Software-Defined Networks: Incremental Deployment with Panopticon

Marco Canini, *Université catholique de Louvain*
Anja Feldmann, Dan Levin, and Fabian Schaffert, *Technische Universität Berlin*
Stefan Schmid, *Telekom Innovation Labs, Technische Universität Berlin*

Practically speaking, most enterprises migrating to a software-defined network (SDN) must do so incrementally. Panopticon offers an approach for designing and operating an interim hybrid network that combines both traditional and SDN switches by exposing a logical SDN abstraction.

Software-defined networks (SDNs) hold considerable promise for automating and radically simplifying computer network management—a manual, error-prone task today. However, an immediate shift from existing network architectures to SDNs is unlikely on a broad scale because, despite some notable real-world deployments such as Google's software-defined WAN,[1] for most organizations, software-defined networking remains a largely experimental technology. Consequently, enterprises increasingly view hybrid networks—those that combine an SDN with traditional network devices—as a transitional step toward full SDN adoption. Yet despite their importance from a practical standpoint[2] and the challenges they likely pose over the long term, research focusing on such hybrid environments has so far been modest.

From the outset, transition to an SDN should meet several specific goals:

- *Provide clear—and immediate—benefits.* Users will want to see an SDN's advantages with the first deployed switch. By contrast, Google's software-defined WAN required years to fully deploy, and benefits were realized only after the network switching infrastructure was completely overhauled. Most enterprises would find such a situation unthinkable. The earlier its return on investment, the more appealing SDN adoption will be viewed, and the more readily it will be accepted.
- *Minimize disruption while establishing confidence.* Even if existing switches already support SDN programmability, it is generally risky and undesirable for an enterprise to replace all production control protocols with an SDN control plane as a single "flag day" event. Rather, to increase chances for successful adoption, network operators must be able to deploy SDN technology incrementally, familiarizing users with its operation and building confidence in its reliability. This means starting with a small initial deployment

 Published by the IEEE Computer Society

that can gradually widen as it encompasses more network infrastructure and traffic.
- *Respect budgetary constraints.* For budgetary reasons, it is generally necessary for any network reengineering to occur in stages, with operators upgrading parts of the network over time.

One approach for dealing with these challenges is to abstract a hybrid network into a *logical SDN*—conceptually, a programmatic interface that exposes the network as if it were a full SDN deployment—providing a logically centralized control plane for the incrementally deployable SDN. Panopticon offers such a network architecture.

PANOPTICON

Panopticon realizes a programming interface for a hybrid network by exposing a logical SDN abstraction. Specifically, as SDN switches are incorporated gradually into an existing network over time, Panopticon allows network operators to abstract away traditional network devices and operate the network as an SDN comprised of SDN-capable switches only. With careful planning, SDN capability can ultimately be extended to every network switchport. Alternatively, because network-resource constraints may prevent the full SDN abstraction in practice, not every port needs to be controlled through the SDN interface.

Architecture

Panopticon's architecture works on the principle that each network packet traversing an SDN switch can be treated according to end-to-end network policies, such as access control, defined via an SDN programming interface. Moreover, traffic that traverses two or more SDN switches can be controlled at finer levels of granularity to enable further, customized forwarding (to facilitate load balancing, for example). Thus, Panopticon extends SDN capabilities to traditional switches by ensuring that all traffic to or from any operator-selected, SDN-controlled (SDNc) port is restricted to a "safe" end-to-end path—that is, a path traversing at least one SDN switch. We call this property *waypoint enforcement*.

Panopticon uses virtual LANs (VLANs) to restrict forwarding on traditional network devices and guarantee waypoint enforcement because VLAN capabilities are ubiquitously available on existing switches. However, because VLAN ID space is limited to 4,096 values (IEEE standard 802.1Q) and hardware often supports even fewer, we devised a scalable waypoint enforcement mechanism, the *solitary confinement tree*. An SCT corresponds to a spanning tree and connects an SDNc port to certain SDN switches. As such, each SCT provides a safe path between an SDNc port and every SDN switch it connects to.

A single VLAN ID is assigned to each SCT, ensuring traffic isolation and providing per-destination path diversity.

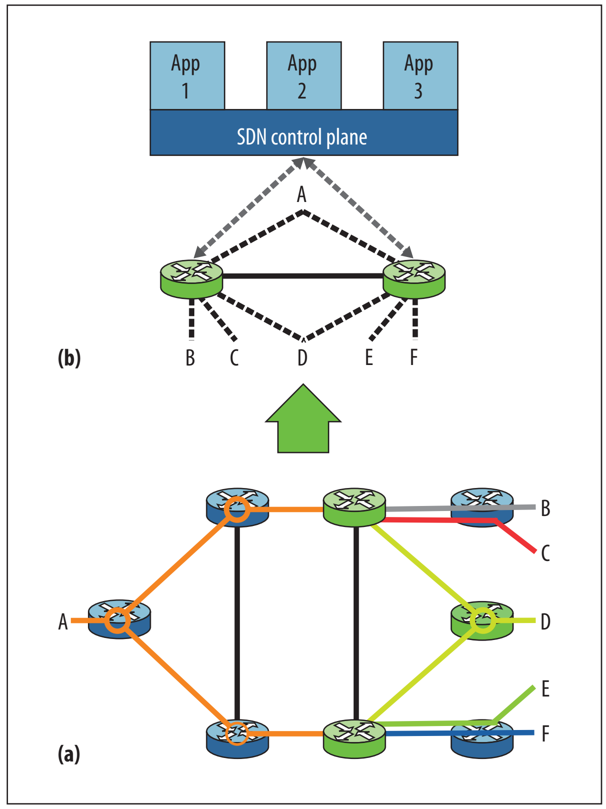

Figure 1. Panopticon overview. (a) In this sample eight-switch hybrid network, the green discs represent software-defined network (SDN) switches, and the blue discs represent traditional switches; overlaid are solitary confinement trees (SCTs) for the SDN-controlled (SDNc) ports A through F. Each SCT is realized by its own virtual LAN (VLAN) ID, represented via different colors. (b) In the corresponding logical SDN, the SDNc ports are virtually connected to SDN switches via "pseudo-wires," indicated by broken lines.

Scalability stems from the fact that VLAN IDs can be reused for "disjoint" SCTs—that is, SCTs that do not traverse a common traditional network device. Moreover, SCTs can be pre-computed and automatically installed onto traditional switches—for example, via the Simple Network Management Protocol. Re-computation is required, however, when the physical topology changes.

To illustrate, consider the eight-switch hybrid network shown in Figure 1a. Here, the orange links depict the SCT for SDNc port A, the gray links depict the SCT for SDNc port D, and the SCTs for the other ports are similarly color-coded. Figure 1b shows the corresponding logical SDN for the physical hybrid network that these SCTs enable. In this logical SDN, every SDNc port is connected to at least one SDN switch via a "pseudo-wire," a connection realized by its SCT.

SDN implementation

Panopticon enables the active burden of network management to be gradually transitioned away from legacy

Figure 2. Using the Panopticon approach, the number of SDNc ports accommodated as a percentage of the number of deployed SDN switches depends on how many VLAN IDs the existing system hardware supports.

devices and onto the SDN control plane—a transition that can be realized at individual switchport granularity. While Panopticon does not strictly mandate how the SDN control plane interacts with the existing traditional control plane, we envision that each SDNc port will first implement the same high-level policies in effect prior to the transitioning process—for example, preserving the original IP subnet address allocation. Thus, all policies governing traffic that originates from or is directed to SDNc ports can be defined exclusively at the SDN switches rather than at a combination of SDN and traditional devices. This strategy should effectively limit added complexity in managing the network during its transition to the SDN.

Still, both the SDN and traditional control planes must coexist during this transition. Consequently, within SCTs, Panopticon relies on standard Spanning Tree Protocol mechanisms, where necessary, to achieve loop freedom and tolerate link failures.

In addition, SDNc ports must be reachable from outside the logical SDN. For simple scenarios in which addressing within the logical SDN maintains compatibility with the existing IP subnet allocation, the traditional routing control plane and logical SDN can remain oblivious to one another. More commonly, though, addressing within the logical SDN violates the IP subnet allocation. To provide reachability in these instances, the SDN control plane could establish adjacencies with existing routing protocols, or routers could be configured with static routes for the IP subnets reachable through SDN switches. However, if at least one SDN switch is deployed in each IP subnet, it is possible to use a tunneling tprotocol such as Generic Routing Encapsulation to avoid interaction with the traditional existing routing control plane while ensuring reachability across IP subnets. Finally, to provide network applications with expected local network semantics, we rely on SDN capabilities to enable in-network proxies for Address Resolution and Dynamic Host Configuration protocols.

OVERHEAD AND FEASIBILITY

Panopticon's logical SDN abstraction does not come without cost: waypoint enforcement through SDN switches can in some cases lead to increased path lengths and require greater link utilization. Consequently, Panopticon presents operators with some resource–performance tradeoffs, particularly in determining how the scope and means of partial SDN deployment will affect traffic generally. Still, the opportunities Panopticon offers for improving network traffic control—for example, by enabling multipath forwarding for load balancing when sufficient path diversity exists—should not be discounted.

In navigating the deployment problem space, we have evaluated the approach's feasibility as follows. We consider a deployment feasible if the SDN switches have sufficient forwarding state to support all traffic policies they must enforce, and VLAN requirements to realize SCTs are within required limits.

We simulated various partial SDN deployment scenarios based on different resource constraints and traffic conditions by using a large campus network topology of roughly 1,700 switches, as we discuss in greater detail elsewhere.[3] These simulations allowed us to evaluate the feasibility space of our architecture, explore the extent to which SDN control extends to the entire network, and understand the impact that partial SDN deployment has on link utilization and path stretch.

As Figure 2 illustrates, the ability to accommodate more SDNc ports with a small number of SDN switches depends largely on the number of VLAN IDs the traditional existing hardware supports for use. Under favorable conditions, with 1,024 VLANs, full SDNc port feasibility requires as few as 33 SDN switches. However, VLAN ID availability is necessary to construct SCTs: when traditional switches support at most 256 VLANs, more than 140 SDN switches must be deployed before full SDNc port feasibility can be achieved.

To complement our simulation-based testing and further investigate the consequences Panopticon has for traffic, we also conducted a series of emulation-based experiments on portions of an actual enterprise network topology and further demonstrated the approach's system-level feasibility with a test-bed prototype.[3]

RECENT RELATED WORK

Our work contributes to a field that is attracting increasing attention from other researchers. Sugam Agarwal, Murali Kodialam, and T.V. Laksham, for example, have demonstrated effective engineering for traffic that crosses at least one SDN switch in a partial deployment.[4] Panopticon is an architecture that enforces this condition for all SDNc ports.

In a paper on software-controlled routing protocols presented at the 2014 Open Networking Summit, Laurent Vanbever and Stefano Vissicchio described mechanisms to enable an SDN controller to indirectly program L3 routers by carefully crafting routing messages.[5] We view this work as complementary to ours in that it could be useful to increase control over traffic whose paths include IP routers.

Ryan Hand and Eric Keller proposed an alternate approach to ours that they call ClosedFlow, which aims to enable SDN control over existing proprietary hardware by mimicking the fine-grained control available in OpenFlow.[6]

Finally, Vissicchio, Vanbever, and Olivier Bonaventure have discussed certain tradeoffs that arise within a diverse set of hybrid SDN models and argue that hybrid SDN architectures deserve more attention from the scientific community.[7] We agree.

We view Panopticon as a concrete step toward systematic, incremental deployment for SDNs. Accordingly, we have presented the approach at the Internet Research Task Force Working Group on SDN, and we plan to contribute our results to the ongoing discussions at the Open Networking Foundation's Migration Working Group.

We hope that our work will offer a helpful reference point for practical hybrid software-defined networking and contribute to ongoing standardization efforts. C

References

1. S. Jain et al., "B4: Experience with a Globally-Deployed Software Defined WAN," *Proc. 2013 Annual Conf. ACM Special Interest Group Data Communication* (SIGCOMM 13), 2013; http://conferences.sigcomm.org/sigcomm/2013/papers/sigcomm/p3.pdf.
2. *Migration Use Cases and Methods*, Migration Working Group, Open Networking Foundation, 2014; www.opennetworking.org/images/stories/downloads/sdn-resources/use-cases/Migration-WG-Use-Cases.pdf.
3. D. Levin et al., "Panopticon: Reaping the Benefits of Incremental SDN Deployment in Enterprise Networks," *Proc. 2014 Usenix Annual Technical Conf.*, 2014, pp. 333–345; www.usenix.org/sites/default/files/atc14_full_proceedings.pdf.
4. S. Agarwal, M. Kodialam, and T.V. Lakshman, "Traffic Engineering in Software Defined Networks," *Proc. 32nd IEEE Int'l Conf. Computer Communications* (INFOCOM 13), 2013, pp. 2211–2219.
5. L. Vanbever and S. Vissicchio, "Enabling SDN in Old School Networks with Software-Controlled Routing Protocols," *Proc. 2014 Opening Networking Summit* (ONS 14), 2014; http://vanbever.eu/pdfs/vanbever_hybrid_sdn_ons_2014.pdf.
6. R. Hand and E. Keller, "ClosedFlow: OpenFlow-like Control over Proprietary Devices," *Proc. ACM SIGCOMM Hot Topics in Software-Defined Networking* (HotSDN 14), 2014, pp. 7–12.
7. S. Vissicchio, L. Vanbever, and O. Bonaventure, "Opportunities and Research Challenges of Hybrid Software Defined Networks," *ACM Computer Communication Rev.*, vol. 44, no. 2, 2014, pp. 70–75.

Marco Canini *is an assistant professor in the Institute of Information and Communication Technologies, Electronics, and Applied Mathematics (ICTEAM) at Université catholique de Louvain, Belgium. His research interests include software-defined networking and large-scale and distributed cloud computing. Canini received a PhD in computer science and engineering from the University of Genoa. He is a member of IEEE and ACM. Contact him at marco.canini@uclouvain.be.*

Anja Feldmann *is a professor and dean of faculty in the Department of Engineering and Computer Sciences at Technische Universität (TU) Berlin. Her research interests include Internet routing and traffic analysis for network performance debugging. Feldman received a PhD in computer science from Carnegie Mellon University. In 2011, she was awarded the Gottfried-Wilhelm-Leibniz-Preis and the Berliner Wissenschaftspreis. Contact her at anja@inet.tu-berlin.de*

Dan Levin *received a PhD in computer science in 2014 from TU Berlin, where his research focused on software-defined networking. His work on hybrid incremental SDN deployment won first prize in the ACM graduate student research competition at SIGCOMM 2013. Formerly a software developer at Andeen Hagerling, Levin is a member of ACM. Contact him at dan@badpacket.in.*

Fabian Schaffert *received an MS in computer engineering from TU Berlin in 2014. His research interests include hybrid deployment of software-defined networks. Contact him at fabian@badpacket.in.*

Stefan Schmid *is a senior research scientist with Telekom Innovation Labs at TU Berlin, as well as a visiting professor at Centre National de la Recherche Scientifique, France. His research interests include distributed systems, especially the design of robust and dynamic networks. Schmid received a PhD in computer science from ETH Zürich. Contact him at stefan.schmid@tu-berlin.de.*