Technische Universität Berlin

Forschungsberichte
der Fakultät IV – Elektrotechnik und Informatik

Panopticon: Reaping the Benefits of Partial SDN Deployment in Enterprise Networks

Dan Levin
Marco Canini
Stefan Schmid
Anja Feldmann

Technische Universität Berlin /
Deutsche Telekom Laboratories

Bericht-Nr. 2013 - 04
ISSN 1436-9915

Panopticon: Reaping the Benefits of Partial SDN Deployment in Enterprise Networks

Dan Levin Marco Canini Stefan Schmid Anja Feldman
TU Berlin / T-Labs
<first name>@net.t-labs.tu-berlin.de

ABSTRACT

The operational challenges posed in enterprise networks, present an appealing opportunity for the software-defined orchestration of the network (SDN). However, the primary challenge to realizing solutions built on SDN in the enterprise is the deployment problem. Unlike in the data-center, network upgrades in the enterprise start with the existing deployment and are budget and resource-constrained.

In this work, we investigate the prospect for partial Software Defined Network (SDN) deployment. We present Panopticon, an architecture and methodology for planning and operating networks that combine legacy and upgraded SDN switches. Panopticon exposes an abstraction of a logical SDN in a partially upgraded legacy network, where the SDN benefits extend potentially over the entire network.

We evaluate the feasibility of our approach through simulation on real enterprise campus network topologies entailing over 1500 switches and routers. Our results suggest that with only a handful of upgraded switches, it becomes possible to operate most of an enterprise network as a single SDN while meeting key resource constraints.

1. INTRODUCTION

Mid to large enterprise campus networks present complex operational requirements: The network must operate reliably and provide high-performance connectivity while enforcing organizational policy. It must also provide isolation across complex boundaries, yet remain easy to manage. All the while, operational and capital costs must be kept low.

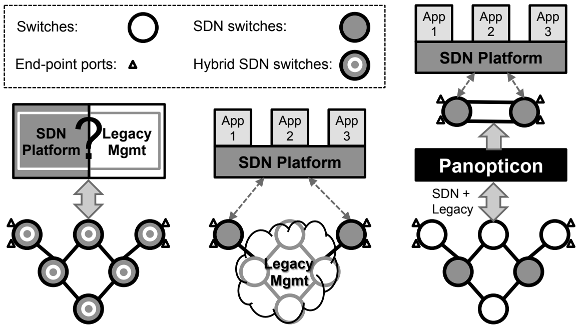
In the face of these requirements, numerous operational challenges threaten high availability and lead to increased costs: *Policy changes and resource reallocations* lead to reconfigurations, often scattered across the network. This demands coordination among multiple human operators to avoid inconsistencies and human reasoning to ensure correctness of the resulting update. In personal correspondence, a network operator remarked that "for the fear of breaking policy, human operators are reluctant to ever remove existing rules." *Troubleshooting* is an expensive, time-consuming activity for network operators. A recent survey [32] shows that 56.2% of reported network tickets take over 30 minutes to resolve, and 35% of surveyed operators receive more than 100 tickets per month. *Scheduled maintenance* poses an additional challenge: The network provides no means for the operator to express that certain devices will be unavailable for a period of time. Therefore, he must reason about the consequences of reconfiguring the network for device removal, the correctness of the order in which he performs the actions, and manually introduce the correct configuration changes to mitigate the impact of service.

Software Defined Networking (SDN) has the potential to provide a principled solution to the orchestration of these challenging tasks. SDN is a paradigm that offers a programmatic, logically-centralized interface for specifying network behavior via direct control of the switch hardware forwarding state. Through this interface, a software program acts as a network controller by writing forwarding rules into switch tables and reacting to topology and traffic changes. Therefore, SDN represents a clear opportunity over manual, error-prone, ad-hoc approaches to handling the above tasks.

However, most of the existing work leveraging SDN (*e.g.*, [4, 9, 13, 27]) has so far assumed a full deployment of SDN switches. Rather than a green field, network upgrade starts with the existing deployment and is typically a staged process —budgets are constrained, and only a part of the network can be upgraded at a time. SDN deployment in the enterprise is no exception.

The realities of network upgrade and the operational challenges facing existing networks lead us to question: (*i*) What are the benefits of upgrading to a partial SDN deployment? (*ii*) How do the benefits of principled network orchestration depend on the location of SDN switches? (*iii*) Given budget constraints, what subset of legacy switches should be SDN upgraded to maximize benefits?

To answer these, we present **Panopticon**, an architecture and methodology for aiding operators in planning and operating networks that combine legacy switches and routers and SDN switches. We call such networks *transitional networks*. Panopticon overcomes



(a) Dual-stack (b) Access edge (c) Panopticon

Figure 1: Current transitional network approaches vs. Panopticon: (a) Dual-stack ignores legacy and SDN integration. (b) Full edge SDN deployment enables end-to-end control. (c) Panopticon partially-deployed SDN yields an interface that acts like a full SDN deployment.

the limitations of current approaches for transitional networks, which we now briefly review.

1.1 Current Transitional Networks

The first approach (Figure 1a) partitions the flow space into several disjoint slices and assigns each slice to either SDN or legacy processing [20]. Individual traffic flows of interest may be explicitly selected for SDN processing. This mode's prime limitation is that it is essentially a dual-stack approach (as with IPv6 + IPv4) rather than a means to integrate legacy hardware and expose the resulting transitional network as SDN. Further, this approach necessitates a contiguous deployment of hybrid programmable switches capable of processing packets according to both legacy and SDN mechanisms, *i.e.*, those switches studied under the ONF Hybrid Working Group [24].

The second approach (Figure 1b) involves deploying SDN at the network access edge [6]. This mode has the benefit of enabling full control over the access policy and the introduction of new network functionality at the edge, *e.g.*, data-center network virtualization [1]. Unlike a data-center environment where the network edge may terminate at the VM hypervisor, the campus network edge terminates at an access switch. In an enterprise network, this approach thus involves upgrading thousands of access switches and incurs a high cost. SDN deployment limited to the edge additionally impairs the ability to control forwarding decisions within the core of the network (*e.g.*, load balancing, waypoint routing).

1.2 Panopticon

Panopticon encompasses (*i*) a methodology for determining the cost-aware partial deployment of SDN switches for specific operational objectives and (*ii*) an architecture for operating transitional networks as SDN.

Our main insight is that *the key benefits of the SDN abstraction to enterprise networks can be realized for every source-destination path that includes at least one SDN switch.* Thus, we do not mandate a full SDN deployment—a relatively small subset of all switches may suffice. Each path which traverses even just one SDN switch, can be used to realize a programmatic, logically-centralized interface for orchestrating *e.g.*, the network access control policy. Moreover, traffic which traverses two or more SDN switches may be controlled at even finer levels of granularity enabling further customized forwarding decisions *e.g.*, for load balancing.

Based on this insight, we first develop a cost-aware optimization framework as a tool for the network operator to determine the location of the partial SDN deployment based on their objectives (*e.g.*, capex or forwarding efficiency). Second, we design *Panopticon* which provably guarantees that traffic destined to operator-selected end-points passes through at least one SDN switch. Just as enterprise networks regularly divert traffic (*e.g.*, one VLAN to reach another on the *same* switch must traverse a gateway), Panopticon explicitly leverages waypoints to control traffic.

As opposed to the dual-stack approach, Panopticon (Figure 1c) fundamentally integrates legacy and SDN switches yielding an abstraction of a logical SDN to the control platform. As we reason later (§ 7), many SDN control paradigms can be achieved. Panopticon enables the expression of any end-to-end policy, as though the network were one big, virtual switch. End-to-end policies include access control and application load balancing. Routing and path-level policy, e.g., traffic engineering can be expressed too, however the fidelity of the global network view (and path diversity) presented to the control logic is reduced to the fidelity of the logical SDN. As more of the network is upgraded to support SDN, more fine-grained path-level policy can be expressed. In a sense, Panopticon generalizes a fabric deployment [6], which is impractical in the enterprise context where the edge is so large and embroiled in legacy equipment.

The namesake of our approach is inspired by the Panopticon prison architecture, in which prisoners are confined to cell-blocks, observable and controlled from strategic vantage points. Analogous to this prison, we isolate end-hosts in the legacy network using VLANs (in what we term Solitary Confinement Trees or SCTs) and restrict their traffic to traverse strategically upgraded SDN programmable switches.[1] Figure 2 illustrates the forwarding in Panopticon which we revisit in § 3.

We face the challenges of (*i*) the need to maintain compatibility with legacy switches and protocols, and (*ii*) scalability issues with VLAN and flow table state. Panopticon overcomes these challenges by (*i*) relying

[1] See § 2.3 for background on OpenFlow, an SDN platform.

on features such as VLANs ubiquitously available on enterprise-grade switches, as confirmed through our operator survey (§ 2), and (*ii*) a rigorous design methodology that considers VLAN and flow table constraints yielding a careful assignment of VLAN IDs that allows us to create a spanning tree for every end-point in a scalable fashion.

1.3 Research Contributions

In summary, Panopticon enables us to expose an interface for operating a transitional network as if it were a nearly fully deployed SDN and reap the benefits of partial SDN deployment for most of the network, not just the part that is upgraded. Through a rigorously designed SDN deployment that takes control over the underlying legacy resources, Panopticon is more than just an ad-hoc tunneled SDN overlay. The novelty of our architecture lies in the way we overcome the challenge of combining existing mechanisms readily available in legacy switches. We make the following contributions:

1. We design a network architecture for operating a partially upgraded network as an SDN (§ 3). Also, we study the interaction mechanisms between legacy and upgraded switches (§ 4). We also include proofs for the correctness of our approach (Appendix).

2. We formalize the problem of determining the optimal upgrade location as a mixed-integer optimization program (§ 5) and we devise efficient algorithms to solve it (§ 5.4).

3. We evaluate our approach using real enterprise network topologies (with over 1500 switches) and traffic traces (§ 6). We find that with only a handful of switches, it becomes possible to operate most of an enterprise as a single SDN while meeting all practical resource constraints.

4. We demonstrate our system-level feasibility with a prototype implementation in (§ 6.5).

To motivate our problem formulation we conduct a survey of network operators, which we now present.

2. ENTERPRISE NETWORK SURVEY

Commercial deployment of SDN started within datacenters. The next steps are enterprise and/or ISP networks. In this paper we focus on mid to large enterprise campus networks, *i.e.*, networks serving hundreds to thousands of users, whose infrastructure is physically located at a campus site. We choose this environment due to its challenging complexity as well as the impact potential that SDN network orchestration promises.

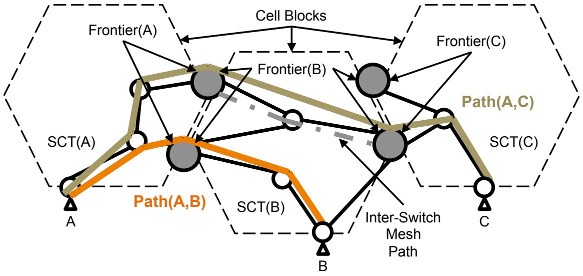


Figure 2: The forwarding path between A and B goes via the frontier shared by SCT (A) and SCT (B); the path between A and C goes via an Inter-Switch Mesh path connecting SCT (A) and SCT (C).

2.1 Enterprise Network Operator Survey

To support our key assumptions on the challenges and operational objectives within enterprise campus networks, we conducted two on-site interviews with operators from both large (≥10,000 users) and medium (≥500 users) enterprise networks. We then solicited 60 responses via e-mail to open-answer survey questions from a wider audience of 60 network operators.

The top three responses to, "what is the most important technical, operational objective you must achieve in your network" were: (1) That it "just works", implying basic, usable IP connectivity for employees to conduct their business, (2) strict traffic filtering and perimeter protection against intrusion prevention and system exploit, and (3) minimize operational costs and complexity, *e.g.*, by maintaining homogeneous hardware deployments with vendor-specific management tools.

The top response to "what is the most demanding operational challenge faced in your network" was related to maintaining a consistent view of the network's physical and configuration state. When asked whether network capacity bottlenecks present problems, no operator indicated that his or her network experienced such issues.

2.2 Legacy Network Assumptions

Consequently, based on our operator conversations, and in conjunction with several design guidelines (*e.g.*, see [7, 15]), we make the following assumptions about medium to large enterprise campus networks and hardware capabilities. Enterprise network hardware consists primarily of Ethernet bridges, namely, switches that implement standard L2 mechanisms (*i.e.*, MAC-based learning and forwarding, and STP) and support VLAN (specifically, 802.1Q and per VLAN STP). Routers or L3 switches are used as gateways to route between VLANs-isolated IP subnets. For our purposes, we assume an L3 switch is also capable of functioning as a L2 switch. In addition, we assume that no enterprise intentionally operates "flood-only" hub devices for general packet forwarding.

2.3 Background on OpenFlow

The current de-facto standard SDN platform is OpenFlow [20]: it defines a switch model and an API to its forwarding tables, as well as a protocol that exposes the API to a controller program. OpenFlow provides us a reference model to reason about the types of forwarding behaviors that an upgraded switch may practically implement. Henceforth, we assume that SDN switches adhere to the OpenFlow programmable switch model, which we now briefly review.

OpenFlow specifies that switches have flow tables which store a list of rules for processing packets. Each rule consists of a *pattern* (matching on packet header fields), *actions* (such as forwarding, dropping, sending to the controller, *etc.*), a *priority* (to distinguish between overlapping patterns), *counters* (to measure bytes and packets processed so far), and a *timeout* (indicating if/when the rule expires). An OpenFlow switch then matches every incoming packet against the flow table rules. Whenever there is a match, the switch selects the highest-priority matching rule, updates the counters, and performs the specified action(s). If there is no matching rule, the switch sends the packet (in full or just its header) to the controller and awaits a response on what actions to take.

3. PANOPTICON SDN ARCHITECTURE

This section presents our architecture for partially-deployed software defined networks. We extend the SDN capabilities to legacy switchports by ensuring that every pair of SDN controlled legacy switchports are restricted to communicate over an end-to-end path that traverses at least one SDN switch. We call this property, the WAYPOINT ENFORCEMENT policy. The WAYPOINT ENFORCEMENT policy can however be violated if legacy devices are allowed to make standard forwarding decisions (*i.e.*, based on destination MAC address).

To guarantee, WAYPOINT ENFORCEMENT we must choose a forwarding set that constrains the space of possible forwarding decisions in such a way that *the traffic always follows safe end-to-end paths.* In addition, we must do so using *only existing mechanisms and features readily available* on legacy switches, since these switches are not being upgraded. We start by formally introducing the concepts of WAYPOINT ENFORCEMENT, FORWARDING SET and SAFE PATH.

3.1 The Transitional Network Model

We define a transitional network as $G = (\mathcal{N}, \mathcal{E})$. G is connected, and consists of a set of nodes $\mathcal{N}$ which are partitioned into end-points Π, legacy switches $\mathcal{L}$ and SDN switches $\mathcal{S}$, *i.e.*, $\mathcal{N} = \Pi \sqcup \mathcal{L} \sqcup \mathcal{S}$, and a set of undirected links $\mathcal{E}$ between two elements of $\mathcal{N}$.

We represent a *end-to-end* path traversed by packets from source end-point $s \in \Pi$ to destination end-point $t \in \Pi$ as a list of traversed links $p(s,t) = (e_1 = \{s, u_1\}, e_2 = \{u_1, u_2\}, \ldots, e_k = \{u_{k-1}, t\})$, where $u_1, \ldots, u_{k-1} \in \mathcal{L} \sqcup \mathcal{S}$. To express paths between switches rather than between end-points, we simply overload the definition of path $p(s,t)$ with $s, t \in \mathcal{L} \sqcup \mathcal{S}$.

We define the *reachability matrix* R where $R_{s,t} = 1$ iff end-point pair (s,t) is allowed to communicate, otherwise 0. From all possible paths between end-points (s,t) in G, we denote with $FS(s,t)$ the subset of utilized paths subject to $R_{s,t}$, which we call the *forwarding set.*

The set of end-points Π is further subdivided into SDN-policed end-points $\Pi^\bullet$ and non SDN-policed end-points Π°. We call a SDN-policed end-point a *SDN port.* In Panopticon, we want to ensure that traffic from or to a SDN port can only reach another end-point (regardless of its type) via a SDN switch. To this end, we introduce the concept of Waypoint Enforcement.

DEFINITION 1 (WAYPOINT ENFORCEMENT). Waypoint Enforcement *requires that every path in the forwarding set* $FS(s,t)$ *includes* at least one SDN switch. *Formally,* $\forall\ s, t \in \Pi^\bullet\ \forall\ p \in FS(s,t)$
$\exists\ u \in \mathcal{S}\ :\ u \in p$.

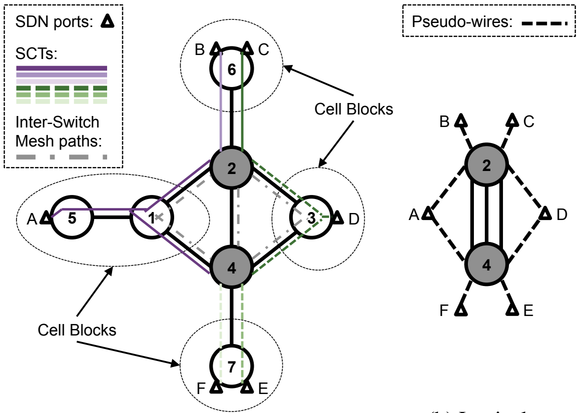
Henceforth, we call an end-to-end path *safe* iff it satisfies the WAYPOINT ENFORCEMENT.

3.2 Selecting the Forwarding Set

We next show how to construct the forwarding set using VLANs *i.e.*, 802.1Q, to isolate and constrain traffic in the legacy network to safe paths, ultimately achieving WAYPOINT ENFORCEMENT. To conceptually illustrate how VLANs restrict forwarding to use safe paths in a transitional network, we first consider a straightforward, yet impractical scheme: For every pair of SDN ports (*i.e.*, end-points subject to waypoint enforcement), choose one SDN switch as a waypoint, and compute the (shortest) end-to-end path that includes the waypoint. Next, assign a unique VLAN ID to every end-to-end path and configure the legacy switches accordingly. This ensures that all forwarding decisions made by every legacy switch only send packets along safe paths.

Due to practical constraints, this simple solution is infeasible as the VLAN ID space is limited to 4096 values. Indeed, VLAN ID space may be smaller yet, due to switch hardware limitations. Thus, the simple solution supports at most 64 hosts (in a full mesh) before depleting all available VLAN IDs. Moreover, end-point ports *must operate in "access mode" with a single VLAN ID* as end-host interfaces may not support or be trusted to correctly use 802.1Q trunking.

A naive work around is to assign a single designated SDN switch per SDN port. However, this rigid solution limits the opportunity to use the best path to the



(a) Physical (b) Logical

Figure 3: An example transitional network of 7 switches (SDN switches are shaded). (a) Shows the SCTs (Solitary Confinement Trees) of every SDN ports overlaid to the physical topology. (b) Presents the corresponding logical view in which all SDN ports are virtually connected to SDN switches via pseudo-wires.

destination according to distance or load. Moreover, it creates immense reliability concerns.

3.3 Solitary Confinement Trees (SCTs)

To construct the forwarding set in Panopticon, we introduce the concept of *Solitary Confinement Tree* (SCT) to provide end-to-end path diversity while ensuring isolation and a parsimonious use of VLAN IDs. We first describe the terms *Cell Blocks* and *Frontier* and then formally define the SCT, using an example to illustrate.

Definition 2 (Cell Blocks). *Given a transitional network G,* Cell Blocks $CB(G) = \{c_1, \ldots, c_k\}$ *is defined as the* set of connected components *of the network G' obtained after removing from G the SDN switches $\mathcal{S}$ and their incident links. Formally, $G' = (\Pi \sqcup \mathcal{L}, \mathcal{E}')$, where $\mathcal{E}' = \mathcal{E} \setminus \{e = \{u_1, u_2\} \mid \exists u \in e : u \in \mathcal{S}\}$.*

Definition 3 (Frontier). *Given a cell block $c \in CB(G)$, we define the* Frontier *$\mathcal{F}(c)$ as the subset of SDN switches that are adjacent in G to a switch in c.*

Intuitively, the solitary confinement tree is a spanning tree within a cell block, *plus* its frontier. The principle role of each SCT is to provide a safe path from each SDN port π to every SDN switch in its frontier. We can then assign a single VLAN ID to each SCT which ensures traffic isolation and overcomes the limitation of binding each SDN port to a single SDN switch. We introduce an algorithm for choosing VLAN IDs in § 4.2. Formally we define SCT as:

Definition 4 (Solitary Confinement Tree). *Let $c(\pi)$ be the cell block to which an SDN port $\pi \in \Pi^\bullet$ belongs. And let* $\mathrm{STP}(c(\pi))$ *denote the STP-computed spanning tree on $c(\pi)$. Then, the* Solitary Confinement Tree $\mathrm{SCT}(\pi)$ *is the network obtained by augmenting* $\mathrm{STP}(c(\pi))$ *with the upgraded frontier $\mathcal{F}(c(\pi))$, together with all links in G connecting a switch $u \in \mathcal{F}(c(\pi))$ with a switch in* $\mathrm{SCT}(\pi)$.

Example. Let us consider the example transitional network of seven switches in Figure 3a. In this example, SCT (A) is the tree that consists of the paths 5 → 1 → 2 and 5 → 1 → 4. Instead note that SCT (B), which corresponds to the path 6 → 2, includes a single SDN switch because switch 2 is the only SDN switch adjacent to cell block $c(B)$. Figure 3b shows the corresponding logical view of the physical transitional network enabled by having SCTs. In this logical view, every SDN port is connected to at least one SDN switch via a pseudo-wire.

3.4 Packet Forwarding in Panopticon

We now illustrate Panopticon's basic forwarding behavior (Figure 2). Let us first consider traffic between a pair of SDN ports s and t. When a packet from s enters $\mathrm{SCT}(s)$'s VLAN, the legacy switches forward the packet to the frontier based on MAC-learning which establishes a symmetric path. Note, that a packet from s may use a different path within $\mathrm{SCT}(s)$ to the frontier for each distinct destination. Once traffic toward t reaches its designated SDN switch $u \in \mathcal{F}(c(s))$, one of two cases arises:

SDN switches act as VLAN gateways: This is the case when the destination SDN port t belongs to a cell block whose frontier $\mathcal{F}(c(t))$ shares at least one switch u with $\mathcal{F}(c(s))$. Switch u acts as the designated gateway between $\mathrm{SCT}(s)$ and $\mathrm{SCT}(t)$: That is, u rewrites the VLAN tag and places the traffic within $\mathrm{SCT}(t)$. For instance, in the example of Figure 3a, switch 2 acts as the gateway between ports A, B and C.

Inter-Switch Mesh (ISM) and path diversity: When no SDN switch is shared, we use an *Inter-Switch Mesh (ISM) path*: point-to-point, VLAN-based tunnels that realize a full mesh between SDN switches. In this case, the switch u chooses one of the available paths to forward the packet to an SDN switch $w \in \mathcal{F}(c(t))$, where w is the designated switch for the end-to-end path $p(s,t)$. In our example of Figure 3a, ISM paths are shown in gray and are used *e.g.*, for traffic from B or C to E or F, and vice versa.

As in any SDN, the control platform is responsible for installing the necessary forwarding state according to the reachability matrix M and for reacting to topology changes (fault tolerance is discussed in § 4.4).

We next turn to the forwarding behavior of non SDN ports. Again, we distinguish two cases. First, when there exists a path in the legacy network between two non SDN ports, forwarding is performed as usual and is unaffected by the partial SDN deployment. Policy enforcement and other operational objectives must be implemented through traditional means, *e.g.*, ACLs.

In the second case, anytime a path between two non SDN ports necessarily encounters an SDN switch, the SDN mechanism can be leveraged to police the traffic. This is also the case for all traffic between any pair of SDN and non SDN ports. In other words, Panopticon *always guarantees safe paths for packets from or to every SDN port.* We formally prove this property in the Appendix.

3.5 Architecture Discussion

Having described all components of the architecture, we now highlight the key properties of SCT and ISM.

A VLAN ID per SCT is scalable. Using SCTs is inherently more scalable than using one VLAN ID on each end-to-end path since we parsimoniously use just one VLAN ID per SDN port. In addition, our scheme does not preclude using a different path within the SCT for every destination ingress port, granting more flexibility over using a single designated switch per SDN port.

VLAN IDs are reused across Cell Blocks. In addition, we observe that SCTs allow us to reuse VLAN IDs because *any VLAN ID can be used once in each cell block* independently. This is because SDN switches effectively act as gateways between cell blocks and modify VLAN IDs accordingly.

SCTs can be statically precomputed. We observe that potentially, there is only a one time cost to compute all SCTs (of course, re-computation is needed whenever switches are added or removed). Also, it is possible to *automatically produce configuration settings* to correctly setup each legacy switch without requiring any manual, tedious and error-prone effort.

ISM path diversity trade-offs. Within the ISM, there may be multiple paths between any given pair of SDN switches. We expect that some applications may require a minimum number of paths. For example, a minimum of two disjoint paths is necessary, to tolerate single link failures. On the other hand, each path consumes a VLAN ID from the ID space of every traversed cell block. Henceforth in this paper, we limit ourselves to a single shortest path per switch pair (unless we specify otherwise).[2] Path control over the mesh is exercised by the logically centralized SDN controller which we describe next.

3.6 Realizing SDN Benefits

By now, we have described how Panopticon shifts the active network management burden away from the legacy devices and onto the upgraded SDN deployment. This conceptually reduces the network to a logical SDN as presented in Figure 3b. The transitional network, as an SDN, stands to benefit in all the ways as previous work has demonstrated—dynamic policy enforcement [4], consistent policy updates [27], network behavior customization and debugging [13, 14, 31], *etc.*

Putting it all together, Panopticon is the first architecture that realizes an approach for operating a transitional network as though it were a fully deployed SDN, yielding the benefits of partial SDN deployment for the entire network, not just the part that is upgraded.

4. LEGACY SWITCH INTERACTION

Next, we consider the interaction of upgraded switches with legacy switches. Accordingly, we discuss how the interaction with STP works, as well as how to choose VLAN IDs, cope with broadcast traffic, and tolerate failures.

4.1 Interacting with STP

The Spanning Tree Protocol (STP) or a variant such as Rapid STP, is commonly used to achieve loop freedom within L2 domains and we interact with STP in two ways. First, to ensure network-wide loop freedom for traffic from non SDN ports, SDN switches behave as ordinary STP participants. That is, the SDN controller implements STP and coordinates the computation and distribution of STP messages on every SDN switch.

Second, within each SCT, we run a per-VLAN spanning tree protocol (*e.g.*, Multiple STP) rooted at the SCT ingress port's switch. For this STP instance, each SDN switch passively listens to learn the least-cost path to the SCT's ingress port, but *does not reply* with any STP messages. Collectively, this behavior guarantees that each SCT is loop free and fault tolerant via the existing STP failover mechanisms.

4.2 Deploying VLANs

The configuration of VLANs and the assignment of VLAN IDs can be computed efficiently in Panopticon: For each cell block $c \in CB(G)$, we compute all $\mathrm{SCT}(\pi)$ (for each SDN port $\pi \in \Pi^{\bullet}$), and use one VLAN ID per SCT. As noted before (§ 3.3), VLAN IDs can be reused across different cell blocks. Subsequently, we add the VLANs for the Inter-Switch Mesh (ISM). For each switch pair connected in the ISM by a path p, we use one VLAN whose ID is the smallest available identifier in the cell blocks traversed by p.

4.3 Coping with Broadcast Traffic

Broadcast traffic can be a scalability concern. We take advantage of the fact that each SCT limits the

[2] Readers familiar with the proposal in [6] for a fabric abstraction for SDN, will notice that our ISM is an instance of such fabric, that spans over legacy devices.

broadcast domain size, and we rely on SDN capabilities to enable in-network ARP and DHCP proxies as shown in [17]. We focus on these important bootstrapping protocols as it was empirically observed that broadcast traffic in enterprise networks is primarily contributed by ARP and DHCP [17, 25].

Last, we note that in the general case, if broadcast traffic must be supported, the overhead that Panopticon produces is proportional to the number of SCTs in a cell block which, at worst, grows linearly with the number of SDN ports of a cell block.

4.4 Tolerating Failures

With Panopticon, we decompose fault tolerance into three orthogonal aspects.

Reusing existing STP mechanisms. Within an SCT, Panopticon relies on standard STP mechanisms to survive link failures, although to do so, there must exist sufficient physical link redundancy in the SCT. The greater the physical connectivity underlying the SCT, the higher the fault-tolerance. Additionally, the coordination between SDN controller and legacy STP mechanisms allows for more flexible fail-over behavior than STP alone.

When an SDN switch at the frontier $\mathcal{F}$ of SCT notices an STP re-convergence, the SDN controller adapts the forwarding decisions at $\mathcal{F}$'s SDN switches to restore per-end-point connectivity as necessary. This may involve assigning a given end-point to a different frontier switch to restore its connectivity, or to re-balance traffic. A similar scheme can address link failures within the Inter Switch Mesh (ISM).

SDN switch and link failure. When SDN switches and/or their incident links fail, the SDN controller recomputes the forwarding state and installs the necessary flow table entries. Furthermore, precomputed failover behavior can be leveraged as of OpenFlow version 1.1. Finding an efficient and lightweight solution is part of our ongoing work.

SDN controller robustness. Third, the SDN control platform must be robust and available. To this respect, previous work [18] demonstrates that well-known distributed systems techniques effectively achieve this goal, although the implications of inconsistent network views remain under-explored [19].

5. COST-AWARE SDN DEPLOYMENT

In principle, the Waypoint Enforcement policy can be obtained with a single SDN switch by forwarding all traffic through this switch. However, this solution is untenable for several reasons, including: (*i*) many end-to-end paths may become excessively long, (*ii*) bottleneck links may experience severe congestion, (*iii*) the forwarding state, bandwidth, and performance requirements of the SDN switch may be stretched beyond feasible limits, and (*iv*) it introduces a single point of failure into the network.

Therefore, in this section we show how to *upgrade to an SDN network* in stages in a *cost-aware and efficient manner*. We present an upgrade planning tool that is given an enterprise campus network topology $G_0 = (\Pi \sqcup \mathcal{L}_0, \mathcal{E})$, where $\mathcal{L}_0$ denotes the set of legacy switches. The *goal* of our planning tool is to select a subset of switches $\mathcal{S} \subseteq \mathcal{L}_0$ for SDN deployment and return the upgraded network $G^+ = (\Pi \sqcup \mathcal{S} \sqcup \mathcal{L}, \mathcal{E})$, where $\mathcal{L} = \mathcal{L}_0 \setminus \mathcal{S}$.

We next describe the parameters, including a simplified switch price model and the properties of desirable solutions. We then develop an optimal upgrade algorithm Opt and devise two heuristic algorithms Deg and Vol to solve it efficiently.

5.1 Tunable Parameters

To accommodate different designers' needs, our tool exposes several parameters: (*i*) priorities for upgrading end-points to SDN ports, (*ii*) switch prices, and (*iii*) several thresholds to encompass resource constraints of VLAN IDs, link utilizations, and flow-table entries.

SDN Port Priorities. Due to practical constraints (*e.g.*, monetary budget or capacity) it may not be feasible to find a partial SDN deployment where all end-points are SDN ports $\Pi^\bullet$ (*i.e.*, policed by an SDN switch). Also, a network administrator may not even wish to police certain end-points Π°, as others take precedence based on specific considerations including budget constraints, security, and reachability ($\Pi = \Pi^\bullet \sqcup \Pi^\circ$). Therefore, we allow a designer to specify (*i*) the end-points that *must* be SDN ports ($\Pi^\bullet_1 \subseteq \Pi^\bullet$), and (*ii*) the end-points that, on a best-effort basis, *should* be made SDN ports ($\Pi^\bullet_2 \subseteq \Pi^\bullet$). Note that $\Pi^\bullet = \Pi^\bullet_1 \sqcup \Pi^\bullet_2$. End-points Π° are ports that may violate the waypoint enforcement policy.

Switch Price Model. The price $\gamma(u)$ of an SDN switch $u \in \mathcal{S}$ depends on features and capabilities such as the number of switch ports $\Pi(u)$, port speeds Ψ, and switch fabric latency and capacity. Also, the price depends on the number k of distinct traffic flows (*e.g.*, source-destination MAC pairs or IP 5-tuples) it must handle: A larger k implies higher TCAM, memory and CPU requirements. Hence, we model the cost of a switch as the function $\gamma(u) = f(\Pi(u), \Psi, k)$. To this end, we assume that the network designer has a *price table* specifying switch prices together with their features and capabilities. Our operator survey (cf also Section 2) confirmed the validity of the price assumptions.

Link Utilizations. When upgrading the network anyway, the designer may choose to ensure a certain degree of link capacity *over-provisioning* w.r.t. the estimated traffic matrix. A tunable safety margin ϵ can be used to influence the individual switch upgrade and path choices to ensure the upgraded network maintains at least an ϵ percentage of the capacity of all links (or per link) as slack.

VLAN IDs and Flow Table Entries. Within each cell block, one may want to specify a certain upper bound on the percentage of used VLAN IDs. That is, given a maximal number of usable VLAN IDs $t_{\max}$ and a percentage ν, the number of VLAN IDs t used in a given cell block should not exceed $\nu \cdot t_{\max}$. Similarly, given a maximal flow table capacity $ft_{\max}$ and a threshold μ, switches may be chosen for upgrade such that the table at a new switch will not exceed $\mu \cdot ft_{\max}$.

5.2 Properties of Desirable Solutions

We want the following properties:

1. *Waypoint enforcement:* Any end-to-end path between two communicating end-points $\pi_1, \pi_2 \in \Pi^\bullet$ *must* be SDN-policed if at least one end-point belongs to $\Pi_1^\bullet$. Paths including an end-point $\pi \in \Pi_2^\bullet$ *should* be policed too if capacity and budget allow.

2. *Cost within budget:* The total upgrade cost does not exceed the given *budget β*.

3. *Feasible:* SDN switches have sufficient capacity to support all end-to-end paths assigned to them and expected link utilizations are within tolerable thresholds.

4. *Path stretch within limit:* As a metric to capture the impact of Waypoint Enforcement, we define *stretch* as $\rho(s,t) = \mathrm{d}^+(s,t)/\mathrm{d}_0(s,t)$, where $\mathrm{d}^+(s,t)$ is the length of the path $p^+(s,t)$ that satisfies Waypoint Enforcement in G^+ and $\mathrm{d}_0(s,t)$ denotes the path in the original network G_0. We require the stretch remain below a tolerable threshold.

5.3 Optimal Upgrade Algorithm

We now formalize an optimal cost-aware upgrade algorithm Opt based on *mathematical programming.* For presentation sake, we first introduce a path-based formulation that ensures Waypoint Enforcement, assuming every end-point must be a SDN port (*i.e.*, $\Pi_1^\bullet \equiv \Pi^\bullet$), but does not take into account traffic matrix volumes nor best-effort SDN ports $\Pi_2^\bullet$ and remaining ports Π°. We later extend our formulation to include these.

Opt is a *Mixed Integer Program (MIP)* based on the following constants and variables. Let the *binary constants* $l_{i,j}$ be 1 iff there exists a link $\{i,j\} \in \mathcal{E}$ in the legacy network G_0. The *integer constants* $\mathrm{d}_0(s,t)$ denote the path lengths (*e.g.*, w.r.t. hops) in G_0.

We define the *binary variable* $x_{i,j}^{s,t}$ that becomes 1 iff link i to j lies on the path $p(s,t)$ in the transitional network G^+. We also define y_i, a *binary decision variable* that is 1 iff switch i is to be upgraded.

One difficulty in the MIP formulation is to describe *paths*: in contrast to classical programs for shortest path or multi-commodity flow computations, we require that a path from $s \in \Pi^\bullet$ to $t \in \Pi^\bullet$ goes via a SDN switch $u \in \mathcal{S}$, *where u is a variable itself.* To avoid sacrificing the linearity of the program, our approach is to introduce a *binary variable* $u_i^{s,t}$ to determine whether path $p(s,t)$ goes through SDN switch i.

Opt can be used with different strategies, depending on which properties of the upgrade are strictly required and which are subject to optimization. In the following, we seek to minimize the total path stretch, subject to the upgrade budget β (Constraint (2)).

$$\min_{x_{i,j}^{s,t}, u_i^{s,t}} \quad \sum_{i,j \in \mathcal{L}_0; s,t \in \Pi^\bullet} \frac{x_{i,j}^{s,t}}{\mathrm{d}_0(s,t)} \tag{1}$$

such that

$$\sum_{i \in \mathcal{L}_0} \gamma(i) \cdot y_i \le \beta \tag{2}$$

$$\forall s,t \in \Pi^\bullet :$$

$$x_{i,j}^{s,t} \le l_{i,j}, \quad \forall i,j \in \mathcal{L}_0 \tag{3}$$

$$\sum_{j \in \mathcal{L}_0} (x_{i,j}^{s,t} - x_{j,i}^{s,t}) = \begin{cases} 1, \text{if } i = s \ \ (\forall i \in \mathcal{L}_0) \\ -1, \text{if } i = t \ \ (\forall i \in \mathcal{L}_0) \\ 0, \text{ otherwise} \end{cases} \tag{4}$$

$$\sum_{i \in \mathcal{L}_0} u_i^{s,t} = 1 \tag{5}$$

$$u_i^{s,t} \le y_i, \quad \forall i \in \mathcal{L}_0 \tag{6}$$

$$\sum_{j \in \mathcal{L}_0} (x_{i,j}^{s,t} + x_{j,i}^{s,t}) \ge u_i^{s,t}, \quad \forall i \in \mathcal{L}_0 \tag{7}$$

$$\sum_{j \in \mathcal{L}_0} x_{i,j}^{s,t} \le 1; \quad \sum_{j \in \mathcal{L}_0} x_{j,i}^{s,t} \le 1, \quad \forall i \in \mathcal{L}_0 \tag{8}$$

$$u_i^{s,t} = u_i^{t,s}, \quad \forall i \in \mathcal{L}_0 \tag{9}$$

$$\sum_i x_{i,j}^{s,t} + \sum_j x_{j,i}^{s,t} \le \mu^i \cdot ft_{\max}^i, \quad \forall i \in \mathcal{L}_0 \tag{10}$$

$$\sum_{\pi \in \Pi^\bullet \wedge \pi \in c} 1 \le \nu \cdot t_{\max}, \quad \forall c \in CB(G) \tag{11}$$

Constraint (3) ensures that a path can only use existing links, and Constraint (4) represents the flow conservation constraints along the switch paths. Constraint (5) states that there must be one SDN switch assigned to each path and Constraint (6) requires that the assigned switch is chosen for upgrade. Constraint (7) ensures that the assigned switch is included in the path; it takes effect when $y_i = 1$. Constraint (8) guarantees that a switch is only used once per path (loop-freedom) and Constraint (9) specifies path symmetry (this constraint can safely be omitted if not required). Constraint (10) guarantees that the number of used flow table entries (or MAC table entries for legacy switches) are within the allowed limit. Finally, Constraint (11) ensures that the number of VLAN IDs used in every given cell block is within a tolerable limit.

Finally, note that our formulation assumes the initial network consists only of legacy switches. In practice, after the first SDN partial-deployment phase has taken place one can reuse our planning tool by fixing the values of y_i to 1 according to the previously upgraded switches.

5.3.1 Extension to Traffic and Best-Effort SDN Ports

We now introduce traffic-awaress into Opt and then extend it to account for best-effort SDN ports.

Traffic matrix. Let $\kappa(e)$ denote the bandwidth of the link $e = \{i, j\}$ between switches i, j. We refer to the traffic matrix as $M_{s,t}$, which denotes the estimated demand between two end-points $s, t \in \Pi$. The following constraint enforces that every link utilization is at most the capacity minus safety margin ϵ:

$$\sum_{s,t\in\Pi} x_{i,j}^{s,t} \cdot M_{s,t} \le (1-\epsilon)\cdot\kappa(\{i,j\}),\quad \forall i,j \in \mathcal{L}\sqcup\mathcal{S}$$

As with any network planning approach, for better results with capacity constraints satisfaction, we suggest using a long-term traffic matrix, which we assume to be sufficiently stable. In an enterprise network, we expect it to be the norm that the utilization of every link is monitored (*e.g.*, via SNMP counter) at a fine level of granularity over a time scale spanning several months. With these measurements one can leverage existing methodologies (*e.g.*, see [21,33]) to obtain a traffic matrix.

Best-effort SDN ports. For end-points in $\Pi_2^\bullet$, we introduce the *binary variable* $h(\pi)$ that is 1 iff end-point $\pi \in \Pi_2^\bullet$ becomes a SDN port. This variable is used to disable Constraints (5) and (7) through an additional helper variable defined similarly to $u_i^{s,t}$. The objective function is extended to allow a parameter $\alpha(\ge 0)$ to trade-off the number of best-effort ports made SDN against path stretch, as follows:

$$\min_{x_{i,j}^{s,t},\,u_i^{s,t}} \sum_{i,j,s,t} \frac{x_{i,j}^{s,t}}{\mathrm{d}_0(s,t)} - \alpha\cdot\sum_{\pi\in\Pi_2^\bullet} h(\pi)$$

Of course, best-effort paths can be further prioritized (*e.g.*, ordered or weighted).

5.4 Heuristics

While Opt computes *optimal* upgrade solutions, the MIP formulation exhibits a high runtime. Alone, the problem of finding *just* the minimal stretch end-to-end paths through *given* waypoints is NP-hard [3]. Hence, to provide fast approximate results, we extend the planning tool with two heuristics, called Vol and Deg.

Both heuristics use a *greedy* strategy, in that Vol and Deg upgrade one switch after another, without backtracking. Both heuristics differ only in the *selection criterion* of the next switch to upgrade.

Vol is based on the intuition that a switch forwarding large volumes of traffic is likely to be on many shortest paths between end-point pairs, and an upgrade of this switch allows us to keep many existing paths even under Waypoint Enforcement. Hence, detours are avoided and the stretch remains small. To also take into account switch costs, Vol chooses the next switch u to upgrade by computing the max *volume-cost ratio* $\mathrm{vol}(u)/\gamma(u)$, where $\mathrm{vol}(u)$ denotes the traffic on physical links incident to u *before* the Waypoint Enforcement.

Deg tries to upgrade switches which are likely to yield many disconnected components (or cell blocks). This allows to reuse VLAN tags and improves the system scalability. Concretely, Deg chooses the next switch u to upgrade by computing the best *degree-cost ratio*, formally, the ratio $\max \Delta(u)/\gamma(u)$, where $\Delta(u)$ denotes the number of links incident to u. Indirectly, the hope is that the high-degree switch is at a location which does not change shortest paths by much.

Algorithm 1 gives the pseudo-code for both heuristics. (Since both greedy algorithms only differ in the switch selection, we present them together and highlight the line where they differ.) Note that due to capacity constraints (*e.g.*, the maximal flow table size at the switches or the capacity of the links), the first couple of switches upgraded by Vol and Deg may not yield a feasible solution yet. The choice of the next switch ignores these constraints: The algorithms return "infeasible" if no solution is found that meets the capacity constraints or if not all ports $\Pi_1^\bullet$ are subject to waypoint enforcement.

Note that similar techniques are used for many classical problems, such as Knapsack, Facility Location, Maximal Independent Sets, *etc.* We leave the formal analysis of the worst-case approximation ratios of the two heuristics for future research, however we observe in small-scale experiments that these heuristics are within 10% of the optimal algorithm.

```
Data: Legacy network $G_0 = (\Pi \sqcup \mathcal{L}, \mathcal{E})$
Result: Upgraded network $G^+ = (\Pi \sqcup \mathcal{S} \sqcup \mathcal{L}, \mathcal{E})$
$\mathcal{S} = \emptyset$;
while ($\sum_{u\in\mathcal{S}} \gamma(u) \le \beta$) do
    (* choose next switch $u \in \mathcal{L}$ to upgrade *);
    (* in case of Vol *);
    $u := \arg\max_{u\in\mathcal{L}} \mathrm{vol}(u)/\gamma(u)$;
    (* in case of Deg *);
    $u := \arg\max_{u\in\mathcal{L}} \Delta(u)/\gamma(u)$;
    $\mathcal{L} = \mathcal{L} \setminus \{u\}$, $\mathcal{S} = \mathcal{S} \cup \{u\}$;
    $G^+ := (\Pi \sqcup \mathcal{S} \sqcup \mathcal{L}, \mathcal{E})$;
end
if (feasible) return $G^+$; else return $\perp$;
```

Algorithm 1: Upgrade heuristics Vol and Deg.

Site	Access/Dist/Core	max/avg/min degree
LARGE	1293/417/3	53/2.58/1
MEDIUM	–/54/3	19/1.05/1
SMALL	–/14/2	15/3/2

Table 1: Topology Characteristics

6. EVALUATION

Our goal in this section is to evaluate different "hypothetical" partial SDN upgrade scenarios using three example campus networks. This simulation lets us (*i*) evaluate the scalability limits of our mechanism, namely, the available VLAN IDs and SDN flow table entries, and (*ii*) explore the extent to which SDN control benefits extend to the overall network. We focus on our simple heuristics Deg and Vol, and remark that more refined heuristics may result in more SDN enabled end-points for a given budget.

6.1 Methodology

To simulate a network upgrade with Panopticon, we must first obtain network topologies, traffic matrices, resource constraints, as well as cost estimates.

Topologies: Detailed topological information, including switch and router-level configuration, link capacities, and end-host placements is difficult to obtain: network operators are reluctant to share these details due to privacy concerns. Hence, we leverage several publicly available enterprise network topologies [30] and the topology of a private, local large scale campus network. The topologies range from SMALL, comprising just the enterprise network backbone, to a MEDIUM network with 54 distribution switches and unknown access characteristics, to a comprehensive large-scale campus topology derived from anonymized device-level configurations from 1713 L2 and L3 switches. Summary information on the topologies is given in Table 1. All links in each topology are annotated with their respective link capacities as well as their switch port densities.

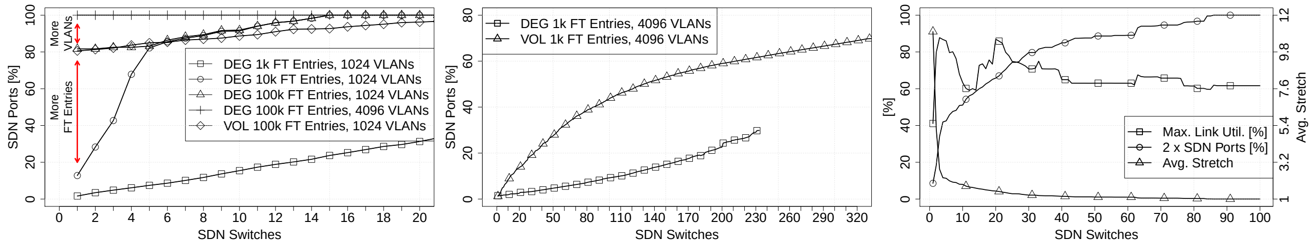
We use the SMALL network only when comparing our heuristics with the optimal solution computed by Opt (see § 6.3). We used the MEDIUM network as a "playground" during heuristic development and evaluation. However, as the results for the MEDIUM network are qualitatively equivalent to the ones for the LARGE network, for the remainder of the paper, we present only results for the LARGE network.

Focus on distribution switches: We distinguish between *access switches* (switches of degree one), *distribution switches*, and *core switches*. As core switches are typically very expensive and specialized, and hence are unlikely to be upgraded, we focus on distribution switches (in the following referred to as the *candidate set* for the upgrade). In case of the LARGE network, this candidate set has cardinality 417. Within this network, we reason about legacy network end-points which we wish to subject to SDN waypoint enforcement as distribution-layer switch ports which lead to individual host-facing access-layer switches. In this topology, we identify 1160 such end-points.

Traffic matrices: We use a methodology similar to that used in SEATTLE [17] to generate a traffic matrix based on actual traces from an enterprise campus network, the *Lawrence Berkeley National Laboratory* (LBNL) [25]. 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 switch ports periodically. We aggregate this data to obtain estimates of traffic matrix entries. More precisely, for all source-destination pairs in each sample we estimate the load they impose on the network over 10 and 60 minute periods, respectively. We note that the data contains sources from 22 subnets.

To project the load onto our topologies, we take advantage of the subnet information to partition each of our topologies into subnets as well. Each of these subnets contains at least one distribution switch. In addition, we pick one node as the Internet gateway. Each end-point in every cluster is associated—in random round-robin fashion—with a traffic source from the LBNL network. Then 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 the designated gateway node. By picking a different random assignment, we generate different traffic matrices which we use in our simulations. Before using a traffic matrix we ensure that the topology is able to support it. For this purpose we project the load on the topology using shortest path routes.

Resource constraints: Most mid- to high-end enterprise network switches support 1024 VLAN IDs for simultaneous use. The maximum number of VLAN IDs expressible in 802.1Q is 4096. Accordingly, we use both numbers as parameters to capture VLAN ID resource constraints, one realistic; one optimistic. Most current OpenFlow capable switches support at least 1,000 flow table entries while many support up to 10,000 entries across TCAM and SRAM. Bleeding edge devices support up to 100,000 flow table entries. Again, we use all numbers as possible flow table capacity resource constraints, one conservative, one realistic; one optimistic. Unless mentioned otherwise, we assume a setting where flow table entries are only persistently (pro-actively) populated along paths that are used (rather than for all potential src-dst endpoint pairs). The results presented in this paper include both optimistic as well as conservative slack factors. With regards to the link utilization when enforcing the waypoint enforcement pol-



(a) Realistic scenario (b) Worst-case scenario (c) Link utilization and path stretch

Figure 4: (a): Percentage of SDN-policed ports as a function of upgraded switches, available flow table entries (FT), and available VLAN IDs. (b): Percentage of SDN-policed ports in a scenario where flow table entries are very scarce and tables are populated along *all* possible paths. (c): Link utilization and average stretch trade-off as a function of the number of SDN switches.

icy of SDN ports, we do not consider a partial-upgrade feasible whenever any projected link exceeds 50% of its nominal link capacity.

Switch cost estimates: We use the following simplified switch cost model. A switch with more than 48 ports (up to 96) costs USD 60k. Switches with 48 or fewer ports are priced w.r.t. per link bandwidth. A switch supporting 10 Gbps links costs USD 30k, 1 Gbps cost USD 3k. These numbers correspond to currently available hardware.

6.2 How much SDN can I get for my budget?

The first key question we address is, "what fraction of an enterprise campus network can Panopticon operate as software-defined, given budget and resource constraints for upgrading legacy switches?"

Scenario 1: To investigate the interactions between flow table capacity and VLAN ID space we run 5 iterations (with different random seeds) of both heuristics. Both heuristics greedily upgrade one switch at a time subject to the resource constraints, and we track for each upgrade the number of end-points that can be waypoint enforced. Figure 4a plots the corresponding results with 95% confidence intervals for different resource parameters. Each point in the graph corresponds to a set of upgraded SDN switches and a specific fraction of upgraded SDN waypoint enforced end-points. The exact number of end-points that can be SDN waypoint enforced depends on the traffic matrix, however, the spread of the intervals is small.

Observations 1: The major take-away from Figure 4a is that switch flow table capacity plays a dominant role with regards to what fraction of the enterprise legacy network can be SDN-enabled. In the optimistic case when the switch supports 100,000 entries and 4,096 VLAN IDs the whole network can be SDN enable by upgrading just a single distribution switch while ensuring reasonable link utilizations. This case, however, is unrealistic.

With more realistic resource assumptions, namely 10,000 flow entries and 1,024 VLANs over 80% of the legacy network end-points can be SDN enabled with just five switches. This corresponds to upgrading **just** 1.2% of the distribution switches of the LARGE network.

For the most conservative case (1,000 flow table capacity and 1,024 VLAN IDs) we must upgrade a larger fraction of the legacy network in order to get a reasonable pay-back in terms of SDN enabled ports. 15 switches allow us to way-point enforce roughly 20% of the end-ports.

In this scenario, VLAN resource constraints play very little role, as the number of large number of poorly-endowed switches leads to more connected components and enables better VLAN ID reuse. Also, note that the difference in performance for the two heuristics for this scenario is relatively minor.

6.3 Which heuristic should one use?

Ideally, it would be optimal to always solve the Opt for each scenario. Due to high runtime complexity, however it may not be practical. For the SMALL and MEDIUM networks, the runtimes of our un-tuned Opt solver on modern hardware are in the order of multiple days, while the performance of the heuristics is minutes. Given our observation that heuristic results are within 10% of the Opt, we next focus on comparing the capabilities of the heuristics.

Scenario 2: Accordingly, the second experiment evaluates what fraction of the end-points can be made SDN waypoint enforced ports under severe flow table scalability limitations. We define a "worst case" where flow-table capacity is very small (1k), and enforce an all-to-all traffic matrix model where flow table entries are required for all paths between end-points; the number of VLAN tags is set such that it does not pose any constraint. We use both Deg and Vol.

Observations 2: The major take away from Figure 4b

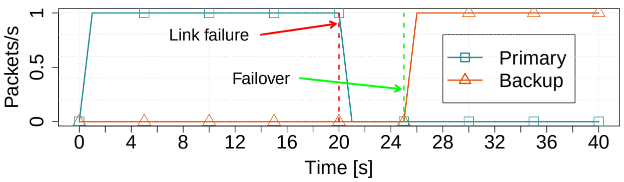


Figure 5: Panopticon SCT fail-over mechanism

is that Vol can significantly outperform Deg when flow table capacity is the major constraint. Vol performs better, and initially, approx. 5% upgraded switches are sufficient to police almost 20% of the ports. However, the marginal utility of additional upgrades declines; eventually, we achieve 70% SDN ports. The intuition behind this is that Deg upgrades switches in decreasing order of their degree. These high-degree switch typically serve more (but smaller) traffic flows while the high-volume switches serve less (but larger) flows. Accordingly, Vol needs a smaller number of flow entries to capture a larger fraction of the traffic. Indeed, for this harsh scenario, no iterative GREEDY upgrade solution to achieve fully upgraded SDN network may exist at all. Finding better heuristics is the subject of our future research.

6.4 How will Panopticon affect my traffic

Scenario 3: Our third experiment investigates the impact of SDN waypoint enforcement on traffic. We start with optimistic resource constraints (1024 VLAN IDs and 10K flow table entries) conservative traffic matrix, namely scaling the amount of traffic by two. This scenario has potentially the most severe impact on link utilization and path stretch.

Observations 3: Figure 4c plots the maximum link utilization over all links (left y-axis) as the number of upgraded switches increases. In addition, it shows the average path stretch across all waypoint enforced end-to-end paths of the traffic matrix (right y-axis). The major take away from Figure 4c is that Panopticon performs rather well in terms of both link utilization and stretch. While upgrading a single switch may yield a large average stretch, upgrading a dozen SDN switches (less than 3% of all distribution switches) reduces the stretch to less than 10%. Moreover, the maximum link utilization stays at reasonable levels. When only a small number of legacy switches are upgraded, the maximum link utilization may exceed 80%. However, as the number of upgraded switches increases the maximum link utilization stabilizes to roughly 60%.

6.5 Evaluating a Panopticon prototype

To cross-check certain assumptions on which Panopticon is founded, we created a prototype OpenFlow controller which implements the key functionalities of legacy switch interaction. The primary goal of our prototype is to demonstrate feasibility for legacy Switch interaction – namely the ability to construct and respond to link failure events and other behaviors within the SCT.

We design a simple experiment run in Mininet [12] which demonstrates the ability of an SDN programmable switch to react to an STP re-convergence, and adapt the network forwarding state accordingly. Figure 5 illustrates one such fail-over event within a topology modeled after Figure 3. Host A sends pings over switch 4 to host F until 20 seconds into the experiment, when a link failure between switch 1 and 4 is simulated and an STP re-convergence is triggered. The resulting BDPU updates are observed by the controller and the connectivity is restored over switch 2.

7. DISCUSSION

As hinted in the introduction, Panopticon exposes an SDN abstraction of the underlying partial-SDN deployment, however the SDN fidelity of the global network view is reduced to the set of upgraded switches. In this section we focus on what this means for the SDN programming abstraction and control applications.

Panopticon SDN vs. full SDN. As opposed to conventional links in a full SDN, links in Panopticon are pseudo-wires made up of legacy switches and links that run STP. Accordingly, the SDN controller must take into account the behaviors of STP within each SCT and ISM. For example, an STP re-convergence in an SCT can create the impression that a pseudo-wire "jumps" from one SDN switch to another. Conceptually, this may correspond to a physical topology change, *e.g.*, the behavior of a multi-chassis link access group in a full SDN.

Hiding the partial deployment from the app. As each Panopticon end-point is not necessarily attached to a SDN switch port, but rather to a group of SDN switch port–the frontier. Consequently, the SDN control platform may present or hide this information from the application to conceal the nature of the partial-deployment as needed. For example, if a control application wants to see the first packet of a flow to know from where the packet originated, the control platform can conceal the pseudo-wire nature of the legacy network, such that—to the application—the packet arrived directly from an end-point and not the physically neighboring legacy switch.

Which SDN applications are possible? The essence of SDN is to remove the application's awareness of the underlying physical network state and operate as a function over the global network view. As the control platform is responsible for morphing the Panopticon partial deployment into a consistent global network view, we do not foresee any restrictions for the control applications that can be supported in a Panopticon network as opposed to a full SDN. Naturally, early generation SDN applications that attempt to interact with legacy network islands will not work in Panopticon—but we do not view this as a limitation. Panopticon subsumes this functionality and thus deprecate these applications.

Scalability The key scaling factor of our system is the number of flow table entries–which is problematic when many wildcard matching rules are needed. Panopticon can leverage recent work on Palette [16] which proposes an approach to decompose large SDN tables into small ones and then distribute them across the network, while preserving the overall SDN policy semantics. Furthermore, as was demonstrated in [29], it is possible to use programmable MAC tables to off-load the TCAM and therefore improves scalability.

Why fully-deploy SDN in enterprise? Why should an enterprise ever move to a full deployment of SDN? Perhaps many enterprise networks do not need to fully deploy SDN. As our results show, it is a question of the trade-offs between budget limitations and resource constraint satisfaction. Our Panopticon evaluation suggests that partial deployment may in-fact be the right long-term approach for some enterprise networks.

8. RELATED WORK

Our overarching goal is to build a scalable network architecture that fundamentally integrates legacy and SDN switches, while exposing an abstract global network view to the benefits of the control logic above. As such, Panopticon differs from previous work in software-defined networking, evolvable inter-networking, scalable data-center network architectures, and enterprise network design and architecture.

SDN. In the enterprise, *SANE* [5] and consecutively *Ethane* [4] propose architectures where a centralized policing system enforces fine-grained network policy. Ethane overcomes SANE [5]'s obstacles to deployment by allowing compatibility with legacy devices. However, its integration with the existing deployment is only ad-hoc and the behavior of legacy devices falls out of Ethane's control. Panopticon supports policy enforcement with an architecture that integrates with the existing deployment through a rigorously-derived network design. Recently, Casado *et al.* in [6] propose a fabric abstraction for SDN to decouple the network "edge" from the "core" so as to introduce the necessary flexibility to evolve network design. We view Panopticon as a generalization of such abstraction for the specific dimension of exposing a global network view as a layer of indirection over an upgrade-able network substrate.

Evolvable inter-networking. The question of how to *evolve* or run a *transitional* network, has been discussed in many contexts. *Plutarch* [8], proposes an *inter-networking* approach that subsumes existing architectures such as IP. The notions of *context* and *interstitial function* are introduced to deal with (and exploit!) network heterogeneity. *Xia* [11] addresses the "narrow waist" design of the Internet as a barrier to evolvability. It natively supports the ability to evolve its functionality to accommodate new, unforeseen, principals over time. Admittedly, Panopticon has more modest aims. Still, we believe it provides a reference demonstration for operating transitional enterprise networks.

Scalable data-center network architectures. There is a wealth of recent work towards improving data-center network scalability. To name a few, Al-Fares *et al.* [2], *VL2* [10], *PortLand* [23], *NetLord* [22], *PAST* [29] and *Jellyfish* [28], offer scalable alternatives to classic data-center architectures at lower costs. However, as a clean-slate data-center architectures, these approaches are less applicable to transitional enterprise networks. Data-center network topologies are often highly regular, and far less embroiled in legacy hardware. The main objective of the data-center network is to support full bisection bandwidth for low- latency any-to-any communication patterns. In contrast, enterprise network structure is less homogeneous and grow "organically" over time. Enterprise networks typically exhibit low utilization, and the main objective is to provide connectivity while ensuring isolation policies. Waypoint Enforcement [5], as in Panopticon is one way to achieve this goal.

Enterprise network design and architecture. Sung *et al.* [30] propose a systematic redesign of *enterprise networks* with the goal of making a parsimonious allocation of VLANs to ensure reachability and provide isolation; this paper focuses on legacy networks only. The *SEATTLE* [17] network architecture uses a one-hop DHT host location lookup service to scale large enterprise Ethernet networks. However, such clean-slate approach is not applicable for the type of transitional networks we consider. Today's scalability issues in large enterprise networks are typically dealt with by building a network out of several (V)LANs interconnected via L3 routers [7, 15]. *TRILL* [26] is an IETF Standard for so-called *RBridges* that combines bridges and routers. TRILL bridges use their own link state routing protocol, improving flexibility and add routing. Similar to our approach, VLANs are used to support multi-path forwarding. While TRILL requires hardware and firmware changes, it can be deployed incrementally. However, we are not aware of any work discussing and rigorously evaluating the use of TRILL for efficient policy enforcement in enterprise networks, or *where* to optimally deploy RBridges.

To the best of our knowledge, there is no previous work on partial SDN upgrade.

9. SUMMARY

Managing and configuring enterprise networks is a non-trivial challenge given their complexity. While SDN promises to ease these challenges through principled network orchestration, it is nearly impossible to fully upgrade an existing legacy network to an SDN in a single operation. Accordingly, in this paper, we systematically tackle the problem of how to partially deploy SDN-enabled switches into existing legacy networks, and reap the benefits for as much of the network as is feasible, subject to budget and resource constraints.

Accordingly, we have developed **Panopticon**, an architecture and methodology for aiding operators in planning and operating networks that combine legacy switches and routers and SDN switches. Our evaluation highlights that our approach can deeply extend SDN capabilities into existing legacy networks. By upgrading just 3% of the distribution switches in a large campus network, it is possible to realize the network as an SDN, without violating reasonable resource constraints. Our results motivate the argument, that partial SDN deployment may indeed be an appropriate long-term operational strategy for enterprise networks. In future work, we plan to expand our prototype implementation to serve end-users, and further refine our algorithms.

10. REFERENCES

[1] Nicira Network Virtualization Platform. `http://nicira.com/en/network-virtualization-platform`.

[2] M. Al-Fares, A. Loukissas, and A. Vahdat. A scalable, commodity data center network architecture. In *SIGCOMM*, 2008.

[3] A. Bley. Approximability of unsplittable shortest path routing problems. *J. Netw.*, 54(1):23–46, 2009.

[4] M. Casado, M. J. Freedman, J. Pettit, J. Luo, N. McKeown, and S. Shenker. Ethane: Taking control of the enterprise. In *SIGCOMM*, 2007.

[5] M. Casado, T. Garfinkel, A. Akella, M. J. Freedman, D. Boneh, N. McKeown, and S. Shenker. SANE: a protection architecture for enterprise networks. In *USENIX Security Symposium*, 2006.

[6] M. Casado, T. Koponen, S. Shenker, and A. Tootoonchian. Fabric: a retrospective on evolving SDN. In *HotSDN*, 2012.

[7] Cisco. Campus Network for High Availability Design Guide, 2008. `http://www.cisco.com/en/US/docs/solutions/Enterprise/Campus/HA_campus_DG/hacampusdg.html`.

[8] J. Crowcroft, S. Hand, R. Mortier, T. Roscoe, and A. Warfield. Plutarch: an argument for network pluralism. *SIGCOMM CCR*, 33(4), 2003.

[9] N. Foster, R. Harrison, M. J. Freedman, C. Monsanto, J. Rexford, A. Story, and D. Walker. Frenetic: A network programming language. In *ICFP*, 2011.

[10] A. Greenberg, J. R. Hamilton, N. Jain, S. Kandula, C. Kim, P. Lahiri, D. A. Maltz, P. Patel, and S. Sengupta. VL2: A scalable and flexible data center network. In *SIGCOMM*, 2009.

[11] D. Han, A. Anand, F. Dogar, B. Li, H. Lim, M. Machado, A. Mukundan, W. Wu, A. Akella, D. G. Andersen, J. W. Byers, S. Seshan, and P. Steenkiste. Xia: efficient support for evolvable internetworking. In *NSDI*, 2012.

[12] 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.

[13] N. Handigol, B. Heller, V. Jeyakumar, D. Mazières, and N. McKeown. Where is the debugger for my Software-Defined Network? In *HotSDN*, 2012.

[14] B. Heller, S. Seetharaman, P. Mahadevan, Y. Yiakoumis, P. Sharma, S. Banerjee, and N. McKeown. ElasticTree: Saving energy in data center networks. In *NSDI*, 2010.

[15] Juniper. Campus Networks Reference Architecture, 2010. `http://www.juniper.net/us/en/local/pdf/reference-architectures/8030007-en.pdf`.

[16] Y. Kanizo, D. Hay, and I. Keslassy. Palette: Distributing tables in software-defined networks. In *INFOCOM*, 2013.

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

[18] T. Koponen, M. Casado, N. Gude, J. Stribling, L. Poutievski, M. Zhu, R. Ramanathan, Y. Iwata, H. Inoue, T. Hama, and S. Shenker. Onix: A distributed control platform for large-scale production networks. In *OSDI*, 2010.

[19] D. Levin, A. Wundsam, B. Heller, N. Handigol, and A. Feldmann. Logically Centralized? State Distribution Tradeoffs in Software Defined Networks. In *HotSDN*, 2012.

[20] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner. OpenFlow: enabling innovation in campus networks. *SIGCOMM CCR*, 38(2), 2008.

[21] A. Medina, N. Taft, K. Salamatian, S. Bhattacharyya, and C. Diot. Traffic matrix estimation: existing techniques and new directions. In *SIGCOMM*, 2002.

[22] J. Mudigonda, P. Yalagandula, J. Mogul,

B. Stiekes, and Y. Pouffary. NetLord: a scalable multi-tenant network architecture for virtualized datacenters. In *SIGCOMM*, 2011.

[23] R. Niranjan Mysore, A. Pamboris, N. Farrington, N. Huang, P. Miri, S. Radhakrishnan, V. Subramanya, and A. Vahdat. PortLand: a scalable fault-tolerant layer 2 data center network fabric. In *SIGCOMM*, 2009.

[24] ONF. Hybrid Working Group. `https://www.opennetworking.org/working-groups/hybrid`.

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

[26] R. Perlman, D. Eastlake, D. G. Dutt, S. Gai, and A. Ghanwani. Rbridges: Base protocol specification. In *Technical report, IETF*, 2009.

[27] M. Reitblatt, N. Foster, J. Rexford, C. Schlesinger, and D. Walker. Abstractions for network update. In *SIGCOMM*, 2012.

[28] A. Singla, C.-Y. Hong, L. Popa, and P. B. Godfrey. Jellyfish: networking data centers randomly. In *NSDI*, 2012.

[29] B. Stephens, A. Cox, W. Felter, C. Dixon, and J. Carter. PAST: Scalable Ethernet for data centers. In *CoNEXT*, 2012.

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

[31] R. Wang, D. Butnariu, and J. Rexford. Openflow-based server load balancing gone wild. In *Hot-ICE*, 2011.

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

[33] Y. Zhang, M. Roughan, N. Duffield, and A. Greenberg. Fast accurate computation of large-scale ip traffic matrices from link loads. In *SIGMETRICS*, 2003.

APPENDIX

THEOREM A.1. *Panopticon* ensures that: (1) Forwarding sets $FS(s,t)$ are *loop-free*. (2) End-to-end paths including an SDN port are subject to WAYPOINT ENFORCEMENT. (3) End-to-end paths are mutually isolated. (4) VLANs can be reused in different cell blocks.

PROOF. We prove the three correctness properties (1), (2), and (3), and the efficiency property (4) in turn.

Property (1): An end-to-end path from $s \in \Pi^{\bullet}$ to $t \in \Pi^{\bullet}$ is of the following form: $SCT(s) \to u_1 \in SCT(s) \to \text{ISM}(u_1, u_2) \to u_2 \in SCT(t) \to SCT(t)$. Therefore, loop-freeness follows directly from the loop-freeness of the VLAN STPs in $SCT(s)$, $\text{ISM}(u_1, u_2)$, and $SCT(s)$. Moreover, even if only a small subset of inter-SDN switch paths are realized as VLANs, and u_1 and u_2 are not directly connected, the SDN controller can make sure that sequence of VLANs used for forwarding traffic from u_1 to u_2 is loop-free.

Property (2): This property directly follows from the definition of the *Solitary Confinement Tree* (SCT). Since each $s \in \Pi^{\bullet}$ is in its own VLAN, the packets sent from s can only reach another VLAN via the SDN switches in the frontier $\mathcal{F}$ of s.

Property (3): The isolation property is due to the fact that each SCT contains exactly one end-point $\pi \in \Pi$, *and* that the inter-switch VLANs (in the ISM) to connect two SDN switches $u_1 \in \mathcal{F}, u_2 \in \mathcal{F}$ $(u_1, u_2 \in \mathcal{S})$ do not traverse any end-point $\pi \in \Pi$.

Property (4): By the *cell block* definition, all communication between two end-points $\pi_1 \in c_1 \in CB$ and $\pi_2 \in c_2 \in CB$ with $c_1 \neq c_2$ passes through the SDN switches in $\mathcal{F}(c_1)$ and $\mathcal{F}(c_2)$. Hence, all VLANs can be isolated from each other, and VLAN IDs reused in the different cell blocks. □