Reflections on a clean slate 4D approach to network control and management

Albert Greenberg
Microsoft
albert@microsoft.com

David A. Maltz
Microsoft
dmaltz@microsoft.com

Jennifer Rexford
Princeton University
jrex@princeton.cs.edu

Geoffrey Xie
Naval Postgraduate School
xie@nps.edu

Jibin Zhan
Conviva
jibin@conviva.com

Hui Zhang
Conviva
hzhang@conviva.com

This article is an editorial note submitted to CCR. It has NOT been peer reviewed.
The authors take full responsibility for this article's technical content. Comments can be posted through CCR Online.

ABSTRACT

It's been 15 years since what we now call Software Defined Network began emerging out of a set of ideas in the networking research community. This editorial note traces how the ideas in one particular paper from that time have evolved and found practical applications.

CCS CONCEPTS

• **Networks → Network design principles**;

KEYWORDS

Software Defined Networking, Clean slate, Direct control

1 INTRODUCTION

At the time of its writing, Roc Guerin's public review for the 4D paper questioned whether the 4D philosophy might find traction when other attempts at centralized control did not. 10 years later, the Test of Time citation for the paper recognizes it for starting a "resurgence of interest in the topic of separated data and control planes to better manage networks that developed into Software Defined Networking (SDN)." The path between those two statements has been an interesting journey in the changing landscape of where and how networks are deployed.

The 4D paper posited that network control and management be designed as a set of logically-centralized network controllers that would assemble a network-wide view of the current state of the network; determine how to use the network resources to meet network-wide objectives; and then send the desired forwarding state to the routers to directly control their forwarding hardware. Since the paper's publication in 2005, these ideas have developed in two primary directions. The first direction is most closely aligned with the system outlined in the paper, and centers on the direct control of network forwarding. This first direction has seen real-life applications in the OpenFlow protocol, Google's intra-data center network architecture, and traffic engineering systems for Wide Area Networks like SWAN and B4. The second direction expanded on the idea of the network-wide objectives maintained by logically centralized network controllers. This second direction became embodied in the network virtualization techniques used by cloud hosting services to offer each tenant a customized set of network policies delivered over a physically shared network infrastructure.

2 DIRECT CONTROL

The philosophy of direct controlling the forwarding state of physical network switches from logically-centralized controllers has had success in major networks. For example, OpenFlow [1] used in Google's networks [2] and traffic engineering systems like SWAN [3] and B4 [4] all embody the concept of direct control. One appeal of this approach is that it converts the problem of reliability into one of distributed systems — thereby enabling all the techniques from that field to be brought to bear. It also shifts the focus from network protocols, where the negotiations required to enable interoperability dramatically slow innovation, to computer science, graph problems, and software engineering, where the best solution from each domain can be rapidly applied and deployed. Examples of this change include how new control planes for End-System Multicast, [5][6] and CDN [7] systems revolutionized video distribution.

The notion of direct-control of the network continues to evolve, with recent work like Contra [8] positing that logically-centralized control is just an abstraction, not a reality. Clearly a controller must be distributed (at least 2-3 nodes) for reliability, security, and scalability reasons. However, the new work proposes that what we really need is central specification of policies (i.e., high-level, network-wide policies). The realization of the control plane for these policies might be distributed (e.g., distributed P4 code synthesized by a compiler from a central specification). Logically-centralized controls computing forwarding state to distribute to the routers is a simple and convenient way to implement a central/network-wide specification, but it is by no means the only way.

3 NETWORK-WIDE OBJECTIVES

The 4D paper was not the first to suggest centralized control of a network, and the public review was right to question why the 4D philosophy might find traction when other attempts at centralized control did not. The answer came from the rise of cloud computing, where massive data centers of physically shared compute and network infrastructure needed to be virtualized so that each tenant could have a network running exactly the policies it wanted. The

new environment of a cloud data center required customization and scale to an extent not previously seen, and meant that physical network devices simply did not have enough resources to track the state needed to handle each tenant individually. This provided a fertile ground for a new approach using servers with enough RAM to track the network-wide state and the network-wide objectives. Combined with the power of flexible virtual switches that could perform any needed transformation on packets, the transformation of the network management and control planes into software services enabled the rapid evolution of new network capabilities without the delays of waiting for new hardware or industry-wide agreement on new routing protocols.

There continues to be great innovation in the direction of transforming data center networking into decision and dissemination planes — software services provide network-wide views and network-wide objectives, and they directly implement the forwarding policies by sending commands to the servers and middle boxes connected to the network routers. As tenants in cloud datacenters demand lower latencies and greater throughputs, we are finding that software virtual switches can no longer keep up. Instead, new solutions must be found that enable the rapid innovation and flexibility of a software virtual switch while providing the deterministic performance of a hardware pipeline. Examples of this work include hardware offloads engines, such as the Azure SmartNIC [9] and the Amazon Nitro, as well as hybrid software/hardware approaches like Google's Hoverboard. [10] An interesting result is that the network and storage systems must now be treated as first-class citizens in server architecture design. No longer can network connectivity be treated as just "a NIC over there on the PCIe bus." The entire server architecture has pivoted to ensure packet transformations and data transfers are efficiently handled. Going further, the demands of cloud tenants are now pushing beyond policy alone and forcing us to understand how QoS is affected by multiple different types of traffic running over the same shared network.

4 DISCOVERY PLANE

While some of the ideas discussed in the 4D paper have developed into wide-spread production systems, one area imagined by the 4D paper is still in the process of ripening: the discovery plane. This area is now getting serious attention, in part because the data-plane targets are starting to be designed with better (and more programmable) telemetry in mind. Among the most promising work in this area are streaming telemetry, which allows switches to push to the network controllers a set of data like that imagined by the discovery plane, and OpenConfig, which creates a language for defining in the schema in which network devices can describe themselves.

5 SUMMARY

Many of the ideas in the 4D paper were present in the networking community long before the publication of the paper, and historical retrospectives [11] help place the work into context. Moreover, the two central ideas of 4D, (a) the separation of control plane and data plane and (b) having logically centralized control plane to work with distributed data plane elements, have proven to have even broader applicability than we anticipated. We came to these principles from the problem space of layer 3 networking, and this idea has been independently invented in the context of other distributed systems. For example, MapReduce introduces centralized master nodes that are separate from the distributed set of computing worker nodes. The idea is used widely in other distributed systems such as CFS, ZooKeeper, and Kafka, to name a few. Whether it is in SDN, C3, or distributed systems such as MapReduce, the "logical centralized" control system is itself a distributed system that can implement powerful algorithms that consider "network-wide" or "system-wide" view and optimize "network-wide" or "system-wide" goals.

In conclusion, the idea of structuring a distributed system intro two subsystems — a locally centralized control subsystem and a set of distributed data-forwarding or computational working nodes — seems to be a general architecture principle of distributed systems, ranging from layer1-3 networks, internet-scale streaming system, CDN, to distributed file and computation systems such as GFS and MapReduce.

REFERENCES

[1] OpenFlow: Enabling innovation in campus networks. *CCR* 38(2), April 2008.
[2] Jupiter Rising: A Decade of Clos Topologies and Centralized Control in Google's Datacenter Network. *SIGCOMM* 2015.
[3] Achieving high utilization with software-driven WAN. *SIGCOMM* 2013.
[4] B4: Experience with a Globally-Deployed Software DeïňĄned WAN. *SIGCOMM* 2013.
[5] A Case for A Coordinated Internet-Scale Control Plane *SIGCOMM* 2012.
[6] C3: An Internet-Scale Control Plane for Video Quality Optimization *NSDI* 2014.
[7] Redesigning CDN-Broker Interactions For Improved Content Delivery. *CoNext* 2017.
[8] Contra: A programmable system for performance-aware routing. *NSDI* 2020.
[9] Azure Accelerated Networking: SmartNICs in the Public Cloud. *NSDI* 2018.
[10] Andromeda: Performance, Isolation, and Velocity at Scale in Cloud Network Virtualization. *NSDI* 2018.
[11] The Road to SDN: An Intellectual History of Programmable Networks *CCR* 44(2), April 2014.