VFTGen: a Tool to Perform Experiments in Virtual Fat Tree Topologies

Tommaso Caiazzi*†, Mariano Scazzariello*, Lorenzo Ariemma*
*Roma Tre University – Rome, Italy
†NAMEX, ROMA Internet eXchange Point – Rome, Italy

***Abstract*—Data centers are a critical part of the Internet infrastructure. In fact, most of the relevant online services are hosted in a data center. Data center networks are complex, since they are characterized by a high density architecture and by a high level of redundancy. Fat tree topologies are currently the most used in hyperscale data centers. Performing tests in such topologies would be unfeasible, because of the high costs of the required equipment and due to the involvement of human resources. This would limit the automation and reproducibility of tests, leading to a more error-prone testing pipeline. This paper presents VFTGen, a tool that, leveraging on the virtualization and the Software Defined Data Center concepts, automatically builds, deploys and configures arbitrary fat tree topologies in a virtual environment. We demonstrate the ease of use of the tool and its value as a support to the study or the development of networking protocols for fat trees.**

***Index Terms*—Network Virtualization, Data Center, Fat Tree.**

I. Introduction

In the latest years, cloud computing lead to a transition from a capital expenditure (CAPEX) to an operating expense (OPEX) model. This caused a rapid growth and expansion of new digital services hosted in data centers. For this reason, data centers became a critical part of the Internet infrastructure and their performance, security and reliability are key qualities for the companies' businesses.

Such qualities heavily depend on the network architecture and on the protocol used to design and manage the data center infrastructure. There are two high-level choices for building a data center network: to use specialized, but more expensive, proprietary hardware with high bandwidth, or to rely on commodity Ethernet switches and routers to connect data center nodes. The price difference between commodity and proprietary hardware provides a solid motivation to build the network infrastructure of a data center using cheap switches.

In hyperscale data centers, cheap switches are typically connected in a *fat tree topology*, a hierarchical network topology that is highly scalable and redundant. Fat trees can rely on several routing solutions. Some of them are at a very mature stage and/or are variations of general-purpose routing protocols (BGP [1], Openfabric [2]). Others have been developed to work specifically on such topologies (RIFT, Routing In Fat Trees [3]).

Testing and comparing such routing solutions in a physical data center is unfeasible due to the high costs and complexity of the network. However, such tests can be performed in a virtual environment, leveraging on virtualization and the concept of *Software Defined Data Center (SDDC)* [4], [5].

978-3-903176-32-4 © 2021 IFIP

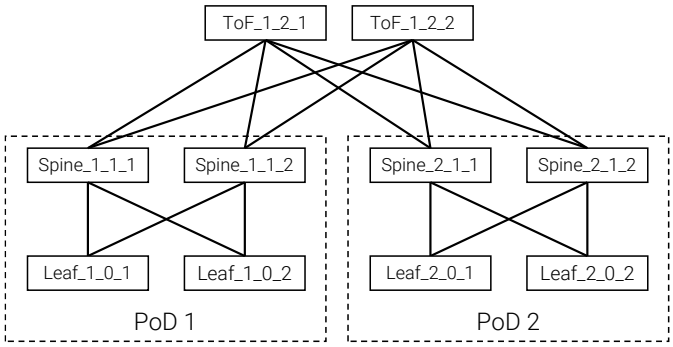

Fig. 1: Single plane fat tree topology $(K = 2, R = 2)$.

In this paper, we present **VFTGen**, a tool that, leveraging on the SDDC concept, automatically builds arbitrary fat tree topologies and configures network devices with different routing protocols. Given the protocol and the topology parameters, the tool creates all the configuration files and folders needed to deploy the desired network scenario in Kathará [6]–[8], a network emulation system. We demonstrate the ease of use of the tool and its value as a support to the study or the development of networking protocols for fat trees.

II. Fat Tree Data Center Topologies

A *fat tree* is a special type of Clos network [9]. Typically, fat tree architectures are composed of three or more levels of switches [10], [11]. A three-level design is composed by an *aggregation layer* that links together different *PoDs* (*Point-of-Delivery*) of two level of switches.

A PoD is a set of *Leaves* (switches at level 0) that are directly connected to the server farm, plus a set of *Spines* (switches at level 1) fully interconnected to the Leaves. For the sake of simplicity, we assume that each switch has $2K$ ports, with K ports pointing north and the other K pointing south. We denote by K_{LEAF} the number of Leaves' ports pointing north or south, and by K_{TOP} the number of Spines' ports pointing north or south. The aggregation layer is composed by *ToF* (*Top of Fabric*, level 2) switches that provide inter-PoD communication. A ToF is connected to at least one Spine per PoD. Depending on the number of links between a ToF and a PoD (redundancy factor R), there are two types of fat trees: single plane and multi-plane.

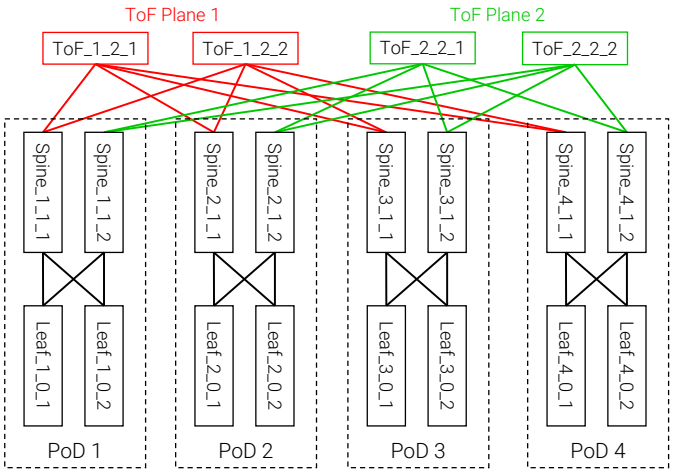

Fig. 2: Multi-plane fat tree topology $(K = 2, R = 1)$.

In a *single plane* topology, depicted in Fig. 1, each ToF is connected to all the top Spines of each PoD. This topology has the maximum value of redundancy factor, with $R = K_{TOP}$.

In a *multi-plane* topology, ToFs are partitioned into $N = K_{TOP}/R$ sets (R being a divisor of K_{TOP}), each with the same number of nodes, called *planes*. The top Spines of a PoD are partitioned into N sets of R nodes. All the Spines of the same set are connected to all the ToFs of the same plane, and all the ToFs of the same plane are connected to the same set of Spines of each PoD. In this way, the redundancy of the network is reduced in order to increase the maximum number of supported PoDs. Fig. 2 shows a multi-plane topology.

III. VFTGen Overview

VFTGen is written in Python and exposes a set of parameters which allows to configure the desired fat tree topology and routing protocol in a very simple way. It takes several input parameters, based on the ones in Sec. II, giving the possibility to specify them in a configuration file or using the command line interface. Mandatory parameters are: (i) K_{LEAF}; (ii) K_{TOP}; (iii) Redundancy Factor R; (iv) Desired routing protocol; (v) Number of servers connected to each Leaf. Additional parameters are used to specify the number of parallel links between nodes and other protocol-specific requirements.

VFTGen outputs files and folders to directly deploy the network scenario in Kathará. This enables to easily run complex tests on virtual topologies, using virtual devices that act like physical ones. Kathará is chosen since it supports Kubernetes for running virtual scenarios, allowing the deployment of arbitrary large topologies.

Currently, the tool supports three protocols: BGP, Openfabric (both using FRRouting [12] implementation), and RIFT (using RIFT-Python [13]). BGP and Openfabric leverage on state-of-the-art configurations for data centers [14], [15]. However, it is easy to customize the protocol configuration or to add new protocols.

The source code of VFTGen is available at [16].

IV. Demonstration

We will demonstrate the ease of use of VFTGen, generating different fat tree topologies of arbitrary complexity by varying the parameters configuration. We will deploy the generated topologies with Kathará, showing that virtual devices correctly emulate real data center switches. We will interact with such virtual devices, dumping routing tables, sniffing network traffic, and digging into the protocol control plane.

After that, we will perform an experiment that reproduces a typical data center scenario: a node failure. It will be performed three times, each time with a different protocol, showing how the supported protocols react to the failure.

This is only an example of the value of VFTGen as a support to the study and development of data center technologies.

References

[1] P. Lapukhov, A. Premji, and J. Mitchell, "Use of BGP for Routing in Large-Scale Data Centers," IETF, RFC 7938, Aug. 2016. [Online]. Available: http://tools.ietf.org/rfc/rfc7938.txt

[2] R. White, S. Hegde, and S. Zandi, "IS-IS Optimal Distributed Flooding for Dense Topologies," Internet Engineering Task Force, Internet-Draft draft-white-distoptflood-04, Jul. 2020, work in Progress. [Online]. Available: https://datatracker.ietf.org/doc/html/draft-white-distoptflood-04

[3] T. Przygienda, A. Sharma, P. Thubert, B. Rijsman, and D. Afanasiev, "RIFT: Routing in Fat Trees," Internet Engineering Task Force, Internet-Draft draft-ietf-rift-rift-12, May 2020, work in Progress. [Online]. Available: https://datatracker.ietf.org/doc/html/draft-ietf-rift-rift-12

[4] Fujitsu. (2020) Software-Defined Data Center – Infrastructure for Enterprise Digital Transformation. [Online]. Available: https://sp.ts.fujitsu.com/dmsp/Publications/public/wp-sddc-infrastructure-for-enterprise-digital-transformation-ww-en.pdf

[5] VMWare. (2015) VMware Software-Defined Data Center. [Online]. Available: https://www.vmware.com/content/dam/digitalmarketing/vmware/en/pdf/techpaper/technical-whitepaper-sddc-capabilities-itoutcomes-white-paper.pdf

[6] G. Bonofiglio, V. Iovinella, G. Lospoto, and G. Di Battista, "Kathará: A Container-Based Framework for Implementing Network Function Virtualization and Software Defined Networks," in *Proc. IFIP/IEEE Network Operations and Management Symposium (NOMS 2018)*, Y.-K. Tu, Ed., 2018.

[7] M. Scazzariello, L. Ariemma, and T. Caiazzi, "Kathará: A lightweight network emulation system," in *NOMS 2020 - 2020 IEEE/IFIP Network Operations and Management Symposium*, 2020, pp. 1–2.

[8] M. Scazzariello, L. Ariemma, G. D. Battista, and M. Patrignani, "Megalos: A scalable architecture for the virtualization of network scenarios," in *NOMS 2020 - 2020 IEEE/IFIP Network Operations and Management Symposium*, 2020, pp. 1–7.

[9] C. Clos, "A study of non-blocking switching networks," *The Bell System Technical Journal*, vol. 32, no. 2, pp. 406–424, March 1953.

[10] LinkedIn Engineering. (2016) The LinkedIn Data Center 100G Transformation. [Online]. Available: https://engineering.linkedin.com/blog/2016/03/the-linkedin-data-center-100g-transformation

[11] Facebook. (2014) Introducing data center fabric, the next-generation Facebook data center network. [Online]. Available: https://engineering.fb.com/production-engineering/introducing-data-center-fabric-the-next-generation-facebook-data-center-network/

[12] FRRouting. (2020) Frrouting. [Online]. Available: https://frrouting.org/

[13] B. Rijsman. (2020) RIFT-Python. [Online]. Available: https://github.com/brunorijsman/rift-python

[14] D. G. Dutt, *BGP in the Data Center*. O'Reilly, 2017.

[15] Philip Smith. (2016) ISIS Tutorial. [Online]. Available: https://www.menog.org/presentations/menog-4/MENOG4-ISIS-Tutorial.pdf

[16] VFTGen. [Online]. Available: https://gitlab.com/uniroma3/compunet/networks/data-center-comparison-tools/fat-tree-generator