Logical SDNs: Reaping Software-Defined Networking Benefits Through Incremental Deployment

Dan Levin[†] Stefan Schmid[†,•] Fabian Schaffert[†] Marco Canini[‡] Anja Feldmann[†]
[†] TU Berlin [•] T-Labs [‡]Université catholique de Louvain

ABSTRACT

Although SDN promises principled approaches to long-standing network operations problems, with the exception of a few notable deployments in the wild, e.g., Google's B4, it remains largely an experimental technology for most organizations. As the transition of existing networks to SDN will not be instantaneous, we consider hybrid networks that combine SDN and legacy networking devices an important avenue of research; yet research focusing on these environments has so far been modest. Hybrid networks posses practical importance, are likely to be a problem that will span years, and present a host of interesting challenges, including the need for radically different networking paradigms to not only co-exist, but also offer clear, immediate benefits.

We argue that an appealing approach to hybrid networking can be achieved by introducing SDN devices into existing networks and abstracting the legacy network devices away as "expensive wires" to expose a programmatic "logical SDN" interface — conceptually, a representation of the network limited to just the SDN devices. Motivated by the need to better understand the potential and limits for the logical SDN abstraction for hybrid networks, in this work, we showcase the power and utility of the logical SDN by reasoning through and implementing use-case control applications built on this abstraction. Realizing this abstraction comes at the cost however, of diverting traffic from legacy switches through SDN devices. We thus explore the impact of the logical SDN on the network traffic flow performance through experiments in a high-performance emulation environment.

1. INTRODUCTION

The term "single pane of glass" [1, 3] has been coined in the systems and network operations community, to describe the ideal, operational "holy grail" where every input to a complex systems (e.g. a computer network) is expressed through a single, unified, common interface. Software-defined networking (SDN) is an attractive paradigm that potentially pushes network management closer to this ideal state.

SDN entails a logically-centralized control plane running different control applications, managing the forwarding behavior of a collection of switches via a standardized interface. The centralized perspective and simple interface have the potential to make the network more programmable, thereby reducing the complexity of network management (today an often cumbersome and manual process), and facilitate better-optimized and -automated network operation and troubleshooting.

Despite the need for principled approaches to long-standing network operational problems, with the exception of a few notable deployments in the wild (*e.g.*, Google's B4 [5]), SDN remains largely an experimental technology for most organizations.

One major reason for this mismatch is the *SDN deployment problem*: on the one hand, potential SDN adopters must first be able to establish confidence in SDN, but on the other hand SDN is not merely a "new feature" that can be "switched on" to provide value to existing networks. Moreover, as budgets are constrained, it is often not possible to replace all existing legacy hardware by SDN in one shot, but rather, *only a part of the network can be upgraded* at a time. An upgrade to SDN hence, does not begin with a green field, but with the existing deployment, and is typically a staged process. Even Goggle's B4 system required a significant multi-year deployment undertaking before its benefits could be realized. Smaller organizations will not typically have the resources to roll out their own SDN in a similar fashion. As such, we envision that transition to SDN will first occur in the form of partial deployments that co-exist with legacy hardware—that is, hybrid networks. Crucially, however, these partial deployments must provide value from the very beginning.

One frequently encountered hybrid deployment model occurs in the datacenter where SDN can be deployed at the edge (*i.e.*, on the server's hypervisor) [8]. In other settings, *e.g.*, in many enterprise networks, an upgrade of the edge is prohibitively expensive and out-of-the-question: the edge constitutes a significant fraction of the entire network; moreover, unlike in the datacenter, the edge is typically a legacy hardware switch not a software switch.

Our Contributions. In this paper, we consider the problem of how to operate an arbitrary hybrid network with the goal of enabling a partial SDN deployment to provide substantial benefits of the SDN programming interface. We propose a very general approach that abstracts the hybrid network as a *logical SDN*. Such a logical abstraction is attractive as it directly supports existing control applications which have been designed for full SDN deployments: the application can simply run on the provided logical SDN abstraction, which appears to the application as a "full deployment" of *just* the SDN switches.

The abstraction of a hybrid network as a logical SDN can be achieved by *SDN waypoint enforcement* [10]: the requirement that every packet between a source and a destination traverses at least one SDN switch, where the network policies are applied to the traffic using *e.g.*, the match-action paradigm.

This paper investigates the opportunities and limitations of such logical SDN abstractions, and we showcase the power of the logical SDN by reasoning through and implementing use-cases. Via these use cases, we demonstrate the utility of the programming interface offered by the logical SDN. The logical SDN abstraction comes at the cost however, of diverting traffic from legacy switches through SDN devices. We thus explore the impact of the logical SDN on the network traffic flow performance through experiments in a large-scale emulation environment. We find encouraging evidence that the traffic performance costs of waypoint enforcement may in many cases be moderate, and in some cases, performance can even improve.

2. LOGICAL SDN

Logical Software-Defined Networks are attractive in different settings. We proceed now to introduce Panopticon, an architecture that enables incremental SDN deployment to realize the logical SDN abstraction. Subsequently, we discuss alternative, useful network control abstractions that can be realized with the logical SDN, e.g., the big switch abstraction or the middlebox view.

2.1 Panopticon: Hybrid SDN

Panopticon [10] is an architecture to enable incrementally deployable, hybrid software-defined networks. Given an *arbitrary* deployment of SDN switches into an existing network, Panopticon allows the network operator to abstract away the legacy devices in the network and operate the network as an SDN comprised of just the SDN-capable switches. Using this approach, with careful planning of the hybrid deployment [9], SDN capabilities can be extended to potentially every switch-port in the network, not just the ports of SDN switches. Alternately, not every port need be included in the logical SDN, and in practice, resource constraints in the network may prevent a full SDN abstraction.

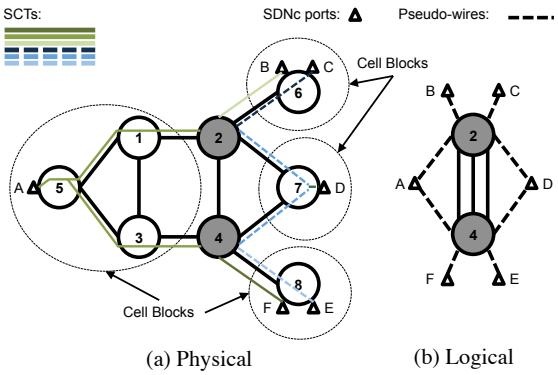

Figure 1: Example hybrid network of 8 switches (SDN switches are shaded). (a) Shows the SCTs (Solitary Confinement Trees) of every SDNc port overlaid on the physical topology. (b) Shows the corresponding logical view in which all SDNc ports are virtually connected to SDN switches via pseudo-wires.

Panopticon works on the principle that that every packet in the network that traverses at least one single SDN switch can have the end-to-end network policy (e.g., access control) applied to it via the SDN programming interface. Panopticon extends SDN capabilities to legacy switches by ensuring that all traffic to or from any SDN-controlled (SDNc) port is always restricted to a safe end-to-end path, that is, a path that traverses at least one SDN switch. We call this property *Waypoint Enforcement*. Panopticon uses mechanisms available on legacy switches (i.e., VLANs) to restrict forwarding on legacy switches to guarantee Waypoint Enforcement.

Example. Let us consider the example hybrid network of eight switches in Figure 1a. In this example, the Solitary Confinement Tree of A, SCT (A) is the tree that consists of the paths $5 \rightarrow 1 \rightarrow 2$ and $5 \rightarrow 3 \rightarrow 4$. Instead note that SCT (B), which corresponds to the path $6 \rightarrow 2$, includes a single SDN switch because switch 2 is the only SDN switch adjacent to cell block $c(B)$. Figure 1b shows the corresponding logical view of the physical hybrid network enabled by SCTs (a.k.a. VLANs). In this logical view, every SDNc port is connected to at least one frontier SDN switch via a pseudo-wire (realized by the SCT). Using this approach, Panopticon can be used to realize a broad spectrum of logical SDNs.

2.2 Big Switch Abstraction

Perhaps the simplest and most elegant network control abstraction achievable by a logical SDN is the Big Switch abstraction: only the ingress and egress ports of the network are exposed through the SDN programming interface and the network itself is considered a black box. Such an abstraction is ideal for defining policies whose implementation (i.e., whose flow-table rules) can be defined on any arbitrary switch located along

the path between two ports. Examples of such policies include access control, mobility management (*e.g.*, via address-locator separation), or application server load balancing. In general, this view is appealing for specifying policies that do not require visibility into the internal ports and connectivity between the devices in the network.

2.3 The Middlebox View

Many networks rely on middleboxes to increase security, improve performance and ensure policy compliance. Network planners and operators however, face challenges to carefully plan the network topology to ensure that traffic traverses the desired sequence of middleboxes in the right order, raising the overall network complexity. As noted in previous work [13], SDN offers a solution to realize middlebox traffic steering.

Through the logical SDN abstraction, we reason that it becomes possible within hybrid networks to benefit from the use of centralized management to orchestrate middlebox policy enforcement. To do so, the fidelity of the logical SDN must include enough detail (SDN switches) to permit the creation of a forwarding policy that steers traffic through the desired sequence of middleboxes. For example, the logical SDN could consist of a virtual "chain" where switches and middleboxes are interleaved. To support this kind of logical SDN, one would define a mapping to embed the logical-layer forwarding policy onto the underlying hybrid network, by using SDN waypoint enforcement.

3. TRAFFIC EMULATION STUDY

The abstraction of the "logical SDN" does not come for free, as the waypoint-enforcement of traffic through SDN switches can lead to increased path lengths in some cases. Panopticon however, also introduces new opportunities to improve traffic control within the network, e.g., enabling multi-path forwarding for load-balancing when sufficient path diversity exists.

To investigate the consequences of Panopticon on traffic, we conduct a series of emulation-based experiments on portions of a real enterprise network topology. These experiments (*i*) provide insights into the consequences of the Panopticon architecture waypoint enforcement on TCP flow performance and (*ii*) let us explore the extent to which the deployment size impacts TCP flow performance when every access point is operated as an SDNc port. To emulate traffic in a Panopticon deployment, we make use of mininet [4] as well as topology metadata from real enterprise networks with associated traffic workloads and network resource constraints.

Topologies: Detailed topological information, including device-level configurations, link capacities, and end-host placements is difficult to obtain for sizeable networks: operators are reluctant to share these details due to privacy concerns. Hence, we leverage publicly available enterprise network topologies [14,17] to provide the input to our emulation experiments. In our topology dataset, every link in the topology is annotated with its respective capacity. When port-channels (bundled links) are present, we represent them as a single link of their aggregate capacity. Summary information on the topology is given in Table 1.

Topology	Access/Dist/Core	max/avg/min degree
Full	1296 / 412 / 3	53 / 2.58 / 1
Emulated	489 / 77 / 1	30 / 6.3 / 1

Table 1: Emulated Network Topology Characteristics

In order to overcome the system resource bottlenecks when emulating such a large network, we necessarily scale down key aspects of the network topology: We (*i*) reduce the network size to a sub-graph of the full topology by pruning the graph along subnet boundaries, (*ii*) scale down the link capacities by 2 orders of magnitude, and (*iii*) correspondingly reduce the TCP MSS to 536 bytes to reduce packet sizes in concert with the reduced link capacities. These measures allow us to avoid system resource bottlenecks which would otherwise interfere with traffic generation and packet forwarding, thus influencing measured TCP throughput.

We run our experiments on a 64-core @ 2.6Ghz AMD Opteron 6276 system with 512Gb RAM running the 3.5.0-45-generic #68 Ubuntu Linux kernel using OpenVSwitch version 2.1.90. Our baseline experiments indicate that our system is capable of sustaining 489 simultaneous TCP connections in excess of 34Gb/s, sufficiently saturating the emulated aggregate link capacity of every traffic sender in our experiments.

Thus, our emulation experiments involve 489 SDNc ports located at "access switches" at which traffic is sent into and received from the network. The distribution network consists of 77 switches and routers, comprising a L2/L3 network in which 28 devices are identified as IP router gateways, bridging Ethernet broadcast domains over the remainder of the switches. Within each Ethernet broadcast domain, we introduce a single spanning tree to break forwarding loops.

Workload: The workload we apply to our experiments is defined both in terms of the traffic matrix defined over the 489 SDNc ports as well as a synthetically generated flow size distribution. We use a methodology similar to that applied in SEATTLE [7] to generate a traffic matrix based on packet-level traces from an enterprise campus network, the *Lawrence Berkeley National Laboratory* (LBNL) [12]. The LBNL dataset contains more than 100 hours of anonymized packet level traces of activity of several thousand internal hosts. The traces were collected by sampling all internal switchports periodically. We aggregate the recorded traffic according

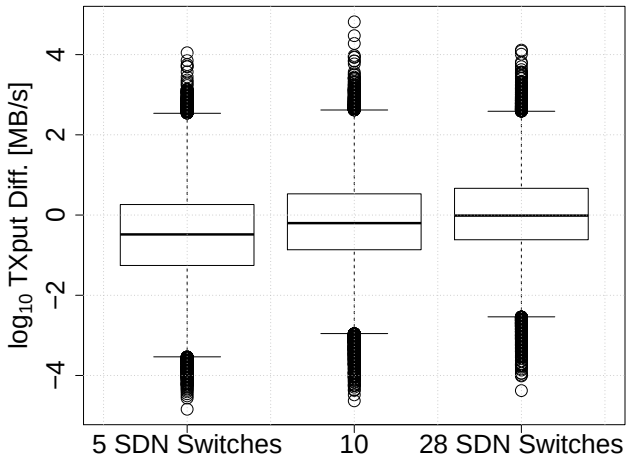

Figure 2: $\log_{10}$ difference in TCP throughput between Panopticon vs. original legacy deployment. In both scenarios B and C (10 and 28 SDN Switches), the median throughput over all experiments remains very close to the performance of the original legacy network.

to source-destination pairs and for each sample, we estimate the load imposed on the network. We note that the data contains sources from 22 subnets.

To map the traffic matrix onto our topology, we use the subnet information from the traces to partition the topology into subnets as well. Each of these subnets contains at least one distribution switch. In addition, we pick one node as the Internet gateway. We associate traffic from each subnet of the LBNL network in random round-robin fashion to candidate SDNc ports. All traffic within the LBNL network is aggregated to produce the intra-network traffic matrix. All destinations outside of the LBNL network are assumed to be reachable via the Internet gateway and, thus mapped to our designated gateway node in the topology. By running 10 different random port assignments for every set of parameters, we generate different traffic matrices which we use in our simulations.

We use a Weibull distribution with shape and scaling factor of 1 to define the object sizes that define our TCP flow size distribution, given in Table 2. Using our flow size distribution with the traffic matrix, we deterministically initiate TCP connections with the same request patterns over 10 repeated experiments, each using a differently seeded traffic matrix. Using iperf, every SDNc port (traffic source) in a sequential iterative fashion initiates TCP connections with the partners defined in the traffic matrix, transferring data defined from the flow size distribution until a maximum limit of 100MB has been sent into the network. Once every SDNc port has reached this limit, the experiment stops and the flow completion times are collected.

Scenarios: We consider three different deployment scenarios in which we evaluate the effects of Panopticon on TCP traffic: Scenario A in which 5 of the 77 switches and routers, selected according to the VOL algorithm outlined in [9] are operated as SDN switches, and scenarios B and C in which 10 and respectively all 28 L3 IP routers of the topology are operated as SDN switches. In each scenario, we compare TCP flow throughput in the Panopticon deployment to the conventional L2/L3 shortest path IP routed network with minimum cost spanning trees. Table 2 gives the relevant path stretch statistics for each topology, namely, the ratio of SDN (waypoint-enforced) to legacy path length for every src-dst path in the network.

In Figure 2 we illustrate the impact of Panopticon waypoint enforcement on TCP performance in the three scenarios. The first observation we make is that in scenario C, when just the 28 IP routers are replaced with SDN switches, the impact on median TCP throughput is negligible. This is perhaps expected, as all traffic across subnets must traverse some IP router in the legacy network, regardless. In extreme cases, 5% of the TCP flows experience significant deviations from the legacy network case: Some flows experience congestion due to the waypoint enforcement. Other flows actually experience a speed-increase due to the availability of multiple alternate paths in Panopticon. As the SDN deployment shrinks to more conservative sizes in scenarios B and A, the effects of waypoint-enforcement clearly become more apparent, although in all scenarios the median TCP connection throughput, never decreases by more than a factor of 3 (keep in mind the $\log_{10}$ y axis) when compared to the legacy network. These results are encouraging, as they demonstrate that a network of fewer than 10% SDN switches can operate as an SDN while accommodating 25% of its traffic with performance better than or equal to the original network and median throughput no worse than 1/3 of its original rate.

4. RELATED WORK

Evolvable inter-networking. The question of how to *evolve* or run a *hybrid* network, predates SDN and has been discussed in many contexts, including Active Networks [16]. Generally, changes in the network layer typically pose a strain to network evolution, which lead to overlay approaches being pursued (*e.g.*, [6]). In this sense, the concept of Waypoint Enforcement used by Panopticon is grounded on previous experience.

Hybrid and transitional networking. Recently the Open Networking Foundation Migration Working Group [2] has shed some light on high-levels guidelines and methods to incrementally deploy SDN in existing networks. This first document [2] focuses on three migration case studies: Google's WAN, the NTT provider edge, and Stanford campus network. What can be clearly seen from these studies is that the mi-

Parameter or Metric	min	25 %ile	50 %ile	avg	75 %ile	max
Flow Size Distribution (in MB)	0.00005	2.88	6.91	9.94	13.72	101.70
Path Stretch (5 SDN Switches)	1.0	1.0	1.33	1.25	1.33	3.0
Path Stretch (10 SDN Switches)	1.0	1.0	1.0	1.16	1.33	3.0
Path Stretch (28 SDN Switches)	1.0	1.0	1.0	1.002	1.0	1.67

Table 2: Traffic Parameter and Path Stretch Metric Statistics

gration strategies have strong dependencies on the specific migration scenario and requirements, as well as how the migration is subdivided into phases. For example, Google's transition to a software-defined WAN involved an overhaul of their *entire* switching hardware to improve network performance [5].

Vissicchio *et al.* [15] explore hybrid SDN models that combine SDN-style control with traditional L3 networking protocols. They show a number of use cases in which hybrid models can mitigate the respective limitations of traditional and SDN approaches, providing incentives to (partially) transition to SDN. Complementary to these works, Panopticon represents an additional point in the space of hybrid networks that takes an explicit stance at transitioning to an SDN control plane without the need for a complete hardware upgrade. An extended abstract on Panopticon appeared at ONS 2013 [10], and a detailed description of Panopticon can be found in our full paper [11].

This paper builds upon the architecture of Panopticon and studies the concept of logical SDNs from a more general standpoint while providing experimental results of its feasibility.

5. CONCLUSION

This paper initiates the study of the logical SDN abstraction for hybrid software-defined networks. We weigh the benefits and limitations of different logical SDN abstractions by implementing and reasoning about several use cases. We also investigate the impact along with the opportunities presented by the abstraction in terms of network traffic performance in emulation.

We understand our work as an initial step towards a better understanding of the implications and trade-offs of SDN abstractions. In particular, we believe that the introduced notion of "logical SDN" is relevant far beyond the specific scope of incremental SDN deployment, but for other instances of hybrid, system architectural transitions. We therefore believe that our work addresses an interesting and relevant field of research.

6. REFERENCES

[1] Manage Devices Through a 'Single Pane of Glass'. `http://www.microsoft.com/en-us/news/features/2012/apr12/wedmfea.aspx`.

[2] Open Networking Foundation Migration Working Group: Migration Use Cases and Methods. `https://www.opennetworking.org/images/stories/downloads/sdn-resources/use-cases/Migration-WG-Use-Cases.pdf`.

[3] L. Bitincka, A. Ganapathi, and S. Zhang. Experiences with workload management in splunk. In *Proceedings of the 2012 Workshop on Management of Big Data Systems*, MBDS '12, pages 25–30, New York, NY, USA, 2012. ACM.

[4] N. Handigol, B. Heller, V. Jeyakumar, B. Lantz, and N. McKeown. Reproducible network experiments using container-based emulation. In *Proceedings of the 8th international conference on Emerging networking experiments and technologies*, CoNEXT '12, pages 253–264, New York, NY, USA, 2012. ACM.

[5] S. Jain, A. Kumar, S. Mandal, J. Ong, L. Poutievski, A. Singh, S. Venkata, J. Wanderer, J. Zhou, M. Zhu, J. Zolla, U. Hoelzle, S. Stuart, and A. Vahdat. B4: Experience with a Globally-Deployed Software Defined WAN. In *SIGCOMM*, 2013.

[6] D. Joseph, J. Kannan, A. Kubota, K. Lakshminarayanan, I. Stoica, and K. Wehrle. OCALA: An architecture for supporting legacy applications over overlays. In *NSDI*, 2006.

[7] C. Kim, M. Caesar, and J. Rexford. Floodless in Seattle: A scalable ethernet architecture for large enterprises. In *SIGCOMM*, 2008.

[8] T. Koponen *et al.*. Network virtualization in multi-tenant datacenters. In *NSDI*, 2014.

[9] D. Levin, M. Canini, S. Schmid, and A. Feldmann. Panopticon: Reaping the benefits of partial sdn deployment in enterprise networks. Technical report, Technical Report TU Berlin `http://bit.ly/1n1U3LD`, 2013.

[10] D. Levin, M. Canini, S. Schmid, and A. Feldmann. Toward transitional sdn deployment in enterprise networks. *Proc. Open Networking Summit (ONS)*, 2013.

[11] D. Levin, M. Canini, S. Schmid, and A. Feldmann. Panopticon: Reaping the Benefits of Incremental SDN Deployment in Enterprise Networks. In *Proceedings of the 2014 USENIX Annual Technical Conference (ATC'14)*, Jun 2014.

[12] R. Pang, M. Allman, M. Bennett, J. Lee, V. Paxson, and B. Tierney. A first look at modern enterprise traffic. In *ACM IMC*, 2005.

[13] Z. Qazi, C.-C. Tu, L. Chiang, R. Miao, V. Sekar, and M. Yu. SIMPLE-fying Middlebox Policy Enforcement Using SDN. In *SIGCOMM*, 2013.

[14] Y.-W. E. Sung, S. G. Rao, G. G. Xie, and D. A. Maltz. Towards systematic design of enterprise networks. In *CoNEXT*, 2008.

[15] S. Vissicchio, L. Vanbever, and O. Bonaventure. Opportunities and research challenges of hybrid software defined networks. *ACM Computer Communication Review*, 44(2), April 2014.

[16] D. J. Wetherall. Service introduction in an active network. Technical report, M.I.T. PhD Thesis, 1999.

[17] H. Zeng, P. Kazemian, G. Varghese, and N. McKeown. Automatic test packet generation. In *CoNEXT*, 2012.