TO_LEAVES_V6
neighbor leaf-group-v6 send-community
neighbor fd01:1458:306:1::2 activate
neighbor fd01:1458:306:3::6 activate
neighbor fd01:1458:306:7::11 activate
neighbor fd01:1458:306:9::19 activate
neighbor fd01:1458:306:c8::ca activate
```

Leaf configuration

```
router bgp
local-as 65004
capability as4 enable
fast-external-fallover
neighbor 10.90.128.5 remote-as 65001
```

```
neighbor 10.90.128.13 remote-as 65002
neighbor fd01:1458:306:3::5 remote-as 65001
neighbor fd01:1458:306:5::d remote-as 65002

address-family ipv4 unicast
maximum-paths 2
multipath multi-as
network 10.90.14.12/32
network 10.90.14.13/32
network 10.90.14.14/32
network 10.90.14.15/32
network 10.90.14.16/32
network 10.90.14.17/32
network 10.90.14.18/32
network 10.90.14.11/32

address-family ipv6 unicast
maximum-paths 2
multipath multi-as
network fd01:1458:306:aa11::/64
network fd01:1458:306:aa12::/64
network fd01:1458:306:aa13::/64
network fd01:1458:306:aa14::/64
network fd01:1458:306:aa15::/64
network fd01:1458:306:aa16::/64
network fd01:1458:306:aa17::/64
network fd01:1458:306:aa18::/64
neighbor fd01:1458:306:3::5 activate
neighbor fd01:1458:306:5::d activate
```

10.3 BGP table

In the Spine BGP table, we can see 16 routes, 8 for each Leaf, as expected.

```
#show ip bgp
Total number of BGP Routes: 16
Status codes: s suppressed, d damped, h history, * valid,  [...]
Origin codes: i - IGP, e - EGP, ? - incomplete
Network            Next Hop         MED    LocPrf     Weight Path
*>  10.90.13.11/32     10.90.128.2     0      100        0      65003 i
*>  10.90.13.12/32     10.90.128.2     0      100        0      65003 i
*>  10.90.13.13/32     10.90.128.2     0      100        0      65003 i
*>  10.90.13.14/32     10.90.128.2     0      100        0      65003 i
*>  10.90.13.15/32     10.90.128.2     0      100        0      65003 i
*>  10.90.13.16/32     10.90.128.2     0      100        0      65003 i
*>  10.90.13.17/32     10.90.128.2     0      100        0      65003 i
*>  10.90.13.18/32     10.90.128.2     0      100        0      65003 i
*>  10.90.14.11/32     10.90.128.6     0      100        0      65004 i
*>  10.90.14.12/32     10.90.128.6     0      100        0      65004 i
```

```
*>  10.90.14.13/32     10.90.128.6     0       100         0       65004 i
*>  10.90.14.14/32     10.90.128.6     0       100         0       65004 i
*>  10.90.14.15/32     10.90.128.6     0       100         0       65004 i
*>  10.90.14.16/32     10.90.128.6     0       100         0       65004 i
*>  10.90.14.17/32     10.90.128.6     0       100         0       65004 i
*>  10.90.14.18/32     10.90.128.6     0       100         0       65004 i
```

When all routers have converged, the size of their BGP tables is consistent with the expressions calculated in section 7. Below are shown the number of installed routes in each router.

Leaf

```
#show ip bgp | include Total
Total number of BGP Routes: 24
#show ipv6 bgp | include Total
Total number of BGP Routes: 24
```

Spine best

```
#show ip bgp | include Total
Total number of BGP Routes: 16
#show ipv6 bgp | include Total
Total number of BGP Routes: 16
```

Spine worst

```
#show ip bgp | include Total
Total number of BGP Routes: 48
#show ipv6 bgp | include Total
Total number of BGP Routes: 48
```

Super Spine

Since there are not other clusters, both have the same number of routes.

```
#show ip bgp | include Total
Total number of BGP Routes: 32
#show ipv6 bgp | include Total
Total number of BGP Routes: 32
```

10.4 Summary

In this chapter we have presented our architecture proposal for CERN's DC. We have constructed a basic IP Fabric in order to get acquainted with the Brocade configuration and verify that the research we have made in previous chapters is correct.

11 Advanced BGP tests

11.1 Setup

To test the behavior of the fabric in a more realistic scenario, it is necessary to have a large number of routers and advertisements. To simulate this, another router has been connected to the Spines which contains multiple Virtual Routing and Forwarding (VRF) instances. Basically, each one represents a virtual router. The script written to generate the configuration can be found in appendix A.

In this test there are 2 clusters, each one has 10 leaves and each leaf advertises 24 networks dual stack.

The debug mode has been enabled in all routers in order to inspect BGP behavior in detail and to be able to calculate convergence times. The commands to enable the mode are:

```
debug destination telnet 1
debug ip bgp
debug ip bgp general
debug ip bgp events
debug ip bgp updates
debug ip bgp route-selection
```

Figure 11.1 shows the physical setup. The green switch is the Brocade MLX, the red switches are Brocade SLX, the blue switches are Brocade ICX and the purple box is the Spirent.

Figure 11.2 shows the logical setup. The two SLX (R10 and R20) act as SS, two ICX (R1 and R2) act as Spines in a cluster, the other two ICX (R3 and R4) act as Leaves in this cluster. The MLX provides 2 VRFs that act as 2 Spines in another cluster, and 8 other VRFs that act as Leaves connected to R1 and R2. Spirent emulates 20 servers connected to R3 and 1 server connected to R4. The 2 VRFs Spines in the second cluster are configured as if they had 10 Leaves connected and there where 20 servers behind.

Figure 11.3 shows the fabric we are simulating, where each level is (in order): Super Spine, Spine, Leaf, server.

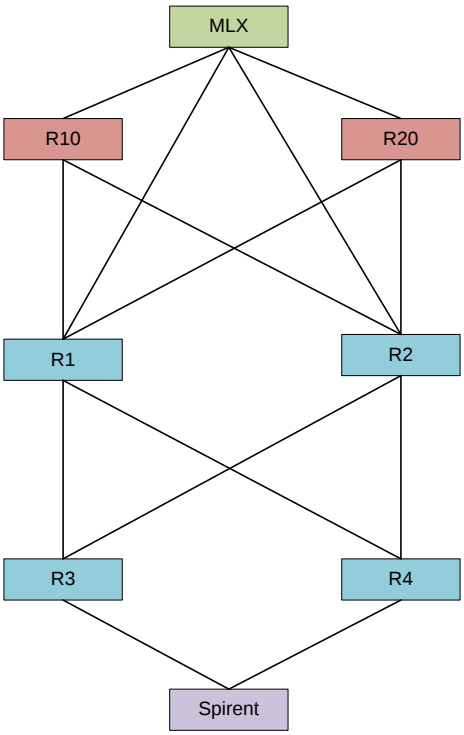


Figure 11.1: Physical setup

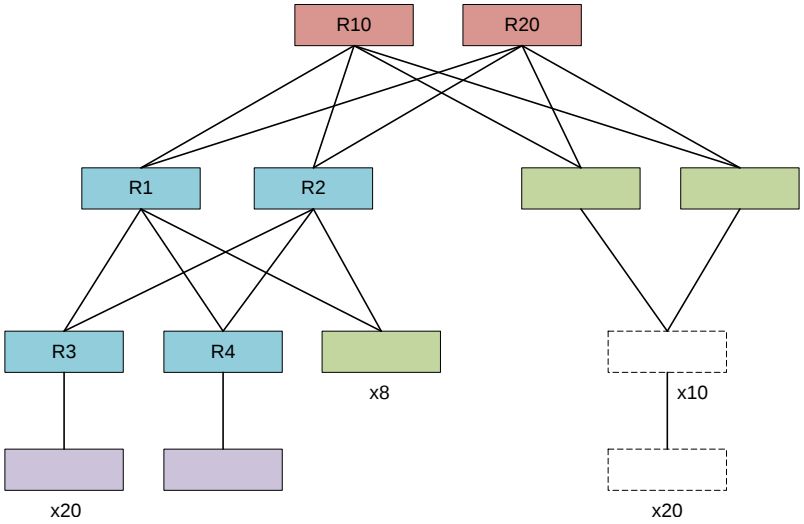


Figure 11.2: Logical setup

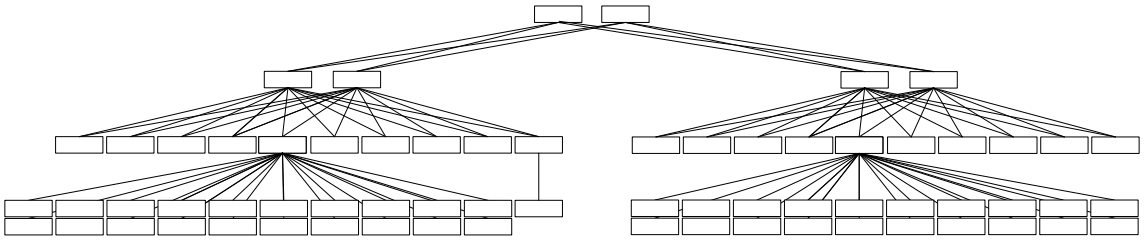


Figure 11.3: Simulation schema

11.2 Route advertisement

Once the fabric has converged, we can check the BGP statistics for each router. Remember that we have 2 SS and 2 clusters with 2 Spines and 10 Leaf each. Each Leaf advertises 24 networks. In this test no paths have been filtered.

Leaves

First we check Leaves R3 and R4. Let's rewrite equation 7.1 to find the number of routes that should be installed in their BGP tables:

$$T_L = N_0 + (N - N_0) \cdot S$$

In this case $N_0 = 24, N = 480, S = 2$. Therefore:

$$T_L = 24 + (480 - 24) \cdot 2 = 936$$

As we can see below, the result is consistent with what the routers state.

R3

```
#show ip bgp | include Total
Total number of BGP Routes: 936

#show ipv6 bgp | include Total
Total number of BGP Routes: 936

#show ip bgp summary
Number of Routes Installed: 936, Uses 80496 bytes
Number of Routes Advertising to All Neighbors: 504 (480 entries), [...]
Number of Attribute Entries Installed: 21, Uses 1890 bytes
Neighbor Address  AS#         State   Time          Rt:Accepted Sent
10.90.128.1       65001       ESTAB   2d19h37m      456         24
10.90.128.9       65002       ESTAB   2d19h37m      456         480
```

R4

```
#show ip bgp | include Total
Total number of BGP Routes: 936

#show ipv6 bgp | include Total
Total number of BGP Routes: 936

#show ip bgp summary
Number of Neighbors Configured: 2, UP: 2
Number of Routes Installed: 936, Uses 80496 bytes
Number of Routes Advertising to All Neighbors: 504 (480 entries), [...]
Number of Attribute Entries Installed: 21, Uses 1890 bytes
Neighbor Address  AS#         State   Time          Rt:Accepted Sent
10.90.128.5       65001       ESTAB   2d19h39m      456         24
10.90.128.13      65002       ESTAB   2d19h39m      456         480
```

Spines

Now we check Spines R1 and R2. Let's rewrite expression 7.3 to find the number of routes that should be installed in their BGP tables:

$$N_L + (N - N_L) \cdot SS \quad \leq \quad T_S \quad \leq \quad N \cdot (L + SS)$$

In this case $N_L = 240, N = 480, SS = 2, L = 10$. Therefore:

$$240 + (480 - 240) \cdot 2 \quad \leq \quad T_S \quad \leq \quad 480 \cdot (10 + 2)$$

$$720 \quad \leq \quad T_S \quad \leq \quad 5760$$

As we can see below, the result is consistent with what the routers state. R1 has been favored by all leaves.

R1

```
#show ip bgp | include Total
Total number of BGP Routes: 720

#show ipv6 bgp | include Total
Total number of BGP Routes: 720

#show ip bgp summary
Number of Neighbors Configured: 12, UP: 12
Number of Routes Installed: 720, Uses 61920 bytes
Number of Routes Advertising to All Neighbors: 6240 (960 entries), [...]
Number of Attribute Entries Installed: 23, Uses 2070 bytes
Neighbor Address  AS#         State   Time          Rt:Accepted Sent
10.90.128.2       65003       ESTAB   2d19h46m      24          456
```

```
10.90.128.6       65004       ESTAB   2d19h47m      24          456
10.90.128.17      65111       ESTAB   2d19h52m      240         240
10.90.128.25      65222       ESTAB   2d19h52m      240         480
10.94.0.2         65010       ESTAB   2d19h52m      24          456
10.94.1.2         65011       ESTAB   2d19h52m      24          456
10.94.2.2         65012       ESTAB   2d19h52m      24          456
10.94.3.2         65013       ESTAB   2d19h52m      24          456
10.94.4.2         65014       ESTAB   2d19h52m      24          456
10.94.5.2         65015       ESTAB   2d19h52m      24          456
10.94.6.2         65016       ESTAB   2d19h52m      24          456
10.94.7.2         65017       ESTAB   2d19h52m      24          456
```

R2

```
#show ip bgp | include Total
Total number of BGP Routes: 5760
#show ipv6 bgp | include Total
Total number of BGP Routes: 5760

#show ip bgp summary
Number of Neighbors Configured: 12, UP: 12
Number of Routes Installed: 5760, Uses 495360 bytes
Number of Routes Advertising to All Neighbors: 6240 (960 entries), [...]
Number of Attribute Entries Installed: 143, Uses 12870 bytes
Neighbor Address  AS#         State   Time          Rt:Accepted Sent
10.90.128.10      65003       ESTAB   2d19h50m      480         456
10.90.128.14      65004       ESTAB   2d19h51m      480         456
10.90.128.21      65111       ESTAB   2d19h56m      480         240
10.90.128.29      65222       ESTAB   2d19h56m      480         480
10.95.0.2         65010       ESTAB   2d19h56m      480         456
10.95.1.2         65011       ESTAB   2d19h56m      480         456
10.95.2.2         65012       ESTAB   2d19h56m      480         456
10.95.3.2         65013       ESTAB   2d19h56m      480         456
10.95.4.2         65014       ESTAB   2d19h56m      480         456
10.95.5.2         65015       ESTAB   2d19h56m      480         456
10.95.6.2         65016       ESTAB   2d19h56m      480         456
10.95.7.2         65017       ESTAB   2d19h56m      480         456
```

Super Spines

Now we check SS R10 and R20. Let's rewrite expression 7.5 to find the number of routes that should be installed in their BGP tables:

$$\sum_{i=1}^{C} S_i \cdot N_i \quad \leq \quad T_{SS} \quad \leq \quad N \cdot S$$

In this case $C = 2, S_i = 2, N_i = 240, N = 480, S = 4$. Therefore:

$$2 \cdot 240 + 2 \cdot 240 \quad \leq \quad T_{SS} \quad \leq \quad 480 \cdot 4$$

$$960 \quad \leq \quad T_{SS} \quad \leq \quad 1920$$

As we can see below, the result is consistent with what the routers state.

There's an event that appears in the outputs and is worth mentioning: in R10 and R20 it can be seen that they don't have the same number of routes installed in the IPv4 and IPv6 sessions. This is likely to happen if both sessions aren't configured at the same time or in different order.

R10

```
#show ip bgp | include Total
Total number of BGP Routes: 960
#show ipv6 bgp | include Total
Total number of BGP Routes: 1440

Number of Neighbors Configured: 4, UP: 4
Number of Routes Installed: 960, Uses 120000 bytes
Number of Routes Advertising to All Neighbors: 1440 (960 entries), [...]
Number of Attribute Entries Installed: 33, Uses 3795 bytes
'+': Data in InQueue '>': Data in OutQueue '-': Clearing
'*': Update Policy 'c': Group change 'p': Group change Pending
'r': Restarting 's': Stale '^': Up before Restart '<': EOR waiting
Neighbor Address  AS#         State     Time     Rt:Accepted Sent
10.90.128.18      65001       ESTAB     2d20h 4m    240      240
10.90.128.22      65002       ESTAB     2d20h 5m    240      480
10.94.100.2       65301       ESTAB     2d20h 2m    240      240
10.94.101.2       65302       ESTAB     2d20h 2m    240      480

Number of Neighbors Configured: 4, UP: 4
Number of Routes Installed: 1440, Uses 180000 bytes
Number of Routes Advertising to All Neighbors: 1440 (480 entries), [...]
Number of Attribute Entries Installed: 24, Uses 2760 bytes
'+': Data in InQueue '>': Data in OutQueue '-': Clearing
'*': Update Policy 'c': Group change 'p': Group change Pending
'r': Restarting 's': Stale '^': Up before Restart '<': EOR waiting
Neighbor Address  AS#         State     Time     Rt:Accepted Sent
fd01:1458:306:7::12
65001       ESTAB     2d20h 7m    480              240
fd01:1458:306:8::16
65002       ESTAB     2d20h 8m    480              480
fd01:1458:306:94a0::2
65301       ESTAB     2d20h 5m    240              240
fd01:1458:306:94a1::2
65302       ESTAB     2d20h 5m    240              480
```

R20

```
#show ip bgp | include Total
Total number of BGP Routes: 1920
#show ipv6 bgp | include Total
Total number of BGP Routes: 1440

Number of Neighbors Configured: 4, UP: 4
Number of Routes Installed: 1920, Uses 240000 bytes
Number of Routes Advertising to All Neighbors: 1440 (960 entries), [...]
Number of Attribute Entries Installed: 55, Uses 6325 bytes
Neighbor Address  AS#         State     Time     Rt:Accepted Sent
10.90.128.26      65001       ESTAB     2d20h 5m    480      240
10.90.128.30      65002       ESTAB     2d20h 6m    480      480
10.95.100.2       65301       ESTAB     2d20h 3m    480      240
10.95.101.2       65302       ESTAB     2d20h 3m    480      480

Number of Neighbors Configured: 4, UP: 4
Number of Routes Installed: 1440, Uses 180000 bytes
Number of Routes Advertising to All Neighbors: 1440 (480 entries), [...]
Number of Attribute Entries Installed: 42, Uses 4830 bytes
Neighbor Address  AS#         State     Time     Rt:Accepted Sent
fd01:1458:306:9::1a
65001       ESTAB     2d20h 9m    240      240
fd01:1458:306:a::1e
65002       ESTAB     2d20h10m    240      480
fd01:1458:306:95a0::2
65301       ESTAB     2d20h 7m    480      240
fd01:1458:306:95a1::2
65302       ESTAB     2d20h 7m    480      480
```

After analyzing all the outputs, we can confirm the fabric works as expected and the number of routes advertised and learned are correct. During these tests we have found that the Brocade ICX switches do not support more than 20 BGP peers in a stable manner. We have found this number by adjusting the number of peers and networks advertised until it was stable. This means that the current hardware is not ideal to implement the IP Fabric.

11.3 Traffic load-balancing

After testing the basic elements of the Fabric, we test it with traffic generated by Spirent to observe the behavior. One important feature is ECMP, so we test it next.

Traffic to same cluster

In this test there are 20 0.5Gbps flows with different source IP and to the same IP destination. It simulates 20 servers connected to R3 sending traffic to network 10.90.14.11/32 advertised by R4, which will drop the packets (routed to null). All links are 10Gbps.

To check ECMP works correctly, the output rates of the relevant interfaces are shown. The traffic originates from a server connected to R3, so this one should split the traffic:

R3

R3 interfaces: 1/1/31 to R1, 1/1/32 to R2

```
#show interfaces ethernet 1/1/31 | include output rate
300 second output rate: [...], 538950 packets/sec, 45.04% utilization

#show interfaces ethernet 1/1/32 | include output rate
300 second output rate: [...], 658717 packets/sec, 55.05% utilization
```

R1 and R2 should receive half the traffic from R3.

R1

```
#show interfaces ethernet 1/1/14 | include output rate
300 second output rate: [...], 542151 packets/sec, 45.27% utilization
```

R2

```
#show interfaces ethernet 1/1/24 | include output rate
300 second output rate: [...], 657719 packets/sec, 54.97% utilization
```

From the outputs it can be seen that the load-balancing is correct, although the load is not perfectly split. One thing to note is that R4 is frozen, seems like dropping 10Gbps is very CPU intensive.

Traffic to other cluster

In this test there are 20 flows with different source IP and different IP destination. It simulates 20 servers connected to R3 sending traffic to 20 servers in the second cluster. For some unexplained reason, in ICX routers the option *load-balancing symmetric* must be enabled for correct load-balancing in the spines.

Again, R3 should split the traffic to both Spines.

R3

```
#show interfaces ethernet 1/1/31 | include output rate
300 second output rate: [...], 478701 packets/sec, 39.99% utilization

#show interfaces ethernet 1/1/32 | include output rate
300 second output rate: [...], 717992 packets/sec, 59.98% utilization
```

R1 and R2 should split their received traffic another time.

R1

```
#show interfaces ethernet 1/1/11 | include output rate
300 second output rate: [...], 239224 packets/sec, 19.98% utilization

#show interfaces ethernet 1/1/12 | include output rate
300 second output rate: [...], 239224 packets/sec, 19.98% utilization
```

R2

```
#show interfaces ethernet 1/1/21 | include output rate
300 second output rate: [...], 298353 packets/sec, 24.92% utilization

#show interfaces ethernet 1/1/22 | include output rate
300 second output rate: [...], 419481 packets/sec, 35.03% utilization
```

Because the spines of the other cluster are virtual, it's not possible to see the load balancing from SS to virtual Spines.

Again, we can see that the load-balancing is not perfect but good.

11.4 Convergence time

The goal is to see how the convergence time changes when all paths are available and when Red paths are filtered.

For the purposes of this test, the convergence time is defined as the time between the instant a BGP session goes down or up, and the instant the BGP table has all routes expected and has selected the best routes in IPv4 and IPv6. One thing to note is that most of the convergence time is due to the BGP session getting established and not due to the learning process.

To get the timestamps, the debug mode for BGP has been enabled in all routers. Example of debug output:

```
Debug: May 31 11:54:44 BGP: Interface 14337 went Down
Debug: May 31 11:54:44 BGP: 10.90.128.1 stop peer, subcode 8
Debug: May 31 11:54:44 BGP: 10.90.128.1 sending NOTIFICATION Cease
Debug: May 31 11:54:44 BGP: 10.90.128.1 reset, BGP notification Cease sent
Debug: May 31 11:54:44 BGP: 10.90.128.1 Closing TCP connection 0x2d5359b8
Debug: May 31 11:54:44 BGP: 10.90.128.1 BGP connection closed
Debug: May 31 11:54:44 BGP: 10.90.128.1 Peer went to IDLE [...]
Debug: May 31 11:54:44 BGP: 10.90.128.1 Peer already in IDLE state
Debug: May 31 11:54:44 BGP: Interface 14337 went Down
Debug: May 31 11:54:44 BGP: fd01:1458:306:1::1 stop peer, subcode 8
Debug: May 31 11:54:44 BGP: fd01:1458:306:1::1 sending NOTIFICATION Cease
Debug: May 31 11:54:44 BGP: fd01:1458:306:1::1 reset, [...]
Debug: May 31 11:54:44 BGP: fd01:1458:306:1::1 Closing TCP connection [...]
Debug: May 31 11:54:44 BGP: fd01:1458:306:1::1 BGP connection closed
```

```
Debug: May 31 11:54:44 BGP: fd01:1458:306:1::1 Peer went to IDLE [...]
Debug: May 31 11:54:44 BGP: fd01:1458:306:1::1 Peer already in IDLE state
Debug: May 31 11:54:44 BGP: select best route 10.93.3.3/32 load_share
Debug: May 31 11:54:44 BGP: eligible route 1
Debug: May 31 11:54:44 BGP: 10.90.128.9 Best path 10.93.3.3/32, no change
Debug: May 31 11:54:44 BGP: multipath 10.93.3.3/32 changed
Debug: May 31 11:54:44 BGP: update all paths in ipv4 fwd for 10.93.3.3/32
Debug: May 31 11:54:44 BGP: select best route 10.93.3.2/32 load_share
Debug: May 31 11:54:44 BGP: eligible route 1
Debug: May 31 11:54:44 BGP: 10.90.128.9 Best path 10.93.3.2/32, no change
Debug: May 31 11:54:44 BGP: multipath 10.93.3.2/32 changed
Debug: May 31 11:54:44 BGP: update all paths in ipv4 fwd for 10.93.3.2/32
Debug: May 31 11:54:44 BGP: select best route 10.93.3.1/32 load_share
Debug: May 31 11:54:44 BGP: eligible route 1
Debug: May 31 11:54:44 BGP: 10.90.128.9 Best path 10.93.3.1/32, no change
Debug: May 31 11:54:44 BGP: multipath 10.93.3.1/32 changed
Debug: May 31 11:54:44 BGP: update all paths in ipv4 fwd for 10.93.3.1/32
Debug: May 31 11:54:44 BGP: select best route 10.93.2.255/32 load_share
Debug: May 31 11:54:44 BGP: eligible route 1
Debug: May 31 11:54:44 BGP: 10.90.128.9 Best path 10.93.2.255/32, no change
Debug: May 31 11:54:44 BGP: multipath 10.93.2.255/32 changed
Debug: May 31 11:54:44 BGP: update all paths in ipv4 fwd for 10.93.2.255/32
Debug: May 31 11:54:44 BGP: select best route 10.93.2.254/32 load_share
Debug: May 31 11:54:44 BGP: eligible route 1
Debug: May 31 11:54:44 BGP: 10.90.128.9 Best path 10.93.2.254/32, no change
Debug: May 31 11:54:44 BGP: multipath 10.93.2.254/32 changed
```

The time is calculated in two scenarios: when the link from R3 to R1 goes down and up. Each test has been repeated 3 times. The times are shown in seconds.

All paths available

	Test 1	Test 2	Test 3	Mean
R3	2	2	1	1.66
R1	1	2	2	1.66
R2	1	1	1	1

Table 11.1: Convergence time when the link goes down

	Test 1	Test 2	Test 3	Mean
R3	17	15	17	16.33
R1	12	9	18	13
R2	7	11	12	10

Table 11.2: Convergence time when the link goes up

Red paths not available

	Test 1	Test 2	Test 3	Mean
R3	2	2	1	1.66
R1	1	1	2	1.33
R2	2	2	2	2

Table 11.3: Convergence time when the link goes down (no red paths)

	Test 1	Test 2	Test 3	Mean
R3	18	12	24	18
R1	10	25	12	15.66
R2	18	12	14	14.66

Table 11.4: Convergence time when the link goes up (no red paths)

Mean time comparison

	All paths	No Red paths
R3	1.66	1.66
R1	1.66	1.33
R2	1	2

Table 11.5: Convergence mean time comparison when the link goes down

	All paths	No Red paths
R3	16.33	18
R1	13	15.66
R2	10	14.66

Table 11.6: Convergence mean time comparison when the link goes up

11.5 Summary

We have tested route advertisement, load-balancing and convergence time in a complex IP Fabric.

In the first test we checked again that the number of installed routes is as expected. In addition, we found that the current switches used in the DC have limited resources in terms

of supported BGP peers and number of installed routes. To implement the IP Fabric at the scale of CERN's DC, we will need to buy new switches that support more BGP peers.

In the second test we found that load-balancing is correct.

In the third test we found that convergence isn't slower when allowing Red paths in this scenario (when in the worst case, a router has 5760 routes instead of 1200). Moreover, convergence time seems to be random and unpredictable when tested multiple times in the same conditions. Therefore, filtering Red paths should be done in order to save router memory, but from this test we cannot guarantee that convergence will be faster.

12 EVPN tests

In this chapter we test how EVPN works in a basic IP Fabric. Because Brocade ICX switches don't support EVPN, we'll use Juniper QFX switches. Note that in practice, QFX switches support only 4000 VXLANs and 2000 remote VTEPs. The devices are:

- Spine: Juniper QFX10016 (firmware Junos 17.3R1-S1.5)
- Leaves: Juniper QFX5110-48s-4c (firmware Junos 17.3R1.10)
- Hosts: Spirent emulated servers, physical servers

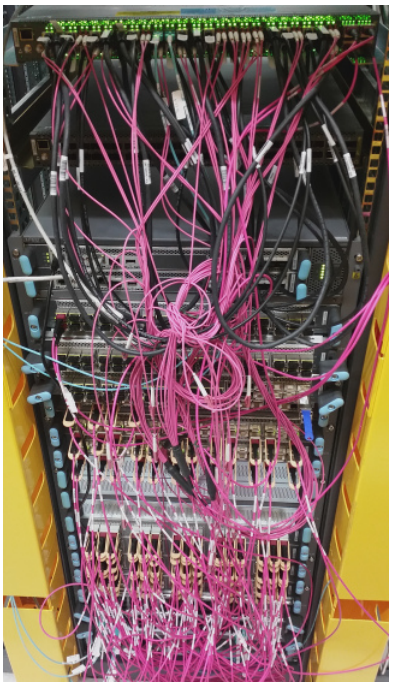


Figure 12.1: Juniper QFX switches during test (most cables are unrelated)

There are two different tests with different setups. In both tests the topology consists of an underlay eBGP IP Fabric and an overlay iBGP EVPN VXLAN. In the overlay, the VTEPs are in the Leaves and the L3 gateway is in the Spine, which also acts as route reflector. In addition, all links are 40GbE.

The goal of the tests is to see how the switches handle multiple overlay flows. There are 3 types of flows with different behaviors (each one has been tested individually to check the description is correct):

- Intra-VXLAN: traffic between two hosts in the same VXLAN over an IP network. If the hosts are connected to different Leaves, the traffic goes through the VXLAN tunnel and the VTEPs in the Leaves encapsulate and decapsulate the packets. If the hosts are connected to the same Leaf, the traffic is switched as normal.

- Inter-VXLAN: traffic between two hosts in different VXLANs. The VTEPs encapsulate the traffic and forward it to the L3 gateway in the Spine, which decapsulates, encapsulates again and forwards to the other VTEP.
- Overlay to Underlay: traffic between one host in a VXLAN, and a host with no VXLAN membership. The VXLAN traffic is encapsulated in the Leaves and decapsulated in the Spine, which forwards it to the other host through the underlay. The normal traffic is encapsulated in the Spine, which forwards it to the VTEP in the Leaves.

12.1 Spirent tests

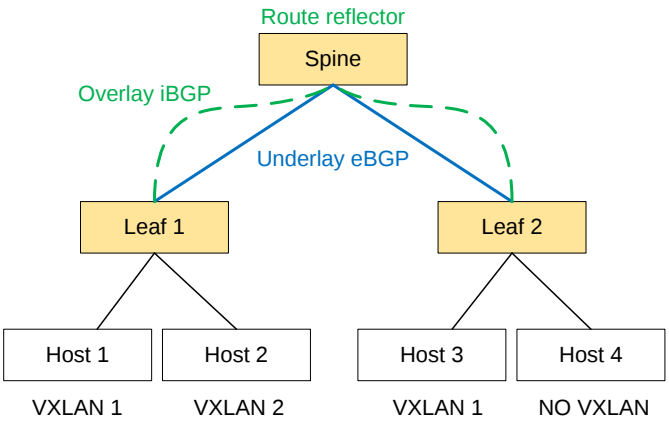


Figure 12.2: Spirent test topology

In this topology there are 4 Spirent emulated servers. The Leaves' ports have been configured with the VXLAN memberships as seen in figure 12.2 (VLANs are untagged). Using the presented setup, each host sends traffic to the other at 10Gbps. There are 6 bidirectional flows:

- Host 1 - Host 2 (inter-VXLAN)
- Host 1 - Host 3 (intra-VXLAN)
- Host 1 - Host 4 (overlay to underlay)
- Host 2 - Host 3 (inter-VXLAN)
- Host 2 - Host 4 (overlay to underlay)
- Host 3 - Host 4 (overlay to underlay)

The goal of this test is to see if the overlay flows can reach line rate. The results are positive.

All flows have been tested independently. All reached line rate and didn't have any losses. Then all flows ran at the same time, also without problems.

	Status	Active	Name	Ta	Index	Controlled By	Source	Destination	Traffic Pattern	Type	Tx Port	Rx Port
▶	●	☑	T1 Host 1 - 2	Cl	0	generator	Host 1 (10.92.100.11/24)	Host 2 (10.92.200.11/24)	Pair	Port	Port //11/1	Port //8/1
	●	☑	T1 Host 2 - 1	Cl	0	generator	Host 2 (10.92.200.11/24)	Host 1 (10.92.100.11/24)	Pair	Port	Port //8/1	Port //11/1
	●	☑	T2 Host 1 - 3	Cl	1	generator	Host 1 (10.92.100.11/24)	Host 3 (10.92.100.12/24)	Pair	Port	Port //11/1	Port //11/2
	●	☑	T2 Host 3 - 1	Cl	0	generator	Host 3 (10.92.100.12/24)	Host 1 (10.92.100.11/24)	Pair	Port	Port //11/2	Port //11/1
	●	☑	T3 Host 1 - 4	Cl	2	generator	Host 1 (10.92.100.11/24)	Host 4 (10.92.150.12/24)	Pair	Port	Port //11/1	Port //8/2
	●	☑	T3 Host 4 - 1	Cl	0	generator	Host 4 (10.92.150.12/24)	Host 1 (10.92.100.11/24)	Pair	Port	Port //8/2	Port //11/1
	●	☑	T4 Host 2 - 3	Cl	1	generator	Host 2 (10.92.200.11/24)	Host 3 (10.92.100.12/24)	Pair	Port	Port //8/1	Port //11/2
	●	☑	T4 Host 3 - 2	Cl	1	generator	Host 3 (10.92.100.12/24)	Host 2 (10.92.200.11/24)	Pair	Port	Port //11/2	Port //8/1
	●	☑	T5 Host 2 - 4	Cl	2	generator	Host 2 (10.92.200.11/24)	Host 4 (10.92.150.12/24)	Pair	Port	Port //8/1	Port //8/2
	●	☑	T5 Host 4 - 2	Cl	2	generator	Host 4 (10.92.150.12/24)	Host 2 (10.92.200.11/24)	Pair	Port	Port //8/2	Port //8/1
	●	☑	T6 Host 3 - 4	Cl	2	generator	Host 3 (10.92.100.12/24)	Host 4 (10.92.150.12/24)	Pair	Port	Port //11/2	Port //8/2
	●	☑	T6 Host 4 - 3	Cl	1	generator	Host 4 (10.92.150.12/24)	Host 3 (10.92.100.12/24)	Pair	Port	Port //8/2	Port //11/2

Figure 12.3: Spirent flows

12.2 Netbench tests

Netbench is a network-testing framework developed at CERN, based on commodity servers and Network Interface Cards (NIC), that enables assessing the devices' behavior when handling multiple TCP flows, which closely resembles real-life usage.

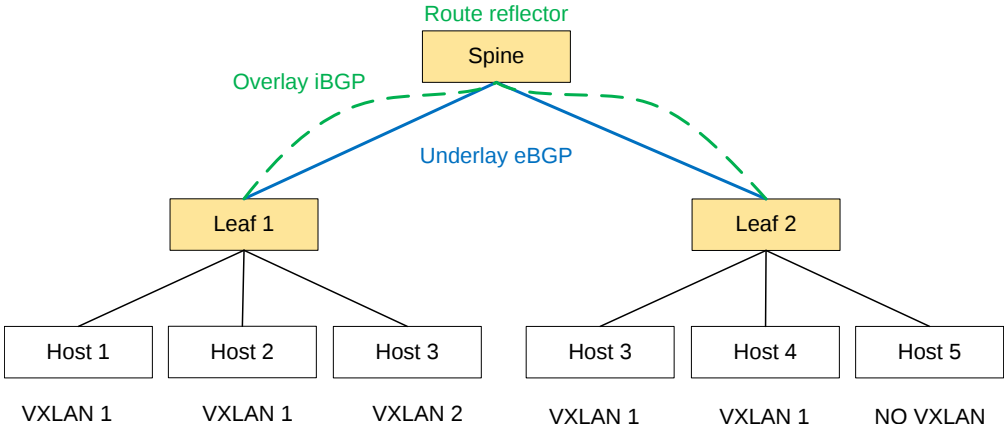


Figure 12.4: Netbench test topology

In this topology there are 6 physical servers running Netbench. The Leaves' ports have been configured with the VXLAN memberships as seen in figure 12.4 (VLANs are untagged). Using the presented setup, each host sends 4 traffic flows to each other. To calculate the total number of flows:

$$\binom{6}{2} \cdot 4 = \frac{6!}{2!(6-2)!} \cdot 4 = 60$$

So in total there are 60 bidirectional flows (24 intra-VXLAN, 16 inter-VXLAN and 20 overlay to underlay). The goal of this test is to see how the overlay TCP flows are balanced. The graphs below show the results. The internal name of the hosts appear in the horizontal axis: nb-n039-01 is Host 1, nb-n039-02 is Host 2, nb-n039-03 is Host 3, nb-n040-01 is Host 4, nb-n040-02 is Host 5, nb-n040-03 is Host 6.

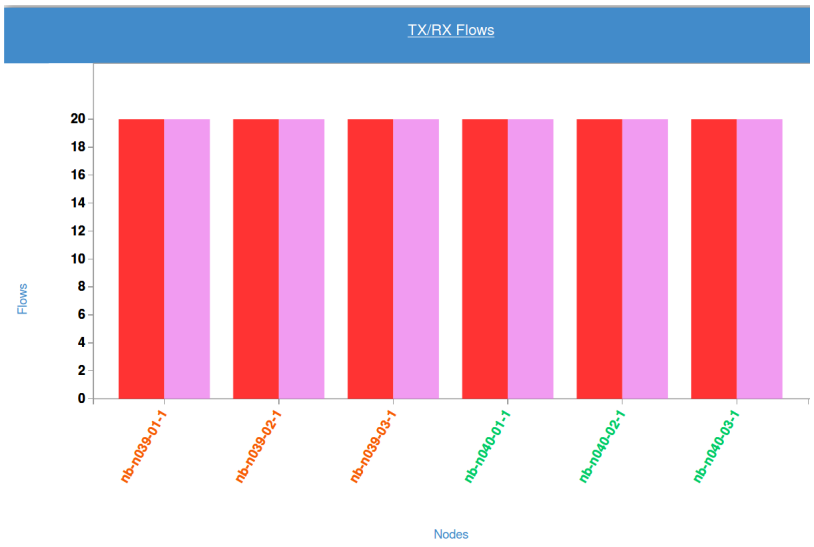


Figure 12.5: Number of flows per node

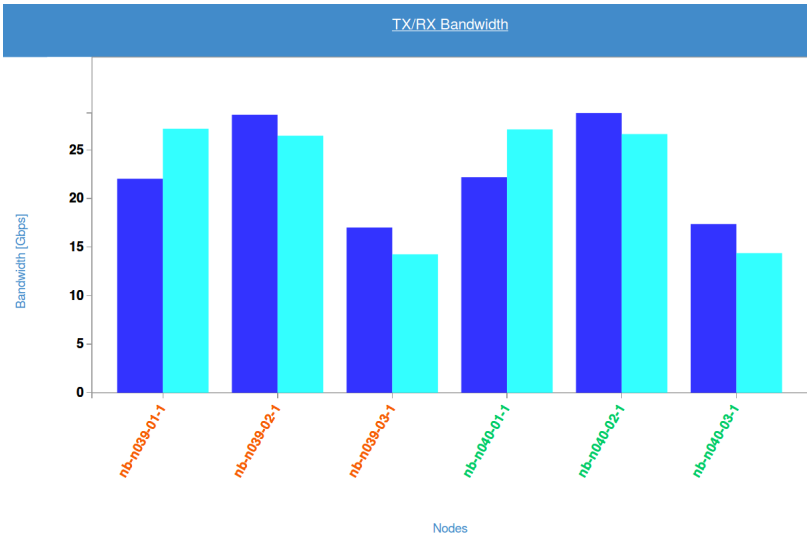


Figure 12.6: Transmission and reception bandwidth per server

Figure 12.6 shows the transmission and reception bandwidth of each server. There are some important things happening in the graph:

- Server bandwidth imbalance: ideally, all servers would have the same bandwidth Normally it's acceptable to have a few small imbalances, however in this case they are significant.
- Tx/Rx imbalance: the transmission and reception bandwidth is not the same in any server.
- Symmetry: the first 3 servers (connected to Leaf 1) have the same patterns as the last 3 servers (connected to Leaf 2). Specifically, the pairs of hosts 1 and 4, 2 and 5, 3 and 6 have the same pattern.

Note that, since all links in the topology are 40GbE, there's a 3:1 oversubscription in the uplinks. This leads to imbalances, depending on which flows traverse these oversubscribed links.

For example, let's focus on the TX traffic from host 3. All traffic will have to cross the leaf-spine uplink. This link is traversed by 52 flows:

- 36: hosts 1,2,3 sending to hosts 4,5,6, multiplied by 4 (number of connections per pair)
- 8: hosts 1,2 sending to host 3, multiplied by 4
- 8: host 3 sending to hosts 1,2, multiplied by 4

Then, the average per-flow bandwidth should be $0.77Gbps$ ($40Gb/52flows$). This matches nicely the result from figure 12.7). The total TX bandwidth from host 3 should be $15.4Gbps$ ($20flows \cdot 0.77Gbps$), which again matches nicely the result from figure 12.6). Therefore, the observed server bandwidth imbalance is not related to crossing a VXLAN tunnel, but to the over-subscription of the leaf-spine uplinks.

Regarding the Tx/Rx imbalance, we doubled the number of flows to see if the balance would improve, but it didn't. We also changed the ports of the servers without success. Therefore, it's not clear if the problem are the switches or the servers. Juniper has stated that the current Junos firmware is not totally stable and has some bugs. A stable release is planned to be released around December 2017.

In addition, hosts don't achieve line-rate. This is most likely due to the fact that 4 TCP flows per pair of servers are not enough to fully drive the NIC to line-rate. Doubling this number would probably allow to fully drive the 40GbE NICs.

In figure 12.7 we can also see that the flows from hosts 1, 2, 4 and 5 have a very high standard deviation. This is because some of these flows are switched locally in the leaves, while others must reach the L3 gateway in the spine. All flows from hosts 3 and 6 must reach the spine.

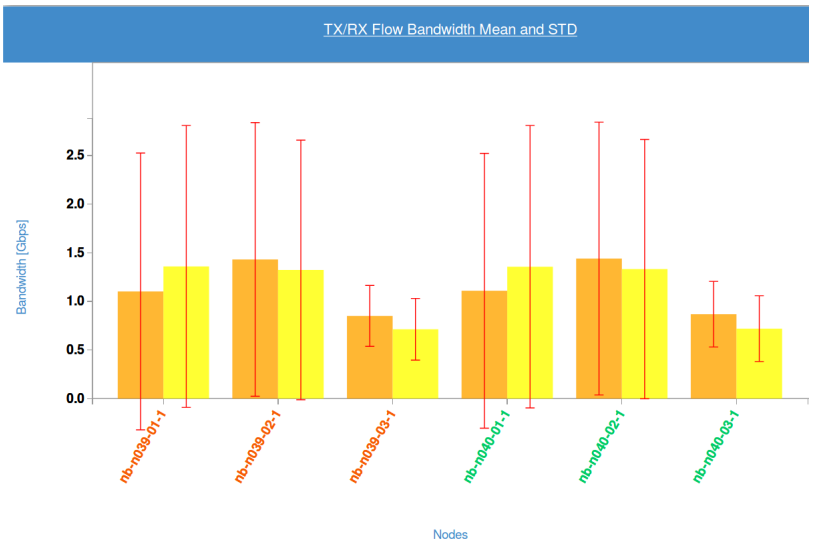


Figure 12.7: Bandwidth mean and standard deviation

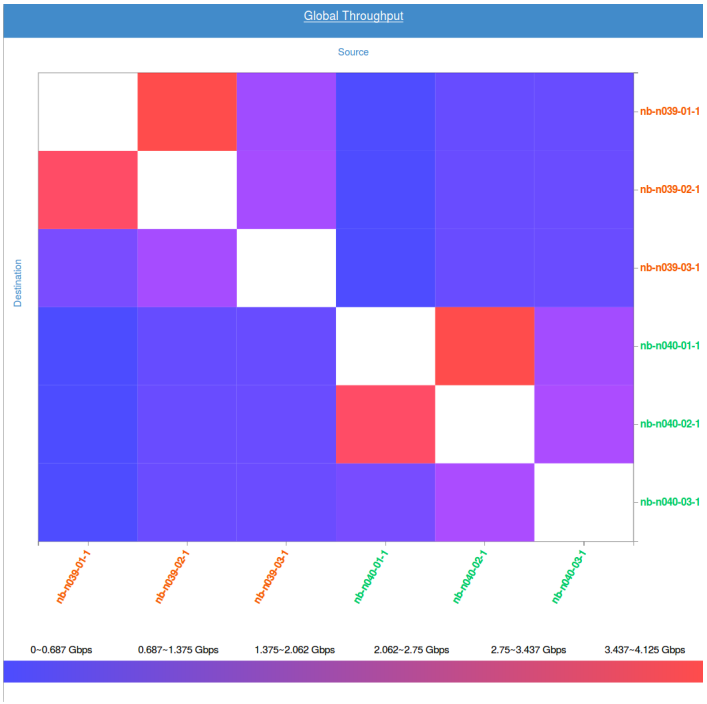


Figure 12.8: Throughput per pair of flows

12.3 Configuration

In this section, the configuration applied to each switch for the Netbench tests is explained. Note that the Juniper configuration language is totally different to Brocade's.

Spine

The Spine has two physical 40GbE interfaces, each one connected to a Leaf. In addition it has two irb interfaces (i.e. L3 virtual interfaces), which act as L3 gateways for each VXLAN. Each has a unique IP address (which is not really used, but JunOS requires it) and a virtual gateway address (which would be shared with other Spines that acted as L3 gateways). Finally, it has two loopback interfaces: one for the physical device and one for the VRF that we will create later.

```
interfaces {
    et-0/0/22 {
        description "to leaf1 et-0/0/48";
        vlan-tagging;
        mtu 9216;
        unit 0 {
            vlan-id 1;
            family inet {
                address 10.92.51.1/24;
            }
        }
    }
    et-0/0/24 {
        description "to leaf2 et-0/0/50";
        vlan-tagging;
        mtu 9216;
        unit 0 {
            vlan-id 1;
            family inet {
                address 10.92.52.1/24;
            }
        }
    }

    irb {
        unit 100 {
            family inet {
                address 10.92.100.211/24 {
                    virtual-gateway-address 10.92.100.1;
                }
            }
        }
        unit 200 {
            family inet {
                address 10.92.200.211/24 {
                    virtual-gateway-address 10.92.200.1;
                }
            }
        }
    }
    lo0 {
```

```
        unit 0 {
            family inet {
                address 10.91.1.1/32;
            }
        }
        unit 1 {
            family inet {
                address 127.10.0.1/32;
            }
        }
    }
}
```

Routing Information Base (RIB) groups can be used to specify the RIBs a routing protocol uses when it is importing and exporting routes. In routing-options we create rib-groups for the purpose of leaking underlay routes to the EVPN VRF so that overlay servers can communicate with underlay servers.

Let's take a look at UNDERLAY-TO-OVERLAY. The import-rib statement indicates that routes from table inet.0 (the default routing table) will be shared with the table cust0100.inet.0 (the routing table of the EVPN VRF we will create later). The import-policy statement indicates that the policy UNDERLAY-FILTER will filter which routes shared.

In routing-options we also specify the AS number for the underlay (which can be also done under protocols bgp). Finally, we use export load-balancing-policy to enable ECMP.

```
routing-options {
    static {
        route 0.0.0.0/0 {
            next-hop 10.16.134.1;
            no-readvertise;
        }
    }
    rib-groups {
        UNDERLAY-TO-OVERLAY {
            import-rib [ inet.0 cust0100.inet.0 ];
            import-policy UNDERLAY-FILTER;
        }
        OVERLAY-TO-UNDERLAY {
            import-rib [ cust0100.inet.0 inet.0 ];
        }
    }
    router-id 10.91.1.1;
    autonomous-system 65000;
    forwarding-table {
        export load-balancing-policy;
    }
}
```

Under protocols we set the configuration for BGP and EVPN. In the underlay we set eBGP: advertise routes based on the policy send-direct, set multipath and set peers. We also need to set the rib-group UNDERLAY-TO-OVERLAY so it has effect. In the overlay we set iBGP: signal EVPN, set the router as RR and set peers. Under evpn, we set the encapsulation and create the VXLAN Network Identifiers (VNI) 100 and 200. The vrf-target statement indicates that the routes which contain the specified Route Target community [45] are identified as belonging to that VNI.

```
protocols {
    bgp {
        group underlay {
            type external;
            family inet {
                unicast {
                    rib-group UNDERLAY-TO-OVERLAY;
                }
            }
            export send-direct;
            multipath;
            neighbor 10.92.51.2 {
                description Leaf1_Underlay;
                peer-as 65001;
            }
            neighbor 10.92.52.2 {
                description Leaf2_Underlay;
                peer-as 65002;
            }
        }
        group evpn {
            type internal;
            local-address 10.91.1.1;
            family evpn {
                signaling;
            }
            cluster 2.2.2.2;
            local-as 65005;
            multipath;
            neighbor 10.92.99.91 {
                description Leaf1_Overlay;
            }
            neighbor 10.92.99.92 {
                description Leaf2_Overlay;
            }
        }
    }
    evpn {
        vni-options {
            vni 100 {
                vrf-target target:10001:1;
            }
```

```
            vni 200 {
                vrf-target target:10001:2;
            }
        }
        encapsulation vxlan;
        multicast-mode ingress-replication;
        extended-vni-list all;
    }

}
```

Next we define the policies and communities. The policy EVPN-VRF-IMPORT is for the EVPN VRF. It just tells to accept the routes from community cust0100. The policy UNDERLAY-FILTER is used for leaking specific networks from the underlay to the overlay. The load-balancing-policy is self-explanatory, and the policy send-direct is the one we used in BGP so Leaves advertise all directly connected networks. cust0100 is the community for the overlay.

```
policy-options {
    policy-statement EVPN-VRF-IMPORT {
        term switch_options_comm {
            from community switch_options_comm;
            then accept;
        }
        term cust0100 {
            from community cust0100;
            then accept;
        }
    }
    policy-statement UNDERLAY-FILTER {
        term 1 {
            from {
                route-filter 10.92.150.0/24 exact;
            }
            then accept;
        }
        term 2 {
            then reject;
        }
    }
    policy-statement load-balancing-policy {
        then {
            load-balance per-packet;
        }
    }
    policy-statement send-direct {
        term 10 {
            from protocol direct;
            then accept;
        }
```

```
    }
    community cust0100 members target:10001:1;
    community switch_options_comm members target:65000:2;
}
```

Here we create the VRF for EVPN: specify the interfaces and set the policies.

```
routing-instances {
    cust0100 {
        instance-type vrf;
        interface irb.100;
        interface irb.200;
        interface lo0.1;
        route-distinguisher 10.91.1.1:2001;
        vrf-import EVPN_VRF_IMPORT;
        vrf-target target:65000:2;
        routing-options {
            interface-routes {
                rib-group inet OVERLAY-TO-UNDERLAY;
            }
        }
    }
}
```

In switch-options, specify that the VTEP interface is the loopback 0.

```
switch-options {
    vtep-source-interface lo0.0;
    route-distinguisher 10.91.1.1:1;
    vrf-import EVPN_VRF_IMPORT;
    vrf-target target:65000:2;
}
```

Here we create the VXLANs and specify the members. In JunOS, each VXLAN must be mapped to a VLAN (but they can have different id). We specify that each irb interface will act as L3 gateway for that VLAN.

```
vlans {
    bd1000 {
        vlan-id 100;
        l3-interface irb.100;
        vxlan {
            vni 100;
            ingress-node-replication;
        }
    }
    bd2000 {
        vlan-id 200;
        l3-interface irb.200;
        vxlan {
            vni 200;
```

```
            ingress-node-replication;
        }
    }
    netbench {
        description "Netbench switching tests";
        vlan-id 3000;
    }
}
```

Leaf 1

The configuration for the Leaves is similar to the Spine's but simpler. Leaf 1 has one interface to the spine and 3 interfaces to hosts 1 to 3.

```
interfaces {
    et-0/0/48 {
        description "to spine et-0/0/22";
        enable;
        vlan-tagging;
        mtu 9216;
        unit 0 {
            vlan-id 1;
            family inet {
                address 10.92.51.2/24;
            }
        }
    }
    et-0/0/49 {
        description "Host 1 VXLAN 100 (39-01)";
        enable;
        unit 0 {
            family ethernet-switching {
                interface-mode access;
                vlan {
                    members 100;
                }
            }
        }
    }
    et-0/0/50 {
        description "Host 2 VXLAN 100 (39-02)";
        enable;
        unit 0 {
            family ethernet-switching {
                interface-mode access;
                vlan {
                    members 100;
                }
```

```
            }
        }
    }
    et-0/0/51 {
        description "Host 3 VXLAN 200 (39-03)";
        enable;
        unit 0 {
            family ethernet-switching {
                interface-mode access;
                vlan {
                    members 200;
                }
            }
        }
    }

    lo0 {
        unit 0 {
            family inet {
                address 10.92.99.91/32;
            }
        }
    }
}
routing-options {
    static {
        route 0.0.0.0/0 {
            next-hop 10.16.134.1;
            no-readvertise;
        }
    }
    router-id 10.92.99.91;
    autonomous-system 65001;
    forwarding-table {
        export load-balancing-policy;
    }
}
protocols {
    bgp {
        group underlay {
            type external;
            export send-direct;
            peer-as 65000;
            multipath;
            neighbor 10.92.51.1 {
                description Spine_Underlay;
            }
        }
        group evpn {
            type internal;
```

```
            local-address 10.92.99.91;
            family evpn {
                signaling;
            }
            local-as 65005;
            multipath;
            neighbor 10.91.1.1 {
                description Spine_Overlay;
            }
        }
    }
    evpn {
        vni-options {
            vni 100 {
                vrf-target target:10001:1;
            }
            vni 200 {
                vrf-target target:10001:2;
            }
        }
        encapsulation vxlan;
        multicast-mode ingress-replication;
        extended-vni-list all;
    }
}
policy-options {
    policy-statement EVPN_VRF_IMPORT {
        term switch_options_comm {
            from community switch_options_comm;
            then accept;
        }
        term cust0100 {
            from community cust0100;
            then accept;
        }
    }
    policy-statement load-balancing-policy {
        then {
            load-balance per-packet;
        }
    }
    policy-statement send-direct {
        term 10 {
            from protocol direct;
            then accept;
        }
    }
    community cust0100 members target:10001:1;
    community switch_options_comm members target:65000:2;
}
```

```
switch-options {
    vtep-source-interface lo0.0;
    route-distinguisher 10.92.99.91:1;
    vrf-import EVPN_VRF_IMPORT;
    vrf-target target:65000:2;
}
vlans {
    bd100 {
        vlan-id 100;
        vxlan {
            vni 100;
            ingress-node-replication;
        }
    }
    bd200 {
        vlan-id 200;
        vxlan {
            vni 200;
            ingress-node-replication;
        }
    }
    default {
        vlan-id 1;
    }
}
```

Leaf 2

Leaf 2 configuration is almost identical to Leaf 1. It has one interface to the spine and 3 interfaces to hosts 4 to 6.

```
interfaces {
    et-0/0/48 {
        description "to spine et-0/0/24";
        enable;
        vlan-tagging;
        mtu 9216;
        unit 0 {
            vlan-id 1;
            family inet {
                address 10.92.52.2/24;
            }
        }
    }
    et-0/0/49 {
        description "Host 4 VXLAN 100 (40-01)";
        enable;
        unit 0 {
```

```
            family ethernet-switching {
                interface-mode access;
                vlan {
                    members 100;
                }
            }
        }
    }
    et-0/0/50 {
        description "Host 5 VXLAN 100 (40-02)";
        enable;
        unit 0 {
            family ethernet-switching {
                interface-mode access;
                vlan {
                    members 100;
                }
            }
        }
    }
    et-0/0/51 {
        description "Host 6 VLAN 99 (40-03)";
        enable;
        unit 0 {
            family ethernet-switching {
                interface-mode access;
                vlan {
                    members 99;
                }
            }
        }
    }
    irb {
        unit 1 {
            family inet {
                address 10.92.150.1/24;
            }
        }
    }
    lo0 {
        unit 0 {
            family inet {
                address 10.92.99.92/32;
            }
        }
    }
}

routing-options {
    static {
```

```
        route 0.0.0.0/0 {
            next-hop 10.16.134.1;
            no-readvertise;
        }
    }
    router-id 10.92.99.92;
    autonomous-system 65002;
    forwarding-table {
        export load-balancing-policy;
    }
}
protocols {
    bgp {
        group underlay {
            type external;
            export send-direct;
            peer-as 65000;
            multipath;
            neighbor 10.92.52.1;
        }
        group evpn {
            type internal;
            local-address 10.92.99.92;
            family evpn {
                signaling;
            }
            local-as 65005;
            multipath;
            neighbor 10.91.1.1 {
                description Spine_Overlay;
            }
        }
    }
    evpn {
        vni-options {
            vni 100 {
                vrf-target target:10001:1;
            }
            vni 200 {
                vrf-target target:10001:2;
            }
        }
        encapsulation vxlan;
        multicast-mode ingress-replication;
        extended-vni-list all;
    }
}
policy-options {
    policy-statement EVPN_VRF_IMPORT {
        term switch_options_comm {
```

```
            from community switch_options_comm;
            then accept;
        }
        term cust0100 {
            from community cust0100;
            then accept;
        }

    }
    policy-statement load-balancing-policy {
        then {
            load-balance per-packet;
        }
    }
    policy-statement send-direct {
        term 10 {
            from protocol direct;
            then accept;
        }
    }
    community cust0100 members target:10001:1;
    community switch_options_comm members target:65000:2;
}
switch-options {
    vtep-source-interface lo0.0;
    route-distinguisher 10.92.99.92:1;
    vrf-import EVPN_VRF_IMPORT;
    vrf-target target:65000:2;
}
vlans {
    bd100 {
        vlan-id 100;
        vxlan {
            vni 100;
            ingress-node-replication;
        }
    }
    bd200 {
        vlan-id 200;
        vxlan {
            vni 200;
            ingress-node-replication;
        }
    }
    default {
        vlan-id 1;
    }
    vlan99 {
        vlan-id 99;
        l3-interface irb.1;
```

```
    }
}
```

12.4 Summary

These tests have been useful to understand the behavior of flows in the EVPN overlay. In particular, the Juniper QFX implementation handles multiple overlay flows quite well, with a few issues that haven't been well understood.

To understand better the Tx/Rx imbalances in the Netbench tests, the tests could be redone doubling the number of flows and converting the uplinks to a 3x40GbE LAG to avoid oversubscription.

13 Conclusion

13.1 Evaluation results

The results of the evaluation are very satisfactory. The IP Fabric has proved to be a good solution to increase the redundancy and provide traffic isolation and VM mobility with EVPN.

In chapter 6 we conclude that eBGP is the most adequate protocol to use for CERN's DC underlay. We have identified necessary and suboptimal paths, the benefits of organizing the routers in clusters and the usefulness of ECMP (one of the most important IP Fabric features). We have discussed the options for managing Point-to-Point links, putting a router on maintenance and minimizing convergence time.

In chapter 7 we have analyzed quantitatively the best-case and worst-case BGP table sizes for each router in the fabric. We have written general mathematical expressions that allow to calculate these sizes depending on the number of advertised routes, the number of routers in each layer of the fabric and cluster, and how they are connected. These numbers allow to verify if the configuration is viable, and if not justify the filtering of certain groups of routes. In an scenario approximate to CERN's DC, we have found that the number of suboptimal paths identified as Red paths is too large. They shouldn't be installed in the BGP tables to save memory and potentially improve convergence time.

In chapter 8 many different ASN distribution policies have been discussed for each layer of the fabric. Some policies require more configuration, extra complexity or explicit filters of unnecessary paths. A combination of policies should be chosen depending on the capabilities of the routers and the preferences of the network engineers.

In chapter 9 we have introduced the need for network virtualization and its advantages. We have discussed EVPN, the state-of-the-art protocol to create overlays in the data center. Unlike in the underlay, iBGP is the best option for an EVPN overlay.

After finishing the theoretical evaluation, in chapter 10 we have presented our architecture proposal for CERN's DC. In chapter 11 we have built test topologies with Brocade switches in the lab in order to verify that the research we have made in previous chapters is correct. The tests comprise route advertisement, load-balancing and convergence time in a complex IP Fabric. We have found that route advertisement and learning works as predicted. The expressions developed in chapter 7 have been confirmed to be correct. Load-balancing in these devices is also working well. The third test has studied the variation in convergence time depending on the filtering of Red paths. The result indicates that convergence isn't slower when allowing Red paths and that convergence time seems to be random and unpredictable when tested multiple times in the same conditions. Therefore, filtering Red paths should be done in order to save router memory, but from this test scenario we cannot conclude that convergence will be faster.

Although the Brocade switches have the necessary features to implement a solid IP Fabric underlay, we have found that their resources are limited in terms of supported BGP peers and number of routes installed. To implement the IP Fabric at the scale of CERN's DC, different switches will be needed, specially if EVPN is desired. The Juniper switches used for the EVPN tests are a good candidate.

Finally, in chapter 12 we have tested an IP Fabric with EVPN VXLAN. We have found that Juniper's QFX implementation handles multiple overlay flows quite well, with a few issues that haven't been well understood.

To conclude, CERN will benefit from implementing the state-of-the-art data center network architecture, *IP Fabric with EVPN VXLAN*. However, better network equipment will need to be installed in order to deploy the solution at the scale of CERN's DC.

13.2 Achievements

This thesis has achieved two goals. First, it is a detailed theoretical and practical evaluation made specifically for CERN's data center, which has been reviewed by network engineers. A working group was created in the IT department in order to collect the requirements from all groups that develop or maintain resources in the DC. The IP Fabric with EVPN complies with the requirements and the architecture change has been approved. This document will be a guide for the staff in the CE section to implement the IP Fabric throughout 2018 and 2019.

Second, it provides new insights of the IP Fabric that are not publicly available in the literature. Chapter 7 shows the importance of calculating the size of BGP tables and provides general expressions to do it. Chapter 8 contains an exhaustive list of different ASN distribution policies and their consequences. Chapter 6 can be read as a detailed introduction and guide for IP Fabric.

On a personal level, this project has allowed me to learn how a real data center works architecturally and operationally. I've been able to work with high-end routers and I have acquired advanced networking concepts which will be useful in my professional career.

13.3 Future work

Port oversubscription hasn't been discussed in this evaluation. Oversubscription depends on the number of ports dedicated to uplinks and their bandwidth capacity. The port density found in today's switches is very high, the market has linecards with 40 100GbE ports. Therefore, for CERN's DC, having oversubscription should be decided based on a future discussion with the teams that provide DC services. Depending on the bandwidth requirements a certain oversubscription may be acceptable – recent discussions aim for a 5:1. For data centers larger than CERN's, such as those companies like Facebook and Google could have,

oversubscription may not be acceptable in any case and thus it cannot be decoupled from the design of the IP Fabric itself.

Future research work which would be of interest would consist in extending the convergence time test done in chapter 11. This would allow a better understanding of the relation between convergence time and the advertisement and learning of a large number of suboptimal paths. Unfortunately, it will be difficult to obtain reliable results for a large-scale data center.

Regarding EVPN, a more detailed study should be carried out, comprising scenarios similar to what CERN needs. This would provide a deeper understanding of the protocol and its limitations.

In addition, more research will be needed to find a router model that has the necessary resources to deploy the IP Fabric at CERN's scale.

Lastly, it must be noted that this evaluation concerns mostly the architectural aspect of the IP Fabric. As a continuation of this work, it is advisable to develop a report on the operational procedures regarding the implementation, upgrade and maintenance of the architecture.

A Appendix: Python script to generate configuration

```
# To edit:
asn = 65301
num_networks = 8

num_routers = 2
offset = 0 # To add more peers after the first time

iface1 = "1/5"
iface2 = "1/6"
##############################################################################
asn = asn + offset
vlan_base1 = 700 + offset
vlan_base2 = 800 + offset
loopback_base = 55
vrfname = "spine"

# To Spine 1
for i in range(num_routers):
    num = vlan_base1 + i
    print("vlan {}".format(num))
    print("tagged ethernet {}".format(iface1))
    print("router-interface ve {}".format(num))
    print("exit")

# To Spine 2
for i in range(num_routers):
    num = vlan_base2 + i
    print("vlan {}".format(num))
    print("tagged ethernet {}".format(iface2))
    print("router-interface ve {}".format(num))
    print("exit")

print("\n\n")

for i in range(num_routers):
    iaux = i + offset
    print("vrf {}{}".format(vrfname, iaux))
    print("rd {0}:{0}".format(10+iaux))
    print("address-family ipv4")
    print("exit-address-family")
    print("address-family ipv6")
    print("exit-address-family")
    print("exit-vrf")
```

```
print("\n\n")

# To Spine 1
for i in range(num_routers):
    num = vlan_base1 + i
    iaux = i + offset
    print("interface ve {}".format(num))
    print("vrf forwarding {}{}".format(vrfname, iaux))
    print("ip address 10.94.{}.2/24".format(iaux))
    print("ipv6 address fd01:1458:306:94{}::2/64".format('{0:02x}'.format(
    iaux)))

# To Spine 2
for i in range(num_routers):
    num = vlan_base2 + i
    iaux = i + offset
    print("interface ve {}".format(num))
    print("vrf forwarding {}{}".format(vrfname, iaux))
    print("ip address 10.95.{}.2/24".format(iaux))
    print("ipv6 address fd01:1458:306:95{}::2/64".format('{0:02x}'.format(
    iaux)))

print("\n\n")

print("router bgp")
current_host = 1 + offset*num_networks%254
current_byte = 1 + offset*num_networks//254
current_host_v6 = 1 + offset*num_networks
for i in range(num_routers):
    iaux = i + offset
    print("address-family ipv4 unicast vrf {}{}".format(vrfname, iaux))
    print("local-as {}".format(asn+i))
    print("neighbor 10.94.{}.1 remote-as {}".format(iaux, 65001))
    print("neighbor 10.95.{}.1 remote-as {}".format(iaux, 65002))
    for j in range(num_networks):
        print("network 10.93.{}.{}/32".format(current_byte, current_host))
        current_host += 1
        if current_host == 256:
            current_byte += 1
            current_host = 1
    print("exit-address-family")

    print("address-family ipv6 unicast vrf {}{}".format(vrfname,iaux))
    print("neighbor fd01:1458:306:94{}::1 remote-as {}".format('{0:02x}'.
    format(iaux), 65001))
    print("neighbor fd01:1458:306:94{}::1 activate".format('{0:02x}'.
    format(iaux)))
    print("neighbor fd01:1458:306:95{}::1 remote-as {}".format('{0:02x}'.
    format(iaux), 65002))
```

```
    print("neighbor fd01:1458:306:95{}::1 activate".format('{0:02x}'.
    format(iaux)))
    for j in range(num_networks):
        print("network fd01:1458:306:4{}::5/64".format('{0:03x}'.format(
    current_host_v6)))
        current_host_v6 += 1

    print("exit-address-family")
print("exit")

print("\n\n")

current_host = 1 + offset*num_networks%254
current_byte = 1 + offset*num_networks//254
current_host_v6 = 1 + offset*num_networks
for i in range(num_routers):
    iaux = i + offset
    print("interface loopback {}".format(loopback_base + iaux))
    print("vrf forwarding {}{}".format(vrfname,iaux))
    for j in range(num_networks):
        print("ip address 10.93.{}.{}/32".format(current_byte,
    current_host))
        current_host += 1
        if current_host == 256:
            current_byte += 1
            current_host = 1
        print("ipv6 address fd01:1458:306:4{}::5/64".format('{0:03x}'.
    format(current_host_v6)))
        current_host_v6 += 1
    print("exit")
```

Bibliography

[1] *About CERN*. URL: https://home.cern/about.

[2] *CERN computing*. URL: https://home.cern/about/computing.

[3] *CERN Data Centre passes the 200-petabyte milestone*. URL: https://home.cern/about/updates/2017/07/cern-data-centre-passes-200-petabyte-milestone.

[4] *About the IT department*. URL: http://information-technology.web.cern.ch/about.

[5] P. Lapukhov, A. Premji, and J. Mitchell. *Use of BGP for Routing in Large-Scale Data Centers*. RFC 7938. RFC Editor, Aug. 2016.

[6] *Routing Design for Large Scale Data Centers*. URL: https://www.nanog.org/meetings/nanog55/presentations/Monday/Lapukhov.pdf.

[7] *Introducing data center fabric, the next-generation Facebook data center network*. URL: https://code.facebook.com/posts/360346274145943/introducing-data-center-fabric-the-next-generation-facebook-data-center-network/.

[8] *Cisco Data Center Spine-and-Leaf Architecture*. URL: https://www.cisco.com/c/en/us/products/collateral/switches/nexus-7000-series-switches/white-paper-c11-737022.html.

[9] *Clos IP Fabrics with QFX5100 Switches*. URL: https://www.juniper.net/assets/fr/fr/local/pdf/whitepapers/2000565-en.pdf.

[10] *Brocade IP Fabric Architecture*. URL: http://www.brocade.com/content/html/en/brocade-validated-design/brocade-ip-fabric-architecture-bvd/GUID-9CE13482-291E-4500-AAFE-33583B69B0D6.html.

[11] *Network Virtualization in IP Fabric with BGP EVPN*. URL: http://www.brocade.com/content/dam/common/documents/content-types/brocade-validated-design/brocade-ip-fabric-bvd.pdf.

[12] *Clos Networks: What's Old Is New Again*. URL: https://www.networkworld.com/article/2226122/cisco-subnet/clos-networks--what-s-old-is-new-again.html.

[13] *Understanding Border Gateway Protocol*. URL: https://www.juniper.net/documentation/en_US/junos/topics/concept/bgp-routing-overview.html.

[14] *Open Shortest Path First*. URL: https://www.cisco.com/c/en/us/products/ios-nx-os-software/open-shortest-path-first-ospf/index.html.

[15] *Understanding Equal Cost Multi Path*. URL: https://www.juniper.net/documentation/en_US/junos/topics/concept/routing-policy-security-ecmp-flow-based-forwarding-understanding.html.

[16] *Understanding Aggregated Ethernet Interfaces and LACP*. URL: https://www.juniper.net/documentation/en_US/junos/topics/concept/interfaces-lag-overview.html.

[17] *Bidirectional Forwarding Detection*. URL: https://www.cisco.com/c/en/us/td/docs/ios/12_0s/feature/guide/fs_bfd.html.

[18] *Brocade SLX switch*. URL: https://www.brocade.com/content/dam/common/documents/content-types/datasheet/brocade-slx-9540-switch-ds.pdf.

[19] *Brocade ICX switch*. URL: http://www.brocade.com/en/products-services/switches/campus-network-switches/icx-7750-switch.html.

[20] *Brocade MLX switch*. URL: http://www.brocade.com/en/products-services/routers/mlx-series.html.

[21] *Spirent TestCenter*. URL: https://www.spirent.com/~/media/Datasheets/Broadband/PAB/SpirentTestCenter/SPT-N11U_Mainframe_Chassis_Datasheet.pdf.

[22] *Brocade SLX price*. URL: https://www.connection.com/product/brocade-slx-9540-48s-48-port-10gbe-switch-w-ac-front-to-back-air-6x100gbe-40gbe/br-slx-9540-48s-ac-f/33527010.

[23] *Brocade ICX price*. URL: https://www.cdw.com/shop/products/Brocade-ICX-7750-48F-switch-48-ports-managed-rack-mountable/3205248.aspx.

[24] *Brocade MLX price*. URL: http://www.dataswitchworks.com/MLX-16.asp.

[25] *Spirent price*. URL: http://www.smartechconsulting.com/SPIRENT-TESTCENTER-HYPERMETRICS-NEXT-SERIES-11U-CHASSIS/.

[26] *Network Engineer salary in Switzerland*. URL: https://www.payscale.com/research/CH/Job=Network_Engineer/Salary.

[27] *Software Engineer salary in Switzerland*. URL: https://www.payscale.com/research/CH/Job=Software_Engineer/Salary.

[28] *Switzerland energy statistics*. URL: http://energymarketprice.com/products/access_SwitzerlandEnergyStatistics.pdf.

[29] *Brocade SLX technical specifications*. URL: http://www.brocade.com/content/html/en/hardware-installation-guide/slxr-9540-installguide/GUID-4CD87DB2-DDE5-4C26-957A-248156B3083B.html.

[30] *Brocade ICX technical specifications*. URL: http://www.brocade.com/content/html/en/technical-specification/fastiron-icx7750-technicalspecification/GUID-E303B307-B13B-4B2C-A176-EA2158FAC83F.html.

[31] *Brocade MLX technical specifications*. URL: http://www.brocade.com/content/html/en/hardware-installation-guide/netiron-05900a-mlxeinstallguide/GUID-2C2921BB-5D39-43A2-B443-EEAADBBA1DB7.html.

[32] *BGP Best Path Selection Algorithm*. URL: https://www.cisco.com/c/en/us/support/docs/ip/border-gateway-protocol-bgp/13753-25.html.

[33] T. Bates, E. Chen, and R. Chandra. *BGP Route Reflection: An Alternative to Full Mesh Internal BGP (IBGP)*. RFC 4456. RFC Editor, Apr. 2006.

[34] D. Walton et al. *Advertisement of Multiple Paths in BGP*. RFC 7911. RFC Editor, July 2016.

[35] A. Retana et al. *Using 31-Bit Prefixes on IPv4 Point-to-Point Links*. RFC 3021. RFC Editor, Dec. 2000.

[36] A. Kirkham. *Issues with Private IP Addressing in the Internet*. RFC 6752. RFC Editor, Sept. 2012.

[37] A. Lindem et al. *Support of Address Families in OSPFv3*. RFC 5838. RFC Editor, Apr. 2010.

[38] D. Katz and D. Ward. *Bidirectional Forwarding Detection (BFD)*. RFC 5880. RFC Editor, June 2010.

[39] M. Bhatia et al. *Bidirectional Forwarding Detection (BFD) on Link Aggregation Group (LAG) Interfaces*. RFC 7130. RFC Editor, Feb. 2014.

[40] C. Villamizar, R. Chandra, and R. Govindan. *BGP Route Flap Damping*. RFC 2439. RFC Editor, Nov. 1998.

[41] C. Pelsser et al. *Making Route Flap Damping Usable*. RFC 7196. RFC Editor, May 2014.

[42] J. Mitchell. *Autonomous System (AS) Reservation for Private Use*. BCP 6. RFC Editor, July 2013.

[43] G. Huston and G. Michaelson. *Textual Representation of Autonomous System (AS) Numbers*. RFC 5396. RFC Editor, Dec. 2008.

[44] T. Narten et al. *Problem Statement: Overlays for Network Virtualization*. RFC 7364. RFC Editor, Oct. 2014.

[45] S. Sangli, D. Tappan, and Y. Rekhter. *BGP Extended Communities Attribute*. RFC 4360. RFC Editor, Feb. 2006.