CS540 - Paper Review Report # XIV

Hailu Belay Kahsay - 20155624

Title: B4: Experience with a Globally-Deployed Software Defined WAN

Author: Sushant Jain, Alok Kumar, Subhasree Mandal, Joon Ong, Leon Poutievski, Arjun Singh, Subbaiah Venkata, Jim Wanderer, Junlan Zhou,Min Zhu, Jonathan Zolla, Urs Hölzle, Stephen Stuart , Amin Vahdat

In the paper it is indicated that WAN links are usually not fully utilized(with only 30-40% average utilization) due to over-provisioning; and such over-provisioning delivers admirable reliability at the very real costs of 2-3x bandwidth over-provisioning and high-end routing gear. Even though Google data center have faced these overheads for building a WAN connecting multiple data centers with substantial bandwidth requirements, they were able to manage the issue due to the fact that Google's data center WAN exhibits a number of unique characteristics. (1) The applications, servers, and the LANs all the way to the edge of the network are controlled. (2) Most bandwidth-intensive applications perform large-scale data copies from one site to another (3) They anticipated no more than a few dozen data center deployments, making central control of bandwidth feasible.

To alleviate these issues, the authors proposed B4, a Soware Defined Networking(WAN) architecture using OpenFlow for Google's data center to data center connectivity that control relatively simple switches built from merchant silicon. B4 has a number of unique characteristics: massive bandwidth requirements deployed to a modest number of sites, elastic trafc demand that seeks to maximize average bandwidth, and full control over the edge servers and network, which enables rate limiting and demand measurement at the edge. B4 has been deployed for more than 3 years, which offers the academic and industrial a valuable and convincing experience of globally-deployed Software Defined WAN. B4 scales tomeet application bandwidth demandsmore eciently than would otherwise be possible, supports rapid deployment and iteration of novel control functionality such as TE, and enables tight integration with end applications for adaptive behavior in response to failures or changing communication patterns.

One of the challenges in B4 design was integration of Openflow based switch with existing protocols to support hybrid network deployments. The authors chose the open source Quagga stack for BGP/ISIS on NCS focsing on core OpenFlow/SDN functionality. They have written a Routing Application Proxy (RAP) as an SDN application, to provide connectivity between Quagga and OF switches for: (i) BGP/ISIS route updates, (ii) routing-protocol packets owing between switches and Quagga, and (iii) interface updates from the switches to Quagga.

The paper also talks about details on the design, implementation and evaluation of B4, using examples. Bottlenecks in bridging protocol packets from the control plane to the data plane and overheads in hardware programming have been identified as an important areas for future work.

Discussion Points:

- Scalability Issue as B4 is implemented at the hardware and software level?
- Is it possible to use B4 with any data center? What are its requirements?
- What transport protocols does B4 use? If it uses TCP, for low-priority, the high packet-drop rate may lead to congestion control, which may slow down the transmit speed?

Title: **Evolve or Die: High-Availability Design Principles Drawn from Google's Network Infrastructure**

Author: Ramesh Govindan, Ina Minei, Mahesh Kallahalla, Bikash Koley, Amin Vahdat

In the paper it is stated that Google runs three qualitatively different types of networks: data center networks, a software-defined WAN called B4 that supports multiple traffic classes and uses centralized traffic engineering, and another global WAN called B2 for user-facing traffic that employs decentralized traffic engineering with an effort to maintain high availability in these networks. But due to Google's scale and heterogeneity, evolution velocity, and management complexity, it is hard to maintain availability in these networks. Despite these challenges, Google delivers some of the highest availability levels in the industry across dozens of individual services. However, they do experience failure events and document these failures in form of post-mortem reports and root cause analysis.

In this paper the authors analyze 100 such reports of unique high impact failures(within two years period) and find that failure rate is comparable across all three networks (data center networks, B2, and B4). These failures occur in data, control and management plane. They also find that 80% of the failures last between 10 mins and 100 mins. Nearly 90% of the failures have high impact i.e. high packet losses, or blackholes to entire data centers or parts thereof. They have found out that when most of these failures happen, a management operation is usually found be in progress in the vicinity. Later they classify these failures on the basis of root causes and find that, for each of data, control and management planes, the failures can be root-caused to a handful of categories like risk assessment failures; lack of consistency between control plane components; device resource overruns; link flaps; incorrectly executed management operation; and so forth.

Finally they discuss about high availability design principles drawn from these failures. These include defense in depth, which is required to detect and react to failures across different layers and planes of the network and can be achieved by containing the failure radius and developing fallback strategies like red button (software controls that let an operator trigger an immediate fallback). Second, fail open, which preserves the data plane when the control plane fails. Third, maintaining consistency across data, control, and management planes can ensure safe network evolution. Fourth, careful risk assessment, testing, and a unified management plane can prevent or avoid failures. Fifth, fast recovery from failures is not possible without high-coverage monitoring systems and techniques for root-cause analysis.

The authors conclude that by applying these principles, together with the idea that the network should be continuously and incrementally evolved, they have managed to increase the availability of their networks even while its scale and complexity has grown many fold. Finally the authors have suggested (as per their experience) that future networks must account for continuous evolution and upgrade as a key part of their availability architecture and design.

Discussion points:

- Availability Vs Cost?