Quantitatively Evaluating (and Optimizing) Software-Defined Networks

Brandon Heller
Stanford University
Stanford, CA, USA
brandonh@stanford.edu

ABSTRACT

Network architectures with decoupled control planes, called software-defined networks (SDNs), are now transitioning from research prototypes to real deployments. While we understand how to build SDNs, we don't know what defines a good one, how to optimize across the set of all possible ones, or how to make tradeoffs within that set. This extended abstract argues for a quantitative approach to evaluating SDNs and describes first results in this direction.

1. INTRODUCTION

Historically, control plane functions in packet networks have been tightly coupled to the data plane. That is, the boxes that decide *where* and *how* to forward packets have also performed the actual packet forwarding. A more recent trend is to decouple the forwarding and control planes. While the details vary, the common change in these software-defined networks (SDNs) is moving the control-plane logic to a set of dedicated control-plane-only boxes — controllers — that each manage one or more simplified packet-forwarding boxes. This trend is highlighted by a range of industry products and academic prototypes: BGP Route Reflectors [1], MPLS Path Computation Elements with Label-Switched Routers [4], enterprise wireless controllers with CAPWAP access points [2], the planes of 4D [6, 10], and OpenFlow controllers and switches [5, 7, 8, 9].

Proponents claim that SDNs simplify control-plane design, improve convergence, lead to closer-to-optimal path choices, and yeild a more flexible, evolvable network. These benefits have led not just to production-intent prototypes [8, 10], but real deployments and products from startups like Nicira and BigSwitch.

We now know why to build these networks, along with possible ways to build them – but we lack a clear definition of a "good one". Hence, we don't know the *best* way to build them, we can't *quantitatively* argue for or against specific design choices, and we don't know how to make the right *tradeoffs* in a real world that dictates engineering compromises.

Today, detractors raise concerns about decision latency, scalability, and availability. If these concerns are not addressed, SDNs might not get adopted. An operator must know that moving to an SDN does not require sacrifices in the metrics they care about, including performance, cost, and reliability. To ensure that years of researcher effort are not wasted, we must quantitatively compare newer decoupled control planes to traditional fully distributed ones, to silence the fears of operators.

This research question is worth studying because we don't know the answer. We might find that these worries are justified, and that an SDN's dependence on a long-distance channel, along with a smaller set of possible control communication paths, results in overly slow event-to-decision update times as well as reduced availability. We might instead find that the worries are unjustified, and that correct decisions made immediately with consistent state not only kill flaps but reduce delays. The ideal outcome from this analysis would not just be comparison methodology, or even optimization methods, but instead, guidelines for the techniques that will yield the best control network, given a specific topology and specific goals.

2. WHY THIS IS HARD

A number of factors complicate the analysis:

Topologies vary: Networks differs in their number of nodes, edges, distance between nodes, and connectivity. Simply obtaining a large set of reliable network graphs is itself a research area.

Finding relevant metrics: What metrics are most relevant to operators? For example, is guaranteeing delay bounds more important than minimizing the average across the set of nodes?

Combining metrics: The right solution is likely to be a combination of metrics. How can we specify a combination of metrics, or a multiple constraints on a "good enough" solution?

Computational complexity: Optimizing every metric we've considered is an NP-Hard problem, including latency, availability, fairness of state distribution, and control channel congestion.

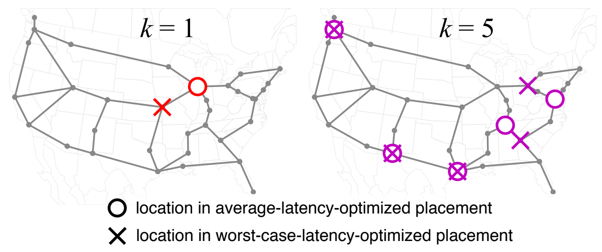
$k = 1$
$k = 5$
location in average-latency-optimized placement
location in worst-case-latency-optimized placement

Figure 1: Optimal placements for 1 and 5 controllers in the Internet2 OS3E deployment.

Design space size: Spreading an application across multiple nodes for scalability and fault tolerance presents many options, including the number of controllers, placement of controllers, number of state replicas, method of distributing processing, and even how many controllers each switch should connect to.

Fortunately, each of these may be addressable. Some might be addressed by repeating an analysis on enough topologies to uncover persistent trends. Others can be addressed by borrowing approximation algorithms from the theory community. The rest could be addressed through simplified models of distributed systems communication.

3. MOTIVATING EXAMPLE: I2

To begin to quantify the concerns, we have started to look at two essential questions:

> *How many controllers are needed, and where in the topology should they go?*

This choice, **controller placement**[1], influences every aspect of a CBA, from state distribution options to fault tolerance to performance metrics. While less significant in the enterprise or data center, propagation latency sets fundamental limits in the wide-area network. Specifically, it bounds the control reactions with a remote controller that can be executed at reasonable speed and stability. For simplicity, we consider only partitioned controllers, whose delays equal the node-to-controller lower bounds, and ignore any delays added by controller-to-controller coordination.

Consider Internet2, which is building a 34-node nation-wide production network [3]. Figure 1 shows placements where the number of controllers, k, equals 1 or 5; the higher density of nodes in the northeast relative to the west leads to metric-specific optimal location combinations. For example, to minimize average latency for $k = 1$, the controller should go in Chicago, which balances the high density of east coast cities with the lower density of cities in the west. To minimize worst-case latency for $k = 1$, the controller should go in Kansas City instead, which is closest to the geographic center of the US. CDFs showing the full set of controller placements, for each value of k, are shown in Figure 2. This example demonstrates that even simple variations of a metric can yield different placements, with their own tradeoffs.

[1] We use "controllers" to refer to geographically distinct controller locations, as opposed to individual servers.

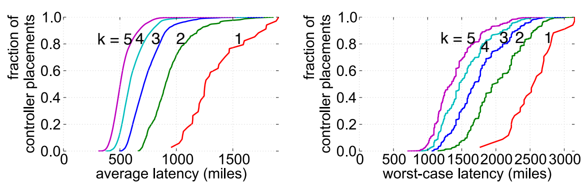

Figure 2: Latency CDFs for all possible controller combinations for $k = [1, 5]$: average latency (left), worst-case latency (right).

4. INITIAL RESULTS

Our current focus is expanding the analysis to many topologies, to find common patterns. Initial insights include: (1) Random placement is a poor strategy. The difference between a random placement and a carefully optimized one is often a factor of 2, and in some cases much larger. (2) Surprisingly, one controller is often enough to meet control response deadlines, such as restoring a link in a SONET ring. (3) Most (75%) of the topologies show tradeoffs between metrics; the graph shows a long tail, with some metrics being off by more than a factor of 2. Going forward, we plan to compare methods of state distribution across topologies, as well as look at metrics that consider fault tolerance. Once we know how to measure and optimize for SDNs, the original goal - a comparison with traditional networks - becomes possible.

5. REFERENCES

[1] BGP Route Reflection: An Alternative to Full Mesh Internal BGP (IBGP). `http://tools.ietf.org/html/rfc4456`.
[2] Control And Provisioning of Wireless Access Points (CAPWAP) Protocol Specification. `http://tools.ietf.org/html/rfc5415`.
[3] Network Development and Deployment Initiative. `http://www.internet2.edu/network/ose/`.
[4] Path Computation Clients (PCC) - Path Computation Element (PCE) Requirements for Point-to-Multipoint MPLS-TE. `http://tools.ietf.org/html/rfc5862`.
[5] M. Casado, M. Freedman, J. Pettit, J. Luo, N. McKeown, and S. Shenker. Ethane: Taking control of the enterprise. *ACM SIGCOMM Computer Communication Review*, 37(4):1–12, 2007.
[6] A. Greenberg, G. Hjalmtysson, and et al. A clean slate 4D approach to network control and management. *ACM SIGCOMM Computer Communication Review*, 35(5):54, 2005.
[7] N. Gude, T. Koponen, J. Pettit, B. Pfaff, M. Casado, and N. McKeown. Nox: Towards an operating system for networks. In *ACM SIGCOMM CCR: Editorial note*, July 2008.
[8] T. Koponen, M. Casado, and et al. Onix: A distributed control platform for large-scale production networks. In *OSDI*. USENIX, 2010.
[9] The openflow switch. `http://www.openflowswitch.org`.
[10] H. Yan, D. Maltz, T. Ng, H. Gogineni, H. Zhang, and Z. Cai. Tesseract: A 4d network control plane. In *Proc. Networked Systems Design and Implementation*, 2007.