The Controller Placement Problem

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

Rob Sherwood
Big Switch Networks
Palo Alto, CA, USA
rob.sherwood@bigswitch.com

Nick McKeown
Stanford University
Stanford, CA, USA
nickm@stanford.edu

ABSTRACT

Network architectures such as Software-Defined Networks (SDNs) move the control logic off packet processing devices and onto external controllers. These network architectures with decoupled control planes open many unanswered questions regarding reliability, scalability, and performance when compared to more traditional purely distributed systems. This paper opens the investigation by focusing on two specific questions: given a topology, how many controllers are needed, and where should they go? To answer these questions, we examine fundamental limits to control plane propagation latency on an upcoming Internet2 production deployment, then expand our scope to over 100 publicly available WAN topologies. As expected, the answers depend on the topology. More surprisingly, *one* controller location is often sufficient to meet existing reaction-time requirements (though certainly not fault tolerance requirements).

Categories and Subject Descriptors

C.2.1 [**Computer-Communication Networks**]: Network Architecture and Design; C.2.3 [**Computer-Communication Networks**]: Network Operations; C.4 [**Performance of Systems**]: Design Studies, Performance Attributes

General Terms

Design, Algorithms, Performance

Keywords

SDN, Software-Defined Networks, OpenFlow, Controller Placement, Latency

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 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 [2], RCP [7], MPLS Path Computation Elements with Label-Switched Routers [5], enterprise wireless controllers with CAPWAP access points [3], the planes of 4D [9, 22], Ethane [8], and in Software-Defined Networks, OpenFlow-based switches and controllers [12, 16, 18].

However, the performance characteristics of these decoupled architectures are largely unstudied, and the debate is not always data-driven. Proponents claim that controller-based architectures simplify control-plane design, improve convergence, and yield a flexible, evolvable network; detractors raise concerns about decision latency, scalability, and availability. To inform this debate and quantify performance concerns, we narrow our focus to two essential questions:

(1) How many controllers are needed?
(2) Where in the topology should they go?

This design choice, the **controller placement problem**, influences every aspect of a decoupled control plane, from state distribution options to fault tolerance to performance metrics. In long-propagation-delay WANs, it places fundamental limits on availability and convergence time. It has practical implications for software design, affecting whether controllers can respond to events in real-time, or whether they must push forwarding actions to forwarding elements in advance. Furthermore, controller placement has immediate relevance: Internet2 is constructing a 34-node SDN [4] and must place the controllers in this production network.

Our contributions are to (1) motivate the controller placement problem and (2) more importantly, quantify the impacts of placement on real topologies. Our goal is *not* to find optimal minimum-latency placements at scale – theorists have already done that, and we can solve it offline anyway – but instead, to present our initial analysis of a fundamental design problem that is worth further study.

After motivating the problem in §2, introducing example users in §3, and defining the metrics in §4, our analysis begins in §5 with an in-depth look at controller placements in the Internet2 OS3E topology, along with their tradeoffs. We then quantify the impacts of controller placement in §6 for over 100 publicly available network topologies from the Internet Topology Zoo [15]. Lastly, in §7 we summarize our findings and describe future extensions to the analysis.

HotSDN'12, August 13, 2012, Helsinki, Finland.

2. CONTROL PLANE DESIGN

Both decoupled and more traditional distributed architectures have a control plane: a network for propagating events, such as routing updates, new traffic engineering policies, or topology changes, to each packet-forwarding device in the network. The key difference between these designs is the structure of their control-plane network. The control topology of traditional distributed architectures like BGP, OSPF, and IS-IS is peer-to-peer: each forwarding device hears events from its peers and makes autonomous decisions based on a local (and likely inconsistent) view of global state.

In contrast, the control networks in decoupled architectures are closer to client-server networks. Packet-forwarding devices (the "clients") have limited decision-making capabilities and must implement control decisions made by controllers (the "servers"). A common example is a BGP route reflector that presents each edge router with a subset of advertised prefixes, rather than propagating the full mesh of learned routes. A more recent example is the relationship between an OpenFlow switch and controller; the switch has *no* local control-plane logic and relies entirely on the controller to populate its forwarding table.[1] Control networks for SDNs may take any form, including a star (a single controller), a hierarchy (a set of controllers connected in a full mesh, which connect to forwarding nodes below), or even a dynamic ring (a set of controllers in a distributed hash table [16]).

Regardless of the exact form, the layout of controllers will affect the network's ability to respond to network events. Understanding *where* to place controllers[2] and *how many* to use is a prerequisite to answering performance and fault tolerance questions for SDNs, and hence also a prerequisite for quantitatively comparing them to traditional architectures. We call this design choice the *controller placement problem*. In this paper, we consider only wide-area networks where the "best" controller placement minimizes propagation delays; in a data center or in the enterprise, one might instead maximize fault tolerance or actively balance controllers among administrative domains.

For WANs, the best placement depends on propagation latency, a quantity fixed by physics and physical topology. Propagation latency bounds the control reactions with a remote controller that can be executed at reasonable speed and stability. With enough delay, real-time tasks (like link-layer fault recovery) become infeasible, while others may slow down unacceptably (BGP convergence). Note that regardless of the state consistency mechanisms in the control plane implementation, these lower bounds apply.

In this paper, we compare placements using node-to-controller latency, for the fundamental limits imposed on reaction delay, fault discovery, and event propagation efficiency. Other metrics matter, such as availability and fairness of state, processing, and bandwidth — but our focus is the WAN, where latency dominates. One can always reduce the effective delay by adding autonomous intelligence into a switch or pushing failover plans, but these may add complexity and make network evolution harder. One goal of this paper is to understand if, and for which networks, extensions to the "dumb, simple switch" model are warranted.

[1] We ignore "bootstrap state" for the control connection.

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

3. MOTIVATING EXAMPLES

Having defined the core problem, we show three types of SDN users and motivate their use for controller placement design guidelines.

Network Operators. Rob Vietzke is the VP of Network Services at Internet2, and his team has committed to a SDN deployment of 34 nodes and about 41 edges, shown in Figure 1. This network, the Open Science, Scholarship and Services Exchange (OS3E) [4], needs to peer with the outside world through BGP. Placement matters to Rob because his network should minimize downtime and multiple controllers are a requirement for high availability.

Controller Application Writers. Nikhil Handigol is a grad student who created Aster*x [13], a distributed load balancer that reactively dispatches requests to servers as well as managing the network path taken by those requests. Nikhil would like to demonstrate the advantages of his algorithm on a service with real users, and ideally on a range of topologies, like GENI. Placement matters to Nikhil because he can't get users if his service goes down or does not perform, but at the same time he would prefer to keep things simple with one controller. Ideally, we could provide Nikhil with guidelines to evaluate the response-time potential of different approaches, from centralized to distributed, before he implements extra code or does a deployment.

Network Management Software Writers. Rob Sherwood built FlowVisor [19], a centralized network slicing tool that enables network access and control to be split among multiple controllers or versions of controllers, given a control policy. Since FlowVisor's only consistent state is its configuration, multiple instances might be used to scale FlowVisor. Placement matters to Rob because FlowVisor sits between controllers and switches, where its presence adds a delay to potentially every network command; this delay should be actively minimized, especially with multiple instances.

In each case, the SDN user must ask the question: "How many controllers should I use, and where should they go?" and benefits from practical methods for analyzing tradeoffs.

4. PLACEMENT METRICS

We now introduce and compare definitions of whole-network latency, along with their corresponding optimization problems. Each is called a facility location problem and appears in many contexts, such as minimizing firefighting response times, locating warehouses near factories, and optimizing the locations of content distribution nodes and proxy servers. All are NP-hard problems with an input for k, the number of controllers to place, and all have weighted variations where nodes have varying importance.

Average-case Latency. For a network graph $G(V, E)$ where edge weights represent propagation latencies, where $d(v, s)$ is the shortest path from node $v \in V$ to $s \in V$, and the number of nodes $n = |V|$, the average propagation latency for a placement of controllers S' is:

$$L_{avg}(S') = \frac{1}{n} \sum_{v \in V} \min_{(s \in S')} d(v, s) \quad (1)$$

In the corresponding optimization problem, *minimum k-median* [6], the goal is to find the placement S' from the set of all possible controller placements S, such that $|S'| = k$ and $L_{avg}(S')$ is minimum. For an overview of the approaches to solving this problem, along with extensions, see [20].

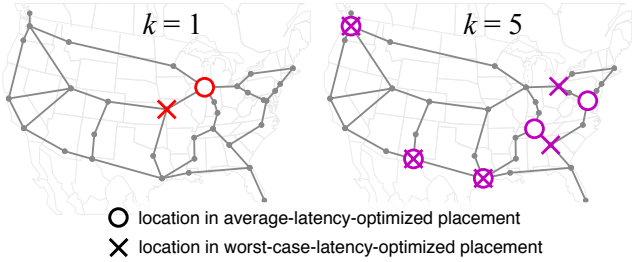


Figure 1: Optimal placements for 1 and 5 controllers in the Internet2 OS3E deployment.

Worst-case latency. An alternative metric is worst-case latency, defined as the maximum node-to-controller propagation delay:

$$L_{wc}(S') = \max_{(v \in V)} \min_{(s \in S')} d(v, s) \quad (2)$$

where again we seek the minimum $S' \subseteq S$. The related optimization problem is *minimum k-center* [21].

Nodes within a latency bound. Rather than minimizing the average or worst case, we might place controllers to maximize the number of nodes within a latency bound; the general version of this problem on arbitrary overlapping sets is called *maximum cover* [14]. An instance of this problem includes a number k and a collection of sets $S = S_1, S_2, ..., S_m$, where $S_i \subseteq v_1, v_2, ..., v_n$. The objective is to find a subset $S' \subseteq S$ of sets, such that $|\bigcup_{S_i \in S'} S_i|$ is maximized and $|S'| = k$. Each set S_i comprises all nodes within a latency bound from a single node.

In the following sections, we compute only average and worst-case latency, because these metrics consider the distance to every node, unlike nodes within a latency bound. Each optimal placement shown in this paper comes from directly measuring the metrics on all possible combinations of controllers. This method ensures accurate results, but at the cost of weeks of CPU time; the complexity is exponential for k, since brute force must enumerate every combination of controllers. To scale the analysis to larger networks or higher k, the facility location problem literature provides options that trade off solution time and quality, from simple greedy strategies (pick the next vertex that best minimizes latency, or pick the vertex farthest away from the current selections) to ones that transform an instance of k-center into other NP-complete problems like independent set, or even ones that use branch-and-bound solvers with Integer Linear Programming. We leave their application to future work.

5. ANALYSIS OF INTERNET2 OS3E

Having defined our metrics, we now ask a series of questions to understand the benefits of multiple controllers for the Internet2 OS3E topology [4]. To provide some intuition for placement considerations, Figure 1 shows optimal placements for $k = 1$ and $k = 5$; the higher density of nodes in the northeast relative to the west leads to a different optimal set of locations for each metric. 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.

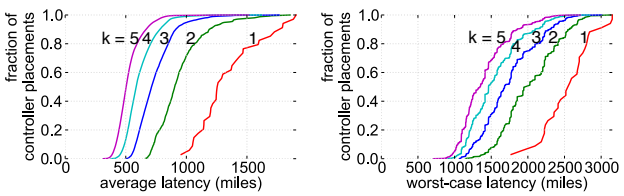


Figure 2: Latency CDFs for all possible controller combinations for $k = [1, 5]$: average latency (left), worst-case latency (right).

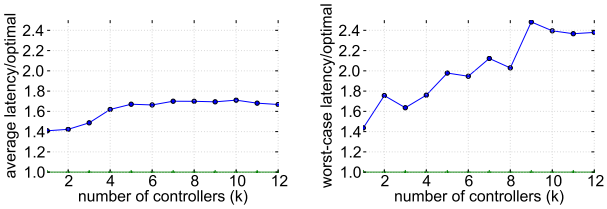


Figure 3: Ratio of random choice to optimal.

5.1 How does placement affect latency?

In this topology, placement quality varies widely. A few placements are pathologically bad, most are mediocre, and only a small percent approach optimal. Figure 2 shows this data as cumulative distributions, covering all possible placements for $k = 1$ to $k = 5$, with optimal placements at the bottom. All graphs in this paper show one-way network distances, with average-optimized values on the left and worst-case-optimized values on the right. If we simply choose a placement at random for a small value of k, the average latency is between 1.4x and 1.7x larger than that of the optimal placement, as seen in Figure 3. This ratio is larger for worst-case latencies; it starts at 1.4x and increases up to 2.5x at $k = 12$. Spending the cycles to optimize a placement is worthwhile.

5.2 How many controllers should we use?

It depends. Reducing the average latency to half that at $k = 1$ requires three controllers, while the same reduction for worst-case latency requires four controllers. Assuming we optimize for one metric, potentially at the expense of the other, where is the point of diminishing returns? Figure 4 shows the benefit-to-cost ratios for a range of controllers, defined as $(lat_1/lat_k)/k$. A ratio of 1.0 implies a proportional reduction; that is, for k controllers, the latency is $1/k$ of

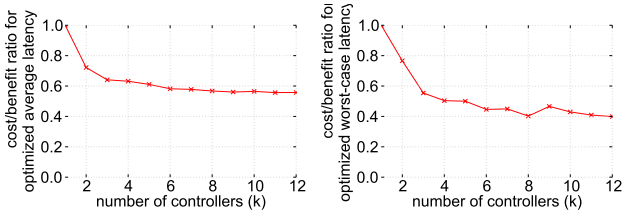
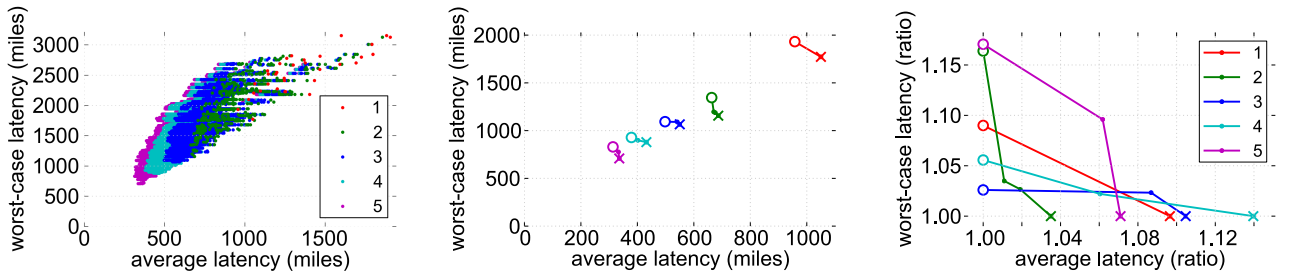


Figure 4: Cost-benefit ratios: a value of 1.0 indicates proportional reduction, where k controllers reduce latency to $\frac{1}{k}$ of the original one-controller latency. Higher is better.



(a) All controller placements (b) Pareto-optimal curves (c) Normalized pareto-optimal curves

Figure 5: Placement tradeoffs for $k = 1$ to $k = 5$; (b) shows the best placements from (a), while (c) normalizes the curves in (b). We see up to a 17% increase in the un-optimized metric.

the single-controller latency. For this topology, each metric stays below 1.0 and shows diminishing returns that level off around 3-4 controllers. Independently, Internet2 operators suggested a $(3 + 1)$-controller setup as a reasonable starting point and expressed interest in having three controllers plus one for fault tolerance.

5.3 What are the tradeoffs?

One must choose between optimizing for worst-case latency or average-case latency. In the OS3E topology, for each input value of k, the optimal placement for each metric comprises a different set of nodes. In some cases, these sets overlap; Figure 1 shows one example, where the choices for $k = 5$ overlap at three locations.

The point cloud in Figure 5(a) shows both latency metrics for all combinations of up to five controllers. We only care about the optimal points in the lower-left region; Figure 5(b) zooms into this region and shows only those points that are on the pareto frontier. Informally, each point on this curve represents either the optimal for one metric, or some mix between the two. To more easily quantify the tradeoffs, Figure 5(c) normalizes the pareto frontiers, where 1.0 represents the optimal metric value for the given k for that axis. Now we can directly evaluate the tradeoff that may be required when choosing one metric over another, which is up to a 17% larger worst-case latency when we optimize for average latency.

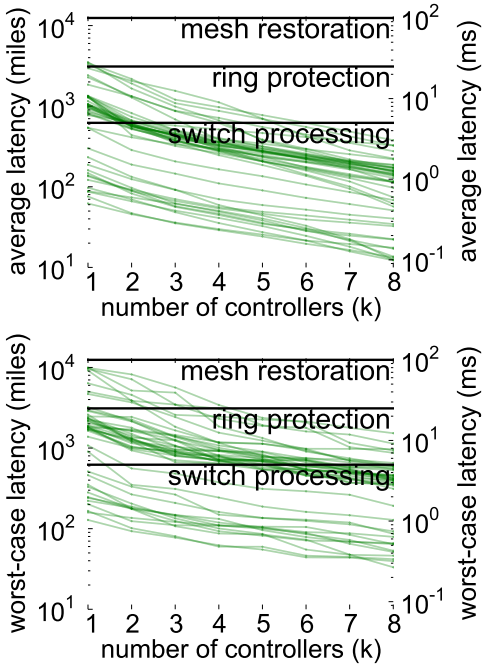


Figure 6: Optimal latencies (log-scaled).

6. ANALYSIS OF MORE TOPOLOGIES

Do the trends and tradeoffs seen for Internet2 apply to other topologies, and what aspects of a topology affect those trends and tradeoffs? In this section, we expand our analysis to hundreds of topologies in the Internet Topology Zoo, a collection of annotated network graphs derived from public network maps [15]. We employ this data set because it covers a diverse range of geographic areas (regional, continental, and global), network sizes (8 to 200 nodes), and topologies (line, ring, hub-and-spoke, tree, and mesh). The graphs in the Zoo do not conform to any single model, which demands a more careful analysis - but in many cases the outliers shed light on why some classes of topologies require more controllers to achieve the same latency reductions.

We include in our analysis most of the 256 topologies in the Zoo. To avoid bias towards maps with multiple versions available, we use the most recent one. A number of network maps, including all those with $n \geq 100$, have ambiguous locations or edges; we ignore these. We also remove the small fraction of nodes in a few topologies that are disconnected from the rest of the graph.

6.1 Is one controller enough?

Surprisingly, one controller location often suffices. We first look in Figure 6 at raw latencies for optimal placements, in miles and milliseconds. Each line represents one topology. Immediately, we are reminded of the variety of topology scales, from regional to global, yet most lines show the same steady downward trend. The thick bundle of lines corresponds to the US- and Europe-sized topologies that make up much of the Topology Zoo.

To get a sense for the magnitude of these fundamental propagation delays, we compare them to bounds relevant to today's networks, such as expected recovery times and delays expected within a forwarding device.[3] The lowest

[3]Scaled by half to align w/one-way latency numbers.

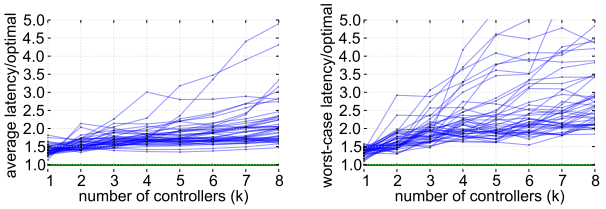


Figure 7: Random-to-optimal ratios.

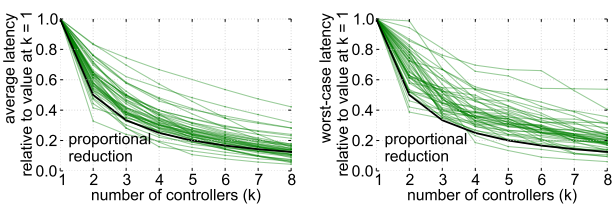


Figure 8: Normalized optimal latencies.

horizontal line is *switch processing:* 10 milliseconds roughly corresponds to the measured delay of today's commercial hardware-based OpenFlow switches, from the time a packet is sent, to the time a response is received, through an unloaded controller directly attached to the switch. *Ring protection:* 50 milliseconds is the target restoration time of a SONET ring, covering the time from fault detection to when traffic is flowing in the opposite direction along the ring. This is a common target, as it prevents circuits from requiring re-establishment, such as those on a voice call. *Shared-mesh restoration*: around 200 to 250 milliseconds is the point at which voice calls start to drop, or ATM circuit rerouting may be triggered [1].

We make no claims that responding to events at these timescales will actually lead to human-perceptible disruption, or whether they still apply. However, they do provide a back-of-the-envelope way to evaluate whether the fundamental latency increases from greater control plane centralization prevent equivalent failure response when compared to today's networks.

Round-trip Latency Target		Safety Margin		
Name	Delay	1.0x	1.5x	2.0x
switch processing	10 ms	27%	22%	18%
ring protection	50 ms	82%	60%	33%
mesh restoration	200 ms	100%	91%	89%

Table 1: Percent of topologies with worst-case round-trip latencies below the target, for *one* controller, which frequently suffices.

Table 1 summarizes the one-controller results, reporting the fraction of topologies for which every control path meets a specified latency bound. This table also considers different safety margins, to help account for forwarding device and controller overheads, as well as processing variability. In 82% of topologies, a single location presents no fundamental limit to meeting SONET ring protection targets. If we use a safety margin of 1.5, leaving half as much time for other overheads and variation as for propagation, 60% of topologies meet the deadline. Availability concerns will always apply, and placing controllers in different locations provides major fault tolerance benefits. However, for many networks, multiple controllers are *not strictly necessary* to meet response-time goals, even with "dumb, simple switches".

6.2 How does placement affect latency?

Random placement is far from optimal, and not just in OS3E. Figure 7 shows a factor of two difference between most random and optimal placements. In almost all topologies, this ratio increases with larger numbers of controllers, and for smaller topologies, it increases relatively faster, due to each controller location having a relatively larger effect. The ratio is also larger (and grows faster) when optimizing for the worst case instead of the average; this is to be expected, since optimizing for the average discounts the effect of the farthest-away nodes.

We do, however, see some high outliers on these graphs, where a random choice might yield a solution with five times worse latency than an optimal one. The top three for both optimization metrics, for which random does poorly, are Highwinds, HurricaneElectric and Xeex. Each of these is a medium-size topology ($n = 18$ to 24) that covers at least three continents; random placements are likely to put the controllers on the same continent, explaining the increase. The low outliers when optimizing for average are Bics and Geant2010, both larger ($n = 33$ to 37) mesh topologies in Europe. These topologies have a highly-connected region in the center, and random placement is likely to put controllers there. For worst-case optimization, the numbers are more variable and no single topology maintains a low ratio.

6.3 How quickly does latency reduce?

For most topologies, adding controllers yields slightly less than proportional reduction. That is, k controllers reduce latency nearly to $\frac{1}{k}$ of the baseline latency with one controller. This was not a given; going from one controller could yield even more than a factor-of-two reduction. Consider the example of a topology with nodes split evenly between two distant islands. The initial choice doesn't matter; either way, the average latency will be half the distance between the islands. However, going to two controllers, one placed on each island, the average and worst case latencies drop significantly.

Figure 8 shows normalized latency reductions, with a line marking proportional reduction ($\frac{1}{2}$ for $k = 2$, $\frac{1}{3}$ for $k = 3$, and so on). The second controller generally drops the latency by slightly less than a factor of two, and the third controller generally drops the latency by a bit less than a factor of three. The lowest lines, where each controller provides a larger-than-proportional benefit, correspond to smaller topologies with $n < 30$. Above the proportional line, the slope of the normalized latency reduction curves is more gradual, indicating a smaller benefit. Looking at data not shown due to space constraints, we find that larger topologies generally require more controllers to see the same fractional reduction in latency. A simple rule like "k controllers ought to be enough for anybody" does not apply.

Two of the three highest outliers are again Bics and Geant2010. For topologies like these, where most nodes are together in one cluster and random placements are quite good, adding controllers provides less of a benefit. With Geant2010, a full seven controllers are required just to drop the latency by a factor of two, both for average and worst-

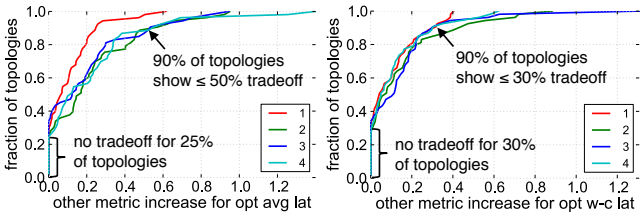


Figure 9: Tradeoffs when optimizing for one parameter instead of another.

case. The other high outlier is Itnet, a hub-and-spoke topology; once the hub is covered, the incremental latency reduction from covering a spoke is small. Low outliers, where the second controller provides a better-than-2 reduction, are all topologies covering multiple islands, either real (Xeex, RedIris) or effective (Arpanet1972: east and west coasts of US).

6.4 What are the tradeoffs?

In three-quarters of topologies, we must trade off one metric for another. When looking at all topologies, we see a range of tradeoffs in the choice of one latency metric over another, as shown in Figure 9. In at least a quarter of topologies, one solution optimizes both metrics, with no tradeoff. When optimizing for average latency, 90% of the time, the worst-case is within 50% of optimal; when optimizing for worst-case latency, 90% of the time, the average latency is within about 30% of optimal. In the other 10%, we see major outliers, including cases where the un-optimized metric is twice the optimal value. Outside of $k = 1$ showing smaller tradeoffs for average-case, we see no k-dependent patterns.

7. DISCUSSION AND CONCLUSION

In this paper, we showed that the answer to *where* and *how many* controllers to deploy depends on desired reaction bounds, metric choice(s), and the network topology itself. Most networks show diminishing returns from each added controller, along with tradeoffs between metrics. Surprisingly, in many medium-size networks, the latency from every node to a *single* controller can meet the response-time goals of existing technologies, such as SONET ring protection (§6.1). Natural extensions for this analysis include:

Availability: Intuition suggests that fully distributed control planes are more resilient to failures than decoupled control planes, but issues of route flapping [17], BGP "wedgies" [10], and protracted route convergence times [11] undermine this belief. On the other hand, decoupled network control planes exchange these concerns for other failure modes such as controller failure or disconnection. One could place controllers to maximize fault tolerance; or, one could minimize the distance to the n^{th}-closest controller, called the α-neighbor k-centers problem, which considers up to $n - 1$ controller failures.

State Distribution: Decoupling potentially enables state-oriented methods such as Distributed Hash Tables and Paxos to replace more traditional message-oriented approaches in wide-area network control planes [16]. One could place controllers not for latency bounds, but to optimize for full event-to-response delays, given a controller communication model.

Controller Selection: Decoupling the data and control planes opens the question of how to maintain a map of forwarding devices to controllers. One could place controllers to minimize latency, as a starting point, then use another algorithm to (possibly dynamically) balance switches among available controllers.

We believe that this exploration of the controller placement problem provides useful guidance for SDN operators and application designers. However, there appear to be no placement rules that apply to *every* network. Whenever an operator wants to add controllers, they should use the methods shown in this paper to determine their own best controller placement; full code to generate and analyze placements is available at `github.com/brandonheller/cpp.git`.

8. ACKNOWLEDGMENTS

This work was supported by an HP Fellowship.

9. REFERENCES

[1] Ansi t1.tr.68-2001 enhanced network survivability performance.
[2] BGP Route Reflection: An Alternative to Full Mesh Internal BGP (IBGP). `http://tools.ietf.org/html/rfc4456`.
[3] Control And Provisioning of Wireless Access Points (CAPWAP) Protocol Specification. `http://tools.ietf.org/html/rfc5415`.
[4] Internet2 open science, scholarship and services exchange. `http://www.internet2.edu/network/ose/`.
[5] Path Computation Clients (PCC) - Path Computation Element (PCE) Requirements for Point-to-Multipoint MPLS-TE. `http://tools.ietf.org/html/rfc5862`.
[6] V. Arya, N. Garg, R. Khandekar, A. Meyerson, K. Munagala, and V. Pandit. Local search heuristics for k-median and facility location problems. *SIAM Journal on Computing*, 33(3):544–562, 2004.
[7] M. Caesar, D. Caldwell, N. Feamster, J. Rexford, A. Shaikh, and J. van der Merwe. Design and implementation of a routing control platform. In *NSDI*. USENIX, 2005.
[8] M. Casado, M. Freedman, J. Pettit, J. Luo, N. McKeown, and S. Shenker. Ethane: Taking control of the enterprise. *ACM SIGCOMM CCR*, 37(4):1–12, 2007.
[9] A. Greenberg, G. Hjalmtysson, and et al. A clean slate 4D approach to network control and management. *ACM SIGCOMM CCR*, 35(5):54, 2005.
[10] T. Griffin and G. Huston. BGP Wedgies. RFC 4264 (Informational), Nov. 2005.
[11] T. G. Griffin and G. Wilfong. An analysis of bgp convergence properties. *SIGCOMM CCR.*, 29:277–288, August 1999.
[12] N. Gude, T. Koponen, J. Pettit, B. Pfaff, M. Casado, and N. McKeown. Nox: Towards an operating system for networks. In *ACM SIGCOMM CCR*, July 2008.
[13] N. Handigol, S. Seetharaman, M. Flajslik, R. Johari, and N. McKeown. Aster*x: Load-balancing as a network primitive. 9th GENI Engineering Conference (Plenary), November 2010.
[14] D. Hochba. Approximation algorithms for np-hard problems. *ACM SIGACT News*, 28(2):40–52, 1997.
[15] S. Knight, H. Nguyen, N. Falkner, R. Bowden, and M. Roughan. The internet topology zoo.
[16] T. Koponen, M. Casado, and et al. Onix: A distributed control platform for large-scale production networks. In *OSDI*. USENIX, 2010.
[17] Z. M. Mao, R. Govindan, G. Varghese, and R. H. Katz. Route flap damping exacerbates internet routing convergence. *SIGCOMM CCR*, 32:221–233, August 2002.
[18] The openflow switch. `http://www.openflowswitch.org`.
[19] R. Sherwood, G. Gibb, K.-K. Yap, G. Appenzeller, M. Casado, N. McKeown, and G. Parulkar. Can the Production Network Be the Testbed? In *OSDI*. USENIX, 2010.
[20] M. Shindler. Approximation algorithms for the metric k-median problem. *Written Qualifying Exam Paper, University of California, Los Angeles. Cited on*, page 44.
[21] V. Vazirani. *Approximation algorithms*. Springer Verlag, 2001.
[22] H. Yan, D. Maltz, T. Ng, H. Gogineni, H. Zhang, and Z. Cai. Tesseract: A 4d network control plane. In *NSDI*. USENIX, 2007.