SD-Access: Practical Experiences in Designing and Deploying Software Defined Enterprise Networks

Jordi Paillisse
UPC-BarcelonaTech, Cisco

Marc Portoles
Cisco

Albert Lopez
UPC-BarcelonaTech, Spain

Alberto Rodriguez-Natal
Cisco

David Iacobacci*
BPM LLP

Johnson Leong*
Uber Technologies Inc.

Victor Moreno
Cisco

Albert Cabellos
UPC-BarcelonaTech, Spain

Fabio Maino
Cisco

Sanjay Hooda
Cisco

ABSTRACT

Enterprise networks, over the years, have become more and more complex trying to keep up with new requirements that challenge traditional solutions. Just to mention one out of many possible examples, technologies such as Virtual LANs (VLANs) struggle to address the scalability and operational requirements introduced by Internet of Things (IoT) use cases. To keep up with these challenges we have identified four main requirements that are common across modern enterprise networks: (i) scalable mobility, (ii) endpoint segmentation, (iii) simplified administration, and (iv) resource optimization. To address these challenges we designed SDA (Software Defined Access), a solution for modern enterprise networks that leverages Software-Defined Networking (SDN) and other state of the art techniques. In this paper we present the design, implementation and evaluation of SDA. Specifically, SDA: (i) leverages a combination of an overlay approach with an event-driven protocol (LISP) to dynamically adapt to traffic and mobility patterns while preserving resources, and (ii) enforces policies to groups of endpoints for scalable segmentation with low operational burden. We present our experience with deploying SDA in two real-life scenarios: an enterprise campus, and a large warehouse with mobile robots. Our evaluation shows that SDA, when compared with traditional enterprise networks, can (i) reduce overall data plane forwarding state up to 70% thanks to a reactive protocol using a centralized routing server, and (ii) reduce by an order of magnitude the handover delays in scenarios of massive mobility with respect to other approaches. Finally, we discuss lessons learned while deploying and operating SDA, and possible optimizations regarding the use of an event-driven protocol and group-based segmentation.

CCS CONCEPTS

• **Networks** → **Network design principles**; *Network mobility*; **Programmable networks**; *Network performance analysis*; *Routing protocols*.

KEYWORDS

Campus networks, Enterprise Networks, Software-Defined Networks, Reactive Protocols, Massive Mobility, Locator-Id Separation Protocol

ACM Reference Format:
Jordi Paillisse, Marc Portoles, Albert Lopez, Alberto Rodriguez-Natal, David Iacobacci, Johnson Leong, Victor Moreno, Albert Cabellos, Fabio Maino, and Sanjay Hooda. 2020. SD-Access: Practical Experiences in Designing and Deploying Software Defined Enterprise Networks. In *The 16th International Conference on emerging Networking EXperiments and Technologies (CoNEXT '20), December 1–4, 2020, Barcelona, Spain.* ACM, New York, NY, USA, 13 pages. https://doi.org/10.1145/3386367.3431288

1 INTRODUCTION

Current Enterprise Networks (EN) present a high degree of complexity derived from their organic evolution. Traditional ENs are built in a three-tier structure: access, distribution and core, with each layer leveraging a distinct set of protocols. The access-distribution segment uses L2 technologies such as Virtual Local Area Networks (VLANs), Spanning Tree Protocol (STP), and VLAN Access Control Lists (ACLs). On the other hand, the distribution-core segment is usually L3 with a combination of Open Shortest Path First (OSPF), Multiprotocol Label Switching with Label Distribution Protocol (MPLS-LDP), and IP ACLs. This design varies from deployment to deployment, but the end result are complex networks that can be running up to tens of different protocols across hundreds of switches. This makes it hard to adapt to new requirements, such as isolating IoT devices or stretching L2 domains across distributed locations. Even if some of these challenges might have been addressed by technologies designed for service providers, such as Virtual Routing and Forwarding (VRF), given the high port density required in typical campus networks, adopting the same technologies in enterprise solutions is not cost effective.

Specifically, current ENs lack three key elements. First, scalable endpoint mobility across all the enterprise facilities, to address the

*This work was performed while at Cisco.

CoNEXT '20, December 1–4, 2020, Barcelona, Spain

ever-increasing amount of roaming devices. Usually this is handled via sending all wireless traffic through centralized Wireless LAN (WLAN) controllers, which limits scalability, and reduces bandwidth. Second, simple to operate segmentation. The most common forms of segmentation in ENs are VLANs or VRFs, which do not scale well and can be difficult to configure at scale. Another example are IP-based ACLs, that over time can easily become long and difficult to map to the original intent. Third, simplified operations. Network administrators might configure each router individually, and, as we mentioned previously, have to deal with a myriad of different protocols. Although there exist more modern solutions [10, 25] that satisfy some of these requirements, we believe that the state of the art does not deal with scale and dynamism in a cost-effective manner, i.e. without requiring large capital expenditures (CAPEX).

CAPEX plays, indeed, a very important role in the context of Enterprise Networks. First, because of brownfield deployments: typically, network administrators do not want to upgrade *all* of their switches for new features. Since these are usually legacy devices with limited features, new network designs require a way to add new functionality without a forklift upgrade. Second, because deploying devices with reduced FIB size or CPU power decreases CAPEX but in turn means less powerful devices that require resource optimization.

In this paper we present the rationale, implementation, evaluation and experience matured in deploying SDA (Software Defined Access). Our objective with SDA was to design a solution that addressed the aforementioned requirements of modern EN. With this goal in mind, we leveraged a vast spectrum of research ideas, architectures and protocols produced by the community in the last decade [7, 9, 12, 14, 16, 23], and integrated them in a practical and deployable solution. First, we leveraged network overlays as discussed in the Fabric architecture [12], in the form of the Locator/ID Separation Protocol (LISP [15]). This offers three benefits: (i) we can upgrade existing deployments (brownfields) with minimal touch, (ii) their layer of indirection makes it easy to support L3 mobility, and (iii) they make segmentation with VRFs more scalable. Second, we applied the Software-Defined Networking (SDN) principle of centralized control to track endpoint location, and map endpoints to segmentation policies across the whole network. Finally, we chose a reactive protocol (LISP) to distribute network state to the data plane. In other words, we populate the switches forwarding tables only if required by the active traffic pattern. This reduces the overall switch requirements in terms of FIB size and CPU power, which results in reduced CAPEX.

We must remark that this paper is not meant to introduce a novel architecture, but rather an account of our experience and lessons learned in designing, and deploying modern enterprise networks. We describe how we combine state of the art techniques to realize a practical solution, and the lessons that we have learnt when operating SDA. We detail our experience through two real-life deployments. First, a medium-sized enterprise campus network serving around 450 endpoints that includes fixed hosts, mobile hosts, application servers, IoT devices, etc. We show that in this scenario, our reactive protocol optimizes data plane state with a 70% reduction of FIB entries in the data plane compared to solutions that store all the state in all routers (e.g. BGP). Second, a large warehouse where hundreds of robots are moving at speed to fulfill shipping orders, such as those ran by large on-line retailers. Here we evaluate the handover delay of 16,000 robots triggering 800 mobility events per second. Our solution achieves 5 times lower handover delay compared to existing approaches. Finally, we present our lessons learned from deploying SDA, such as reducing the initial connection delay due to the reactive protocol, coping with connectivity issues in the underlay or dealing with policy updates at scale.

2 REQUIREMENTS AND DESIGN DECISIONS

This section delves deeper into the requirements of current ENs, explains limitations of current strategies and discusses design decisions, summarized in table 1. We must remark that by *current strategies* we mean the common practice when deploying enterprise networks, rather than state of the art solutions. Section 6.1 describes the latter and compares them with our work. In addition, at the time of this writing there exist several commercial solutions from the major vendors aimed towards enterprise networks. In comparison, the key difference of our solution is the usage of a reactive protocol.

Resource optimization: Routing protocols commonly found in enterprises, such as OSPF, IS-IS or BGP make it difficult to reduce the FIB space without losing granularity due to IP prefix aggregation[1]. We approached this challenge by leveraging a reactive protocol (LISP in our implementation), rather than a proactive. Instead of pushing all routing entries beforehand to the routers, we only retrieve the necessary forwarding entries from a centralized server on-demand, and only for the routers that need them [9, 28]. We track endpoints by their IP address, so that routers download routes for the remote endpoints they need to reach, based on incoming traffic from local endpoints.

In addition, this reactive approach is helpful with mobility because we can reduce convergence time. This arises from the fact that a reactive approach reduces the churn generated by the location updates: we only notify the parties affected by a specific mobility event. We must remark that a reactive protocol presents several challenges, such as a potential initial delay for the establishment of flows (section 3.2.3), or detecting connectivity outages in the underlay (section 5 discusses our learnings in this space).

Mobility: the current trend of wireless first makes it critical to support a large amount of wireless endpoints. Traditionally, ENs handle mobility at L2 in a centralized way for both data plane and control plane. A gateway device (WLAN controller) acts as a sink for all traffic from all access points, performs access control, and re-injects it to the L3 network. This approach presents a serious scalability limitation because the gateway device becomes a bottleneck [2]. In addition, it creates triangular routing because all L3 traffic is forced to go to the gateway and then back to the actual destination.

SDA tackles mobility at layer 3 using network overlays [6, 33], specifically with the mobility features of LISP. We keep the wireless control plane centralized for authentication purposes, but we

[1] It is usual to leverage BGP prefix aggregation or OSPF Areas to reduce the overall number of routes in the network, at the price of less granularity.
[2] Although this particular concern can be alleviated with hierarchical controllers, it comes at the price of increased complexity and number of devices.

Table 1: Summary of strategies in current deployments, challenges, and design decisions

Requirement	Current Approach	Limitations	Our approach	Benefits
Resource efficiency	BGP and OSPF prefix aggregation	granularity at IP prefix level	Traffic-driven route learning	Reduced CAPEX, device-level granularity
Mobility	L2 centralized control and data plane	Scalability, triangular routing	L3 centralized control, distributed data plane	Increased scalability, optimized routing
Segmentation	VLANs and VRFs	Scalability, network-wide policies	Limited L2 stretching, map local VRFs to global VNs	Increased scale with less resources
Simplified administration	VLAN and IP-based ACLs	Error-prone, no mobility	Group-based policies	Smaller ACLs, end-to-end policy enforcement

let packets coming from the access point to be directly routed to destination. This distributed data plane greatly increases scalability.

Segmentation: traditionally, the most common form of segmentation in enterprise networks are Virtual Local Area Networks (VLANs [2]) and Virtual Routing and Forwarding tables (VRFs [34]). Despite their simplicity, the scope of a VLAN must be kept limited to prevent flooding of broadcast traffic or L2 forwarding loops, hence, they do not scale well. Regarding VRFs, they scale better than VLANs, but since each device has to be individually configured it is hard to implement global polices across the whole network. A direct consequence is that administration becomes too cumbersome as the number of VRFs increases. In addition, both of these approaches present similar limitations that make it hard to deal with mobility at scale.

SDA addresses these issues at different levels: for L2 segmentation [23, 31], we carefully stretch L2 domains (c.f. section 3.5). For L3, we still use VRFs, but map local VRFs to global virtual networks in order to handle L3 segmentation at scale. This way, network administrators only have to specify the virtual network for each endpoint [7]. Finally, we add a layer of indirection to ease administration, detailed in the next paragraph.

Simplified administration: a direct consequence of using network primitives such as IP addresses or VLANs for segmentation and access control is operational complexity [44]. In other words, network administrators have to translate business intent into IP addresses and ACLs and backwards. In the long run, this approach does not scale, is error prone, and increases complexity.

To overcome this problem, we make use of the well-established group-based paradigm [16, 41] to define ACLs between groups, instead of IP prefixes. First, the network operator defines a connectivity matrix among all groups. Then it adds endpoints to each group. On the network level, routers track each endpoint by its IP address and add a 16-bit tag representing its group, so they can enforce the connectivity rules in the matrix. The benefit is that network administration is radically simplified and common operations, such as IP address planning or ACL configurations can be automated.

3 DESIGN AND IMPLEMENTATION

In this section, we describe the design and implementation details of SDA. First we provide an overview, then we describe the control plane and data plane. Finally, we detail how we support endpoint mobility and L2 services.

3.1 Overview

On a conceptual level our design presents the usual data/control plane layer separation typical of SDN [29]. Figure 1 presents an overview of the design. We expose a simple declarative interface for network operators to define: (i) an endpoint's group and the Virtual Network (VN) where they can connect, (ii) the endpoint authentication data, and (iii) the connectivity matrix across groups of endpoints.

We store this information in the control plane in two different servers, a routing server and a policy server. The policy server authenticates endpoints, assigns them a group and configures the data plane routers with the required group rules from the connectivity matrix. The routing server keeps track of all endpoints by their IP address and provides routes upon demand by the data plane.

On the data plane, the overlay routers - hereafter referred to as *edge* routers - enforce the connectivity matrix and route packets to the corresponding edge router. A special *border* router provides access to external networks. Finally, endpoints can roam freely across edge routers.

3.2 Control Plane

We based the control plane design on a database-focused approach similar to [14], as opposed to more traditional designs [8, 18, 24]. In summary, the control plane consists of two logically centralized servers: policy server and routing server[3]. Instead of OVSDB [14], we leverage three protocols (LISP [15], SXP [39], and RADIUS [37]) to distribute the network state to the data plane. Table 2 presents a summary of all the control plane data.

3.2.1 Control Plane Policy Server. We offer two degrees of segmentation: Virtual Networks (VN) and the group connectivity matrix. These group rules are independent for each VN. On one hand, VNs offer strong isolation at a 'macro' level. An example is a hospital network, where we want to isolate the doctors, guest and medical devices networks, we never expect them to be able to communicate with each other. This is especially relevant to isolate legacy devices susceptible to attacks, e.g. an MRI machine running an outdated OS. In addition, it is a way to mitigate lateral spread attacks.

[3]The reason for this separation is that usually the host onboarding process needs both the endpoint credentials and group permissions (policy server), while the normal packet flow only needs the location mappings (routing server).

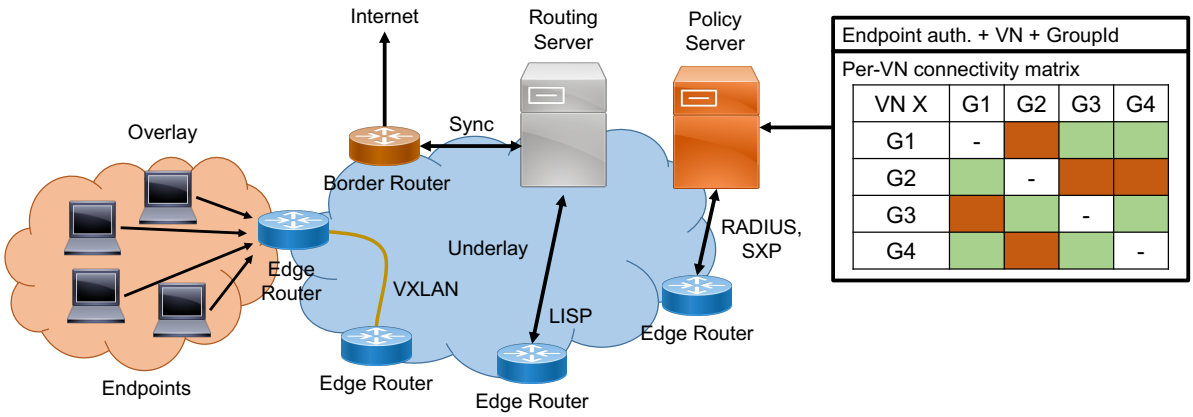


Figure 1: Global design

On the other hand, the group rules offer a 'micro' segmentation for finer grain control inside a VN. For example, this level of segmentation can separate different types of devices within a VN in Bring Your Own Device scenarios.

The policy server stores the connectivity rules from the connectivity matrix, and, for each endpoint: its authentication data, associated GroupId, and VN. GroupId and VN are 16-bit and 24-bit identifiers, respectively. The authentication data is variable since we support different RADIUS-based authentication protocols [37], both with EAP or without. We use a specific protocol, Scalable-Group Tag eXchange Protocol (SXP [39]) to distribute the GroupIds and connectivity rules to edge routers. From the network perspective, VNs are mapped to isolated routing-switching domains, while GroupIds are inputs to group-based ACLs.

3.2.2 Control Plane Routing Server. The routing server stores the endpoint location, i.e. pairs of overlay-to-underlay IP addresses plus its associated VN. The overlay IP is the IP used by endpoints, while the underlay IP is the IP of the edge router serving this endpoint. The remaining edge routers encapsulate traffic for such endpoint towards the underlay IP. After a successful device onboarding (section 3.3.1), or upon detecting a mobility event, edge routers update the underlay location of an overlay IP address. Edge routers also retrieve this mapping when they receive a connection request to a particular device. We leverage the control plane aspects of the Locator/ID Separation Protocol (LISP, [15]).

In a nutshell, the LISP control plane offers two messages: Map Request, to retrieve the underlay address of an overlay endpoint, and Map Register, to update the location of an endpoint, i.e. the overlay to underlay mapping. This way, the data plane can retrieve the overlay-to-underlay mappings that edge routers require to serve incoming traffic. In addition, the LISP control plane supports mobility, is well suited for SDN architectures [36], and accommodates different overlay address families apart from IP, e.g MAC addresses. This is especially helpful to support L2 services (section 3.5). Finally, some deployments may have more than one routing server or policy server for redundancy and load balancing.

3.2.3 Border Router. A drawback of using a reactive protocol such as LISP is the initial packet loss until the edge router downloads the route for a new destination. We have overcome this issue by installing a default route in all edge routers that points to the border router, and by synchronizing the routing state in the border with the information in the routing server (sync arrow in figure 1). This way, edge routers forward packets to the border until they finish the resolution process, during this time the border router forwards such packets to the destination.

The border router provides external network connectivity, and handles this extra load thus, playing a key role in the architecture. Hence, it is common that deployments have at least two or more border routers for redundancy. This is implemented by provisioning the aforementioned default route with the IP addresses of all the border routers (this use-case is contemplated in LISP protocol).

In addition, the border router is designed to handle traffic from all edge routers even if all traffic from edge routers is going outside the enterprise network (since traffic in enterprise networks is mostly north-south). This accounts for the worst case scenario (all endpoints communicating outside the enterprise network) and also for the extra load due to the traffic from edge routers during the resolution process.

However, we must remark that: (i) this sub-optimal routing through the border router is a transient phase, (ii) it only happens for the first flow between two particular endpoints, so that subsequent flows are unaffected, and (iii) based on our operational experience, the extra load in the border router due to this sub-optimal routing is a negligible fraction of its overall traffic. Moreover, we have observed that the duration of the TCP handshake is enough to establish the optimal path, hence mitigating the risk of out-of-order packets.

Finally, another benefit of the border router is in case that the edge router fills all of its FIB entries, due to a spike in traffic or a excessive number of route requests. In this situation, all the traffic for which entries cannot be allocated in the edge router would be redirected to the border router. Eventually, the unused entries in the edge router would disappear following a Least Recently Used (LRU) eviction policy.

Table 2: Types of Control Plane Data

Name	Key	Value	Updated by
Endpoint Data	Endpoint authentication info	GroupID, VN	Network Operator
Group Rules	Source GroupId + Dest. GroupId	Action (allow, deny)	Network Operator
Endpoint Location	VN + Overlay IP address	Underlay IP address (i.e. edge router)	Edge Routers

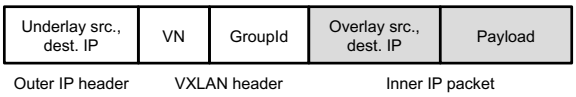


Figure 2: Packet Format

3.3 Data Plane

The data plane presents two distinct routers (figure 1): edge and border.

Edge Routers: perform four key functions. First, they encapsulate and decapsulate traffic to and from endpoints, respectively. Second, provide inter-VN isolation ('macro' segmentation). We implement such segmentation with VRFs: LISP populates the VRF tables, and each entry has an associated GroupId. Third, they detect roaming endpoints and update their location in the routing server. And fourth, enforce group permissions from the connectivity matrix ('micro' segmentation).

Border Routers: perform the same functions as edge routers, with two exceptions. First, their FIB table is synchronized with the routing server. In other words, they don't use a reactive protocol, rather they are subscribed to all route updates from the routing server [35]. And second, they have routes to other networks, e.g. Internet, datacenter. Because of this, border routers have more powerful CPU and larger FIB tables.

The data plane encapsulation leverages VXLAN with Group Identifier extension [40]. We chose this encapsulation over the native LISP data plane because of the need to encapsulate both L2 and L3 payloads (LISP supports L3 only). Additionally, in this VXLAN variant we can add the source GroupId in the group field (figure 2). Finally, the underlay is a network with plain IP connectivity that routes encapsulated packets between edge routers. The underlay routing is provided by either OSPF or IS-IS, we leverage MACsec [3] for packet integrity protection and confidentiality, and ECMP for redundancy [19]. The rest of this section describes how the network plugs a device for the first time (onboarding) and the standard packet flow.

3.3.1 Host Onboarding. When an endpoint connects to the overlay, the edge router detects it and starts the authentication process with the policy server (step 1 in figure 3). After a successful authentication, the edge router downloads the endpoint's VN, GroupId, and associated connectivity rules from the policy server (step 2). Specifically, it downloads the rules where the endpoint's group is the destination (c.f. section 5.5), stores locally the GroupId value, and associates it to the switch port where the endpoint is connected. Then it can assign an overlay IP address to the endpoint (step 4), obtained from a DHCP server. Finally, the edge router stores the location of the endpoint in the control plane, i.e. update the (VN + overlay IP, underlay IP) pair in the database (step 4).

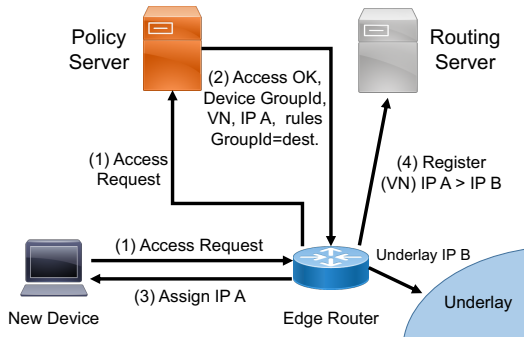


Figure 3: Host Onboarding Process

3.3.2 Packet Flow. On ingress, packets from endpoints are assigned their corresponding GroupId and VN (figure 4, ingress edge router). The router knows these values from the onboarding process. Then, it does a VN + overlay destination IP lookup in the VRF table for that VN. If there is no match, it will query the control plane database. This query returns the underlay IP address of the destination overlay IP. Finally, the packet is encapsulated towards the corresponding edge router, carrying both VN and GroupId.

On egress, the destination edge router decapsulates the packet and injects it into a two-stage pipeline (figure 4, egress edge router). First, it performs a VN + overlay destination IP lookup in the local VRF table corresponding to the VN in the packet. This query returns the output port and the associated destination GroupId. Each entry in the VRF has its associated GroupId, that is stored during the onboarding process. After authentication, the edge router creates an `(Overlay IP, GroupId tag)` association in the VRF table.

The second stage is an exact match lookup in an group-based ACL of (source GroupId, destination GroupId). This ACL enforces the aforementioned group rules. Finally, the router forwards the packet to the destination overlay IP address. We perform the policy enforcement on egress due to increased scalability (section 5.5).

3.4 Mobility Support

When an endpoint roams and attaches to a new edge router, the latter triggers the authentication process again, and registers the new location (figure 5, step 1). After this registration, the routing server sends a message to the previous edge router instructing it to (i) pull the new location data (steps 2 and 3), and (ii) forward all traffic for that particular endpoint to the new edge router. Hence, handover signaling is linear with the number of roaming endpoints,

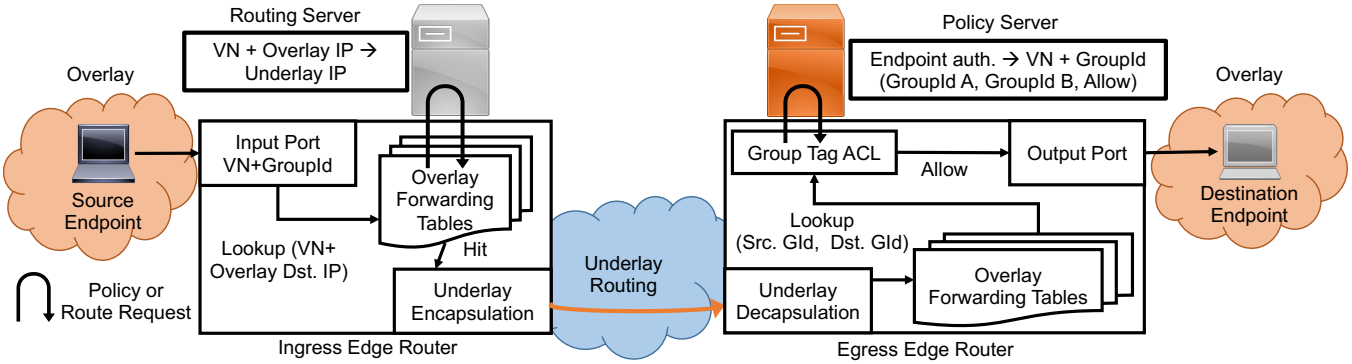


Figure 4: Ingress and egress pipelines

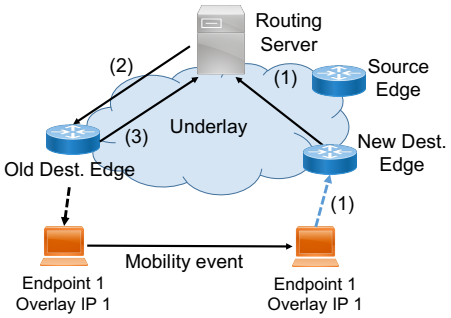


Figure 5: Endpoint mobility

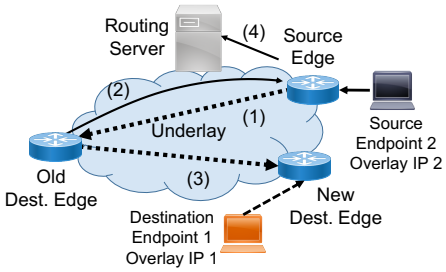


Figure 6: Updating stale entries via data plane messages.

as opposed to proactive protocols, in which it also depends on the number of routers. Section 4.4 quantifies the handover delay and compares to a proactive protocol.

Additionally, we apply the event-driven approach to update edge routers storing stale entries. To this purpose, we use a specific data-triggered control plane message (figure 6): when the old edge router receives traffic for the roaming endpoint (1) it sends a control message to the source (2), instructing it to retrieve the new location (4). At the same time, the old destination router forwards this traffic to the new destination router (3).

Regarding signaling scalability, this method depends on traffic patterns: if the roaming endpoint is very popular, we will have to update a significant portion of edge routers. On the contrary, endpoints that receive traffic from few sources, require less signaling. The advantage of this technique is that it is triggered by traffic, in other words, the control plane doesn't need to update *all* edge routers that have the stale location, but only those that *require* it. In addition, these control plane messages will be staggered over time as traffic from different senders will arrive at different times, thus spreading the signaling load in time.

Finally, the impact on traffic of the events in figure 5 (mobility event) and figure 6 (traffic redirection) is transient, suboptimal routing through the border or edge router, respectively.

3.5 Support for L2 services

In enterprise scenarios, it is common that some devices or users require L2 connectivity. Common use-cases are some forms of load-balancing, certain IoT devices, and basic services such as DHCP or service discovery[4].

In order to support such services, but avoid sending broadcast traffic to the entire network, our implementation leverages four elements: (i) VLANs (limited to the edge router ports), (ii) indexing endpoints by MAC address in the routing server (in addition to IP address), (iii) storing overlay IP to MAC pairs in the routing server, and (iv) deployment of L2 gateways in edge routers.

The combination of these four elements helps us to provide scalable L2 connectivity: first, VLANs limit broadcast domains; second, MAC address indexing locates endpoints of the same VLAN that are in different edge routers. Finally, L2 gateways absorb broadcast traffic and convert it to unicast: for instance, they capture ARP requests and perform a lookup in the routing server to find the MAC associated to the IP in the ARP request. Then they use this MAC to replace the broadcast MAC in the ARP request, creating a unicast L2 message. This message is injected in the L2 pipeline, which will use the MAC-to-underlay IP to encapsulate the request to the intended L2 MAC.

[4] A significant amount of applications rely on broadcast domains, e.g. Apple Bonjour

Table 3: Deployments used for evaluation

Deployment	# Border Routers	# Edge Routers	Number of endpoints
Building A	1	7	150
Building B	2	6	450
Warehouse	2	200	16,000 (emulated)

Table 4: Details of campus deployments

	Bldg. A	Bldg. B
Border Routers	1	2
Edge Routers	7	6
Floors	3	3
AP per floor	40	40
Total AP	120	120
AP per edge	~20	20

4 EVALUATION

We have implemented SDA in a commercially available line of routers, leveraging the protocols mentioned previously: LISP [15], RADIUS [37], and Scalable-Group Tag eXchange Protocol [39] for the control plane, and VXLAN [27] for the data plane. We have deployed our implementation in two different real-life scenarios: two campus networks with 150 and 450 endpoints, and a large warehouse (partially emulated) with massive mobility serving 16,000 emulated endpoints. Table 3 presents a summary of their characteristics.

We evaluated three key elements of SDA: first, the response of the routing server under stressful conditions, because it is critical in the process of establishing data flows. Second, quantifying the state optimization in the data plane due to the edge-border design. And third, assess the difference between an proactive and a reactive protocol in face of massive mobility events.

4.1 Routing Server Scalability

The routing server is a critical part of the design, because it allows establishing communication between any pair of endpoints. Because of this, we evaluated its performance depending on the number of routes and queries per second. In addition, sections 5.1 and 5.2 discuss how our solution reacts to network failures.

To this end, we setup a routing server implemented in a commercial virtual router with 8 GB RAM, 8 vCPU, on top of a virtualization platform with an 8-core 2.1 Ghz processor. We measured the delay to answer route requests and route updates with a script running in a local machine that sent 800 queries per second. We repeated the experiment for different number of routes configured in the routing server. Each query requested or updated a different route, in order to avoid optimizations due to intermediate caches. We consider the network delay negligible. Figures 7a and 7b present boxplots of the time required to answer an IPv4 route request and a route update, respectively, for four different number of configured routes in the routing server. The values are relative to the minimum delay of a routing server with only one route. Route requests are in the sub-milliseconds range and route updates in the single digit milliseconds range.

We can see that the delay is not dependent on the number of routes. Since this architecture is designed to store network state hierarchically, it makes it easy to implement the routing server with a Patricia Trie. The delay of this data structure depends on the number of bits of the keys, not the number of elements [30]. Based on the data collected for this particular test, an equivalent deployment using a similar setup should scale to at least 3k endpoints without noticeable performance degradation. Each endpoint requires registering 3 routes (IPv4, IPv6 and MAC addresses), then 10k / 3 = ~3k.

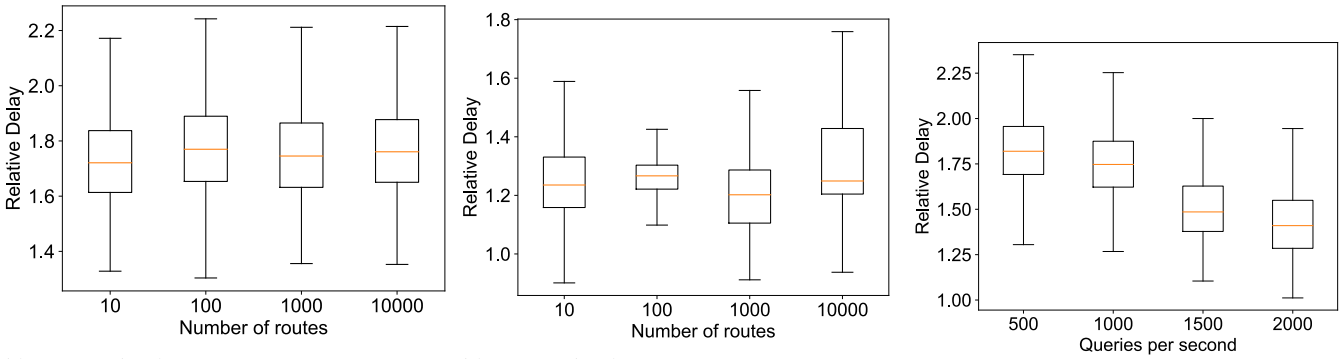
We chose to send 800 queries/s to the server since it is the peak requirement in the massive mobility scenario (section 4.4). We consider this a highly loaded server, however, in case we needed to increase such figure, the architecture scales horizontally and can deploy more routing servers. Then, we load balance across edge routers by grouping them and pointing each group to a different routing server for the route requests, and perform route updates on all servers.

Finally, using the same routing server as before, we repeated the previous experiment but for different number of queries/s. Figure 7c presents boxplots of the delay to answer IPv4 route requests for four different number of queries per second to the routing server. The values are relative to the minimum delay of all samples, and route requests are also in the sub-milliseconds range. Assuming a mobility pattern similar to the warehouse scenario (section 4.4) with 800 moves/s and that each move triggers 2 queries to the routing server, we conclude that the routing server could support this use case (800*2 = 1600 queries/s).

4.2 State Reduction

Reducing the amount of data plane state in edge routers is a key requirement of our design, since this allows for smaller FIB space, which in turn reduces CAPEX. In order to quantify the state reduction due to the reactive protocol, we counted the number of overlay-to-underlay IPv4 mappings in the FIB of the edge and border routers. We setup a VM that collected this data hourly from the router CLI. The routers were in two separate buildings (A and B), with three floors each, and providing network connectivity to between 200 and 500 users. Table 4 provides additional details about each deployment, and figure 8 shows the network topology. The control plane of border routers has a 4-core 2.4GHz, x86 CPU with 16 GB RAM, and edge routers a 4-core 1.8 GHz, x86 CPU with 8 GB RAM. Both of them use a custom ASIC for the data plane. The border-to-edge links are 10 Gbps and the edge-to-AP 1 Gbps.

Figure 9 shows the average number of FIB entries for the border and edge routers, for both buildings, and for three different weeks. We can see that, on average edge routers store less FIB entries than border routers: in building A edge routers carry only about 30% of the FIB entries present in border routers, while in building B this figure is as low as 6%. Table 5 presents also averages of FIB entries for border and edge routers for a period of 5 weeks. We can also see reductions in FIB entries, even if we calculate separate averages for day and nighttime. We can conclude that the reactive approach



(a) Boxplot (95%) of the delay of 10k route requests for four different number of configured routes. Relative values to a routing server with 1 route.

(b) Boxplot (95%) of the delay of 10k route updates for four different number of configured routes. Relative values to a routing server with 1 route.

(c) Boxplot (95%) of the delay of route requests for four different number of queries per second. Values are relative to the minimum of all.

Figure 7: Routing Server Performance Evaluation.

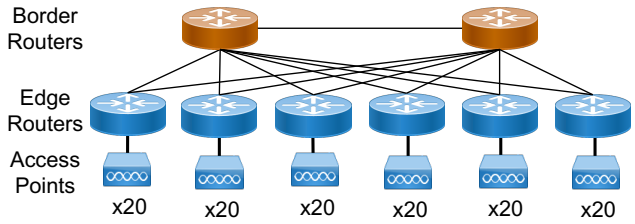


Figure 8: Campus Network Topology (underlay routers not shown for clarity)

Table 5: Average number of FIB entries for a 5 week period, for day work hours (9 am to 19 pm), and nighttime.

Bldg.	Border			Edge			Decr.
	All	Day	Night	All	Day	Night	
A	50	85	19	42	47	38	16%
B	291	362	227	34	42	27	88%

helps in optimizing data plane state, while the border router absorbs the convergence delay of the route resolution. Such decrease in data plane state in edge routers can be correlated to a reduction in CAPEX. Since there is a reduction of at least 70% of entries, the edge routers will need TCAMs or SRAMs proportionally 70% smaller, which can be translated to more cost effective hardware. In addition, their CPUs will need to process proportionally 70% less routing events.

It is also worth mentioning that the usage pattern of the FIB table on border routers shows a common daily and weekly pattern: during daytime in working week days routers host more routes than during nighttime and over weekends. This is due to the fact that the border router is always up to date with the information in the routing server regarding the endpoints in the deployment. Thus, the number of entries in the border router follows closely the presence of authenticated users in the network (i.e. users in the office). In contrast, edge routers cache routes learned on demand and may retain them during longer periods, even when users have left the office. In fact, the configured lifetime of such entries is 24h. We can clearly see this in building A (figure 9, top row), where edge routers seem to keep their routes for most of the time between workdays, but eventually clear they caches during weekend: approximately on Saturday afternoon, the number of entries in edge routers suffers a significant drop. However, the selection of the timeout does not play a critical role, although it has been previously studied [20]. It is more common that entries are updated or deleted before the 24h timeout due to mobility events or traffic triggers. We can appreciate this effect in building B, where edge routers follow the daytime/nighttime routine more closely. The reason behind this could be nighttime traffic patterns: when some endpoints leave the office at night, the remaining ones may initiate communication with those that left, that will trigger a route resolution with a negative result, and thereby deleting that FIB entry earlier. Such traffic patterns, especially the day-night rotation, and are in line with previous research on enterprise network traffic [22, 32].

Another relevant aspect of building B is a substantial amount of end-hosts that are permanently connected to the network and do not follow the day/night routine. Examples of such end-hosts are desktops and IoT devices (VoIP phones, cameras) that do not necessarily move with the users.

Finally, in larger networks (i.e. the number of endpoints increases), and assuming that the traffic pattern is the same, we would see a proportional increase in traffic in the border router.

4.3 Routing Server Load

We also measured the number of route requests to the routing server as the number of new FIB entries each hour. For each edge router, we compared the FIB hourly and counted the number of new entries. We assume that each new route is equivalent to a route request to the server. Weeks two and three of figure 9 show such results: building A requests correspond to all ethernet and IPv4 route requests, while building B also includes IPv6 requests. We

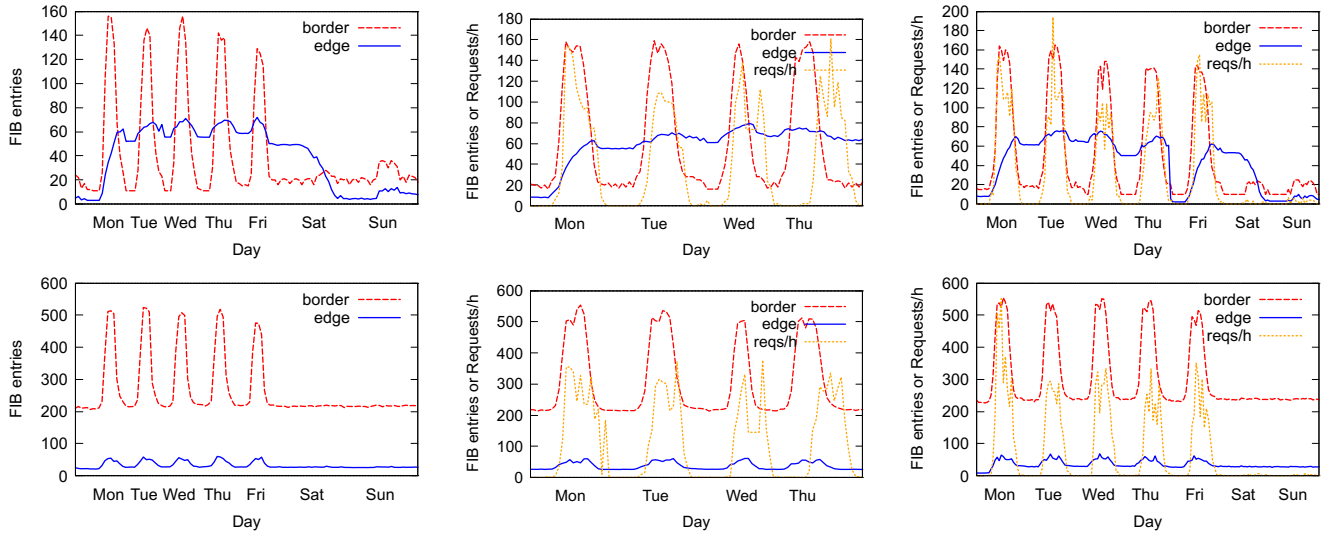


Figure 9: Number of FIB entries in border vs. edge router for three different weeks. Top row corresponds to building A, and building to B. Second and three weeks also present the number of queries per hour to the routing server.

can see that the number of requests follows closely the day-night pattern. In addition, the peak number of requests is around 600 requests/h. According to our previous evaluation, the routing server can easily handle this load.

4.4 Massive Mobility Events

In this experiment we focus on the handover delay of large mobility events for a reactive and proactive protocol. We recreated in the lab the specifications of a real life deployment, a large warehouse with hundreds of robots continuously roaming across WiFi access points. Figure 10 presents the topology: a commercial border router (4-core 2 GHz CPU, 8 GB RAM for the control plane and custom ASIC in the data plane), two commercial edge routers (4-core 1.8GHz CPU, 8 GB RAM for the control plane and custom ASIC in the data plane) and 198 edge routers emulated with a commercial traffic generator. The border router had an embedded routing server.

We configured the traffic generator to: (i) create unidirectional UDP traffic from the 200 edge routers towards the border router (yellow arrow in figure 10), (ii) send 1500 bytes packets from 16,000 emulated hosts, and (iii) generate 800 mobility events per second by changing the attachment port of the end hosts between the two physical edge routers. We selected this amount of mobility events because of the requirements from the warehouse scenario. In the data provided by the warehouse operator, a regular warehouse has 16,000 robots that move between APs at a rate of 10 moves per robot per minute. However, only 30% of these moves trigger a change of underlay IP address (e.g. when the robot moves to an AP in a different switch). That means an average of 0.005 IP mobility events per second per robot, and for a warehouse with 16,000 robots this translates to the 800 moves/s we simulated.

We measured the convergence time as the handover delay, i.e. the time since the emulated host is detached until traffic is restored after it attaches to the new edge router. In order to compare both

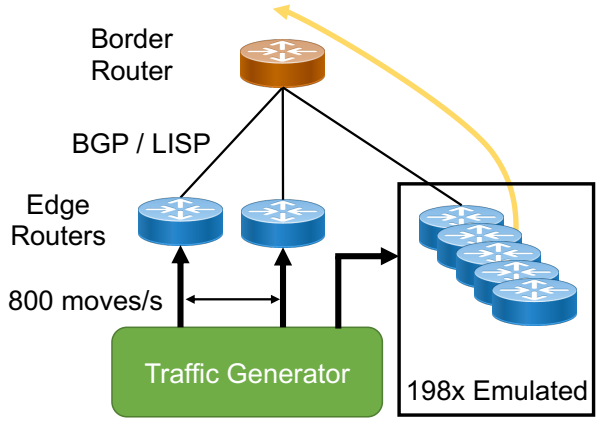


Figure 10: Warehouse Network Topology

proactive and on-demand approaches, we measured the handover delay in the same topology, but with two different control plane configurations: BGP as the proactive, and LISP as the reactive. In the BGP case we used a centralized route-reflector in the edge router to distribute route updates.

Figure 11 plots the CDF of the handover delay for BGP and LISP. All values are normalized to the minimum convergence time observed during the measurement process. We can see that the proactive solution takes around 10 times more to converge than the reactive one. The reason is that the proactive approach replicates the network update to all 200 edge routers, while the on-demand approach follows traffic patterns and only updates edge routers that have active traffic to roaming hosts.

Another important observation is that the variance of the handover delay is consistently higher in the proactive approach than

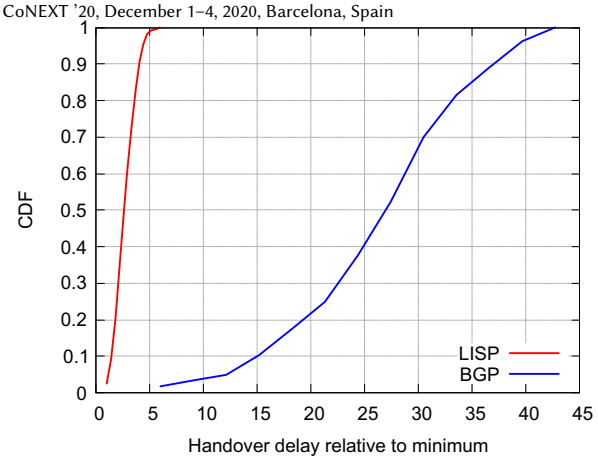


Figure 11: Handover Delay for Event-driven (LISP) and Proactive (BGP) Protocols

the reactive. Again, this is due to the fact that the reactive architecture selectively updates only routers that are actively sending traffic to the end-host that moved, while the proactive approach updates edge routers randomly, i.e. not by their need for such update. These results show that SDA's reactive approach can be beneficial in stressed environments such as automated warehouses or large gatherings with highly mobile end-hosts. Finally, since SDA can satisfy the aforementioned requirements for the warehouse scenario, it will also handle those of a campus network because the mobility requirements of the latter are less stressful than the warehouse.

5 LESSONS LEARNT

In this section we summarize several challenges and our learnings from implementing and deploying SDA in enterprise networks.

5.1 Underlay Connectivity Issues

In order for the overlay to work, there needs to be underlay connectivity. However, it is possible that an edge router fails or that an underlay IP changes, interrupting normal traffic flow. It can be challenging for the overlay to know the state of the underlay without explicit probing. To cope with these situations, edge routers monitor the address announcements of the underlay routing protocol (IS-IS or OSPF) to know about their reachability to underlay IP addresses of the other edge routers.

This way, when they detect a connectivity outage, they update their local forwarding table deleting such route and falling back to the default route to the border, until a new edge router registers the overlay address in the routing server.

5.2 Edge Routers Rebooting

One issue that can happen is a forwarding loop between the edge and border router. Assume the network is forwarding traffic, and an edge router reboots. It will start with an empty FIB for the overlay entries. When it receives traffic for one of its former endpoints, it will use the default route and forward it back to the border router, since it does not know yet its endpoints. The border router will forward this traffic to the rebooted router according to its current information, creating a forwarding loop.

Although this loop is transient and disappears once the edge router detects its endpoints, we rely on two mechanisms in such situation. First, consider that the edge router will not announce its underlay IP address through the underlay routing protocol while rebooting. This means that the other edge routers will remove the routes to the rebooting edge router, because they are continuously tracking the connectivity state of the underlay IP addresses. Second, the rebooting router will not recognize the incoming traffic, so it will send the data plane message we mentioned in section 3.4 to the originating edge router. This will trigger a refresh in the overlay FIB entries of the sending edge router.

5.3 Reducing OPEX

Network operators reported a decrease in network planning, which translated to a reduction in OPEX. This is due to the usage of a reactive protocol: since forwarding entries are downloaded on demand, it is not necessary to perform network planning and prefix aggregation, as opposed to other protocols. In addition, this makes it easier to re-use configuration templates in different devices.

This simplification in deployment is especially relevant for VLAN planning. Typically, operators have to manually specify the placement of every VLAN and subnet in each switch, in order to economize FIB usage. On the other hand, a reactive protocol optimizes such placement by downloading only the MAC addresses or subnets required by the traffic crossing that particular switch. In turn, this makes it simpler to add new switches to the network because they don't need additional network planning.

5.4 Benefits of the Routing Server

We built the centralized routing server as a Network Information Base (NIB) to store and distribute routes, but in line with common SDN practice, it is possible to use this centralized database for several network applications such as security, analytics, etc. For example, a security team used this server to perform duplicate address detection for IPv4 because they found this method more efficient than L2 flooding to detect duplicate hosts.

5.5 Selecting the Policy Enforcement Point

In this section we discuss a bandwidth vs. network state trade-off related to the deployment of group-based policies. Recall from section 3.2.1 that these are pairs of source GroupId, destination GroupId allowing or denying communication between two specific groups of hosts. During the design phase, we decided to enforce these policies at the *egress* edge router (as detailed in section 3.3.2), the reason being that on egress we save data plane state. This is due to the fact that the egress router only needs policies for the local destination groups of the endpoints that are attached to it. On the other hand, when enforcing policies on ingress, we would need policies for *all* possible destination groups. Note that we are assuming that we need the policies in the data plane before receiving traffic, in order to avoid additional delays when pulling them from the policy server. Conversely, if we enforce policies on ingress we can save bandwidth because we don't forward the part of the traffic that will eventually be dropped on egress due to deny rules. Hence, there is a trade-off between enforcing on ingress (save bandwidth) or enforcing on egress (save dataplane state).

In order to quantify the wasted bandwidth due to enforcing policies on egress, we analyzed the packet drops due to group-based

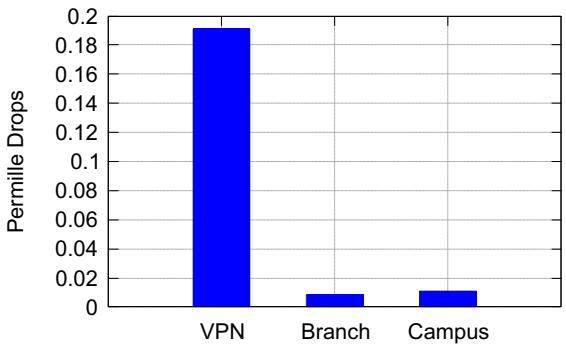


Figure 12: Permille hits on drop rules over all hits.

ACLs in a real-life deployment leveraging egress-based policy enforcement. This deployment is a medium-large enterprise network with a campus and a few branches. For our analysis, we looked at three different devices in this deployment: a branch router, an edge device in the campus, and a VPN gateway. Combined, these three devices were serving around 11,000 endpoints during the period we monitored them. Figure 12 presents the permille of dropped traffic for the period of 5 days. We can see that in the worst case the drop rate is extremely low: 2 out of each 10k packets. The VPN router has a significantly larger amount of drops than the other routers due to the fact that it receives all the traffic from remote users, which present a different usage pattern from the users in the office.

Since this deployment performs the policy enforcement on egress, we expected a significant percentage of drops. Surprisingly, we discovered that after a new policy is applied, there is a transient period with an increase in drops, but when endpoints (which are usually humans) realize they cannot access this particular destination, they stop requesting it. Hence, the operational experience in the most common enterprise use cases shows that *enforcing policies on egress does not impact significantly the amount of wasted bandwidth.*

An additional benefit of enforcing on egress is a simplification of signalling. When a group rule is updated, it is necessary to notify all the affected edge routers. However, this is not easy in our design, because requests to the control plane are only triggered by data plane events. For example, consider that the policy check is implemented on ingress and that the edge router has already learned the GroupId associated to a particular destination (figure 13, top part). Now suppose that the group associated to this destination endpoint is updated, the ingress router has no way to detect this update. Hence, we need a way to signal this change.

On the contrary, if the policy is enforced on egress, the (Overlay IP, GroupId) pair in the VRF is automatically updated, because the modification of endpoint data automatically triggers the authentication process again. In other words, on egress the (Overlay IP, GroupId) pair is always up to date because it is linked to the endpoints connected to that edge router. This way, we can avoid implementing an extra signaling mechanism, and the associated complexity.

5.6 Updating Policies

In our deployment experience, we found an interesting trade-off when updating policies: it can be more scalable (i.e. less signaling) moving users to different groups rather than directly updating the

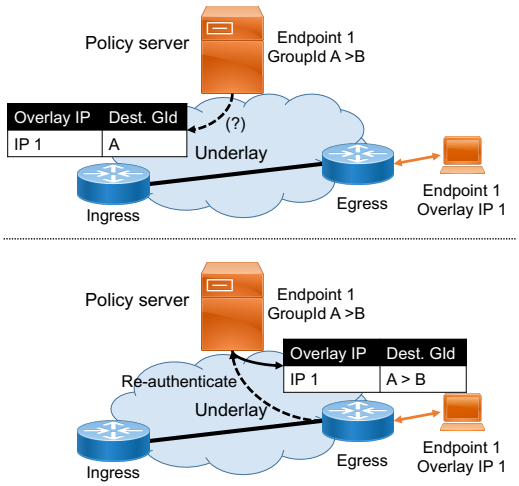


Figure 13: Policy Enforcement on Ingress (top) and Egress (bottom)

group-based ACLs. Indeed, this trade-off depends on the distribution of endpoints within groups of each particular deployment, i.e. few groups with large amounts of endpoints vs. high number of groups with few endpoints each. Thus, it is not always the case that changing the endpoint's group is more scalable, but here we present two examples form our experience:

Acquisitions: in case of an acquisition, the new employees are progressively moved through different groups until they get the same group as regular employees. The reverse also holds, when part of a company is sold, their users are moved to a group that is associated with more restrictive policies.

Service insertion: it is common that traffic has to go through middleboxes, e.g a firewall or a WAN optimizer. In some deployments, the SDA operators decide to update the group in the packets so that devices in the service chain decide whether to apply a policy or not. In other words, instead of applying different policies across the path for the same group, they change the group along the way so that different policies are applied across this same path.

6 RELATED WORK

6.1 SDN Pioneering Work

Ethane [10], one of the first SDN designs, presents numerous similarities with our proposal: it also targets enterprise networks, presents a similar architecture with a centralized controller, and supports incremental deployment. However, Ethane specifically focuses on three key elements, mostly around the control plane: (i) providing network access control, (ii) a rich high-level policy language, and (iii) controller design and fault-tolerance. Conversely, in this paper we pay more attention data plane related aspects such as network isolation, resource efficiency, and mobility.

Campus networks also motivated the inception of OpenFlow [29], that champions the decoupling of data plane and control plane, gives a strong focus to the interface between the router and the controller and defines and approach to implementing rich policies on capable switches. SDA in its turn gives more emphasis to ability

to support scalability in heterogeneous environments with devices with different capabilities.

Finally, SANE [11] offers a simple, high-level policy interface like the one used by SDA, but SANE tackles the problem in the border between L2 and L3 and does not trust the data plane routers.

6.2 BeyondCorp and Zero Trust Networks

In terms of securing the enterprise network, a closely related work is Beyond Corp [45], also known as the Zero Trust model. Like SDA, Beyond Corp focuses on access control between network endpoints, with an especial emphasis on user-to-server connections. On the other hand, it should be noted that networking improvements (e.g. seamless mobility, etc) are not in the scope of BeyondCorp and therefore this section only discusses how SDA relates to BeyondCorp in terms of enterprise security.

BeyondCorp offers a solid approach to build secure enterprise networks by means of keeping healthy endpoints and redirecting traffic through access proxies. By doing so, BeyondCorp presents a security model that is agnostic to the underlying networking infrastructure. While this is a reasonable approach in certain scenarios, in our operational experience, we have found that certain enterprise requirements are hard to meet with a BeyondCorp-only approach. First, while BeyondCorp protects the access to the enterprise network, its focus is on protecting the access to enterprise applications at layer 7. However, it is not always possible to redirect traffic through the proxies (e.g. L2 traffic). Second, despite enterprise efforts around building healthy fleets of devices, the reality we observe is that insecure devices are still present in typical enterprise networks (even more due to the explosion of different IoT devices). These endpoints make the BeyondCorp approach harder to implement and, in many cases, secure on-boarding and admission still need to be performed by the network infrastructure. Third, with only a BeyondCorp security model the overall network performance could be degraded by malicious actors attempting to get access (even if unsuccessful) or explicitly looking to disrupt the network operation. SDA operates lower in the stack and not only protects the connection of devices to the network but also the network infrastructure itself. We believe that SDA complements BeyondCorp and the combination of both could contribute to strengthen the overall security of an enterprise network.

6.3 Current SDN Related Work

The challenges of implementing SDN have been extensively discussed [38] for different kinds of networks: WAN [21], data-center [23], and cloud providers [13]. Regarding enterprise networks, we can find evaluations of SDN controllers in a campus network [43], or provisioning of corporate networks [17]. More related to our work, Software Resolved Networks [25] also leverages a reactive protocol (DNS extensions) and a centralized database, but requires involving endpoints when resolving routes, allows a wider range of policies than our proposal, and does not deal with mobility.

Other proposals for enterprise networks center their work on specific elements of an enterprise network, such as ACL configurations [44], systematic design of VLANs and ACL placement [42] or incremental SDN deployment [26]. Finally, Shortest Path Bridging [1] is also a technology suitable for enterprise networks that supports mobility, redundancy, and routing at L2. It is gaining momentum among several vendors [4, 5]. However, compared to our solution it does not support group-based policies, and segmentation is more complex because of the mapping between VLAN IDs and Service IDs.

7 CONCLUSION

This paper presents SDA, a solution designed for modern enterprise networks. The main goal of the architecture is supporting emerging requirements, with a strong focus on mobility, segmentation, and incremental deployment. At the same time, it provides scalability and optimizes data plane resources for heterogeneous environments with devices of diverse capabilities.

SDA leverages common practices in networking (centralized control, network overlays) and makes use of a reactive approach to distribute network information and support mobility. Experimental results show that, when compared with traditional approaches, SDA exceeds a 70% reduction in the overall forwarding state used by the network. Also, in very large deployments that need to deal with massive mobility events, network convergence is an order of magnitude faster than the status quo. For all these reasons we believe that SDA is a proper choice to address the unique requirements of modern enterprise networks that include scalable mobility, end-to-end segmentation, simplified administration, and overall resource optimization.

ACKNOWLEDGMENTS

The authors would like to thank Lorand Jakab for his invaluable help with the data collection scripts. Also Satish Kondalam regarding insights about customers, Satya Nanduri, Nirav Turakhia, and David Dai for the access to the deployments, Dipesh Patel, Ragupathi Rajavel, Raj Kumar, and Arash Albandi for the policy data, and Kevin Regan, Darrin Miller, and Mark Basinski for their help with the policy section. Thanks to Balaji Pitta Venkatachalapathy for the data collection, analysis and protocol tuning for the mobility use case. Thanks also to Pere Monclus for his valuable feedback on the manuscript, and Dino Farinacci for his LISP insights, inspiration, and vision. Finally, thanks to the anonymous reviewers and our shepherd for their feedback and guidance to improve our manuscript.

This work was partially supported by the Spanish MINECO under contract TEC2017-90034-C2-1-R (ALLIANCE) and the Catalan Institution for Research and Advanced Studies (ICREA).

REFERENCES

[1] 2012. IEEE Standard for Local and metropolitan area networks–Media Access Control (MAC) Bridges and Virtual Bridged Local Area Networks–Amendment 20: Shortest Path Bridging. *IEEE Std 802.1aq-2012 (Amendment to IEEE Std 802.1Q-2011 as amended by IEEE Std 802.1Qbe-2011, IEEE Std 802.1Qbc-2011, IEEE Std 802.1Qbb-2011, IEEE Std 802.1Qaz-2011, and IEEE Std 802.1Qbf-2011)* (2012), 1–340. https://doi.org/10.1109/IEEESTD.2012.6231597

[2] 2018. IEEE Standard for Local and Metropolitan Area Network–Bridges and Bridged Networks. *IEEE Std 802.1Q-2018 (Revision of IEEE Std 802.1Q-2014)* (July 2018), 1–1993. https://doi.org/10.1109/IEEESTD.2018.8403927

[3] 2018. IEEE Standard for Local and metropolitan area networks-Media Access Control (MAC) Security. *IEEE Std 802.1AE-2018 (Revision of IEEE Std 802.1AE-2006)* (Dec 2018), 1–239. https://doi.org/10.1109/IEEESTD.2018.8585421

[4] Alcatel. 2020. Alcatel SPB Intelligent Fabric. https://www.al-enterprise.com/-/media/assets/internet/documents/spb-based-transporation-networks-design-guide-en.pdf

[5] Avaya. 2012. Avaya - Shortest Path BridgingMAC (SPBM) Configuration. https://downloads.avaya.com/css/P8/documents/100128510

[6] H. Badis and K. A. Agha. 2003. An efficient mobility management in wireless overlay networks. In *14th IEEE Proceedings on Personal, Indoor and Mobile Radio Communications, 2003. PIMRC 2003.*, Vol. 3. 2500–2504 vol.3.

[7] Ayoub Bahnasse, Mohamed Talea, Abdelmajid Badri, Fatima Ezzahraa Louhab, and Sara Laafar. 2020. Smart hybrid SDN approach for MPLS VPN management on digital environment. *Telecommunication Systems* 73, 2 (2020), 155–169.

[8] Pankaj Berde, Matteo Gerola, Jonathan Hart, Yuta Higuchi, Masayoshi Kobayashi, Toshio Koide, Bob Lantz, Brian O'Connor, Pavlin Radoslavov, William Snow, and et al. 2014. ONOS: Towards an Open, Distributed SDN OS. In *Proceedings of the Third Workshop on Hot Topics in Software Defined Networking* (Chicago, Illinois, USA) *(HotSDN '14)*. Association for Computing Machinery, New York, NY, USA, 1–6. https://doi.org/10.1145/2620728.2620744

[9] A. Boukerche, R. W. N. Pazzi, and R. B. Araujo. 2005. HPEQ A Hierarchical Periodic, Event-driven and Query-based Wireless Sensor Network Protocol. In *The IEEE Conference on Local Computer Networks 30th Anniversary (LCN'05)l.* 560–567.

[10] Martin Casado, Michael J Freedman, Justin Pettit, Jianying Luo, Nick McKeown, and Scott Shenker. 2007. Ethane: Taking control of the enterprise. In *ACM SIGCOMM Computer Communication Review*, Vol. 37. ACM, 1–12.

[11] Martin Casado, Tal Garfinkel, Aditya Akella, Michael J Freedman, Dan Boneh, Nick McKeown, and Scott Shenker. 2006. SANE: A Protection Architecture for Enterprise Networks.. In *USENIX Security Symposium*, Vol. 49. 50.

[12] Martin Casado, Teemu Koponen, Scott Shenker, and Amin Tootoonchian. 2012. Fabric: A Retrospective on Evolving SDN. In *Proceedings of the First Workshop on Hot Topics in Software Defined Networks* (Helsinki, Finland) *(HotSDN '12)*. ACM, New York, NY, USA, 85–90. https://doi.org/10.1145/2342441.2342459

[13] Michael Dalton, David Schultz, Jacob Adriaens, Ahsan Arefin, Anshuman Gupta, Brian Fahs, Dima Rubinstein, Enrique Cauich Zermeno, Erik Rubow, James Alexander Docauer, et al. 2018. Andromeda: Performance, isolation, and velocity at scale in cloud network virtualization. In *15th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 18)*. 373–387.

[14] Bruce Davie, Teemu Koponen, Justin Pettit, Ben Pfaff, Martin Casado, Natasha Gude, Amar Padmanabhan, Tim Petty, Kenneth Duda, and Anupam Chanda. 2017. A Database Approach to SDN Control Plane Design. *SIGCOMM Comput. Commun. Rev.* 47, 1 (Jan. 2017), 15–26. https://doi.org/10.1145/3041027.3041030

[15] Dino Farinacci, Fabio Maino, Vince Fuller, and Albert Cabellos-Aparicio. 2020. *Locator/ID Separation Protocol (LISP) Control-Plane.* Internet-Draft draft-ietf-lisp-rfc6833bis-27. Internet Engineering Task Force. https://datatracker.ietf.org/doc/html/draft-ietf-lisp-rfc6833bis-27 Work in Progress.

[16] OpenStack Foundation. 2017. *Group-Based Policy for OpenStack Whitepaper.* https://wiki.openstack.org/w/images/a/aa/Group-BasedPolicyWhitePaper_v3.pdf

[17] Humberto Galiza, Marcos Schwarz, Jeronimo Bezerra, and Julio Ibarra. 2016. Moving an ip network to sdn: a global use case deployment experience at amlight. In *Anais do WPEIF 2016 Workshop de Pesquisa Experimental da Internet do Futuro*. 15.

[18] Natasha Gude, Teemu Koponen, Justin Pettit, Ben Pfaff, Martín Casado, Nick McKeown, and Scott Shenker. 2008. NOX: towards an operating system for networks. *ACM SIGCOMM Computer Communication Review* 38, 3 (2008), 105–110.

[19] Christian Hopps and Dave Thaler. 2000. Multipath Issues in Unicast and Multicast Next-Hop Selection. RFC 2991. https://doi.org/10.17487/RFC2991

[20] Luigi Iannone and Olivier Bonaventure. 2007. On the Cost of Caching Locator/ID Mappings. In *Proceedings of the 2007 ACM CoNEXT Conference* (New York, New York) *(CoNEXT '07)*. Association for Computing Machinery, New York, NY, USA, Article 7, 12 pages. https://doi.org/10.1145/1364654.1364663

[21] Sushant Jain, Alok Kumar, Subhasree Mandal, Joon Ong, Leon Poutievski, Arjun Singh, Subbaiah Venkata, Jim Wanderer, Junlan Zhou, Min Zhu, Jon Zolla, Urs Hölzle, Stephen Stuart, and Amin Vahdat. 2013. B4: Experience with a Globally-Deployed Software Defined Wan. *SIGCOMM Comput. Commun. Rev.* 43, 4 (Aug. 2013), 3–14. https://doi.org/10.1145/2534169.2486019

[22] T. Karagiannis and R. Mortier. 2008. Address and traffic dynamics in a large enterprise network. In *2008 16th IEEE Workshop on Local and Metropolitan Area Networks*. 102–107. https://doi.org/10.1109/LANMAN.2008.4675852

[23] Teemu Koponen, Keith Amidon, Peter Balland, Martin Casado, Anupam Chanda, Bryan Fulton, Igor Ganichev, Jesse Gross, Paul Ingram, Ethan Jackson, Andrew Lambeth, Romain Lenglet, Shih-Hao Li, Amar Padmanabhan, Justin Pettit, Ben Pfaff, Rajiv Ramanathan, Scott Shenker, Alan Shieh, Jeremy Stribling, Pankaj Thakkar, Dan Wendlandt, Alexander Yip, and Ronghua Zhang. 2014. Network Virtualization in Multi-tenant Datacenters. *Nsdi* (2014), 203–216. http://blogs.usenix.org/conference/nsdi14/technical-sessions/presentation/koponen

[24] Teemu Koponen, Martin Casado, Natasha Gude, Jeremy Stribling, Leon Poutievski, Min Zhu, Rajiv Ramanathan, Yuichiro Iwata, Hiroaki Inoue, Takayuki Hama, et al. 2010. Onix: A distributed control platform for large-scale production networks.. In *OSDI*, Vol. 10. 1–6.

[25] David Lebrun, Mathieu Jadin, François Clad, Clarence Filsfils, and Olivier Bonaventure. 2018. Software resolved networks: Rethinking enterprise networks with ipv6 segment routing. In *Proceedings of the Symposium on SDN Research*. ACM, ACM, 6.

[26] Dan Levin, Marco Canini, Stefan Schmid, Fabian Schaffert, and Anja Feldmann. 2014. Panopticon: Reaping the Benefits of Incremental SDN Deployment in Enterprise Networks. In *2014 USENIX Annual Technical Conference (USENIX ATC 14)*. 333–345.

[27] Mallik Mahalingam, Dinesh Dutt, Kenneth Duda, Puneet Agarwal, Larry Kreeger, T. Sridhar, Mike Bursell, and Chris Wright. 2014. Virtual eXtensible Local Area Network (VXLAN): A Framework for Overlaying Virtualized Layer 2 Networks over Layer 3 Networks. RFC 7348. https://doi.org/10.17487/RFC7348

[28] C. Mbarushimana and A. Shahrabi. 2007. Comparative Study of Reactive and Proactive Routing Protocols Performance in Mobile Ad Hoc Networks. In *21st International Conference on Advanced Information Networking and Applications Workshops (AINAW'07)*, Vol. 2. 679–684.

[29] Nick McKeown, Tom Anderson, Hari Balakrishnan, Guru Parulkar, Larry Peterson, Jennifer Rexford, Scott Shenker, and Jonathan Turner. 2008. OpenFlow: enabling innovation in campus networks. *ACM SIGCOMM Computer Communication Review* 38, 2 (2008), 69–74.

[30] Donald R. Morrison. 1968. PATRICIA-Practical Algorithm To Retrieve Information Coded in Alphanumeric. *J. ACM* 15, 4 (Oct. 1968), 514–534. https://doi.org/10.1145/321479.321481

[31] Yukihiro Nakagawa, Kazuki Hyoudou, and Takeshi Shimizu. 2012. A Management Method of IP Multicast in Overlay Networks Using Openflow. In *Proceedings of the First Workshop on Hot Topics in Software Defined Networks* (Helsinki, Finland) *(HotSDN '12)*. Association for Computing Machinery, New York, NY, USA, 91–96. https://doi.org/10.1145/2342441.2342460

[32] Ruoming Pang, Mark Allman, Mike Bennett, Jason Lee, Vern Paxson, and Brian Tierney. 2005. A First Look at Modern Enterprise Traffic. In *Proceedings of the 5th ACM SIGCOMM Conference on Internet Measurement* (Berkeley, CA) *(IMC '05)*. USENIX Association, USA, 2.

[33] Charles E. Perkins. 2002. IP Mobility Support for IPv4. RFC 3344. https://doi.org/10.17487/RFC3344

[34] Yakov Rekhter and Eric C. Rosen. 2006. BGP/MPLS IP Virtual Private Networks (VPNs). RFC 4364. https://doi.org/10.17487/RFC4364

[35] Alberto Rodriguez-Natal, Vina Ermagan, Johnson Leong, Fabio Maino, Albert Cabellos-Aparicio, Sharon Barkai, Dino Farinacci, Mohamed Boucadair, Christian Jacquenet, and Stefano Secci. 2020. *Publish/Subscribe Functionality for LISP.* Internet-Draft draft-ietf-lisp-pubsub-05. Internet Engineering Task Force. https://datatracker.ietf.org/doc/html/draft-ietf-lisp-pubsub-05 Work in Progress.

[36] Alberto Rodriguez-Natal, Marc Portoles-Comeras, Vina Ermagan, Darrel Lewis, Dino Farinacci, Fabio Maino, and Albert Cabellos-Aparicio. 2015. LISP: a southbound SDN protocol? *IEEE Communications Magazine* 53, 7 (2015), 201–207.

[37] Allan Rubens, Carl Rigney, Steve Willens, and William A. Simpson. 2000. Remote Authentication Dial In User Service (RADIUS). RFC 2865. https://doi.org/10.17487/RFC2865

[38] S. Sezer, S. Scott-Hayward, P. K. Chouhan, B. Fraser, D. Lake, J. Finnegan, N. Viljoen, M. Miller, and N. Rao. 2013. Are we ready for SDN? Implementation challenges for software-defined networks. *IEEE Communications Magazine* 51, 7 (2013), 36–43. https://doi.org/10.1109/MCOM.2013.6553676

[39] Michael Smith, Rakesh Reddy Kandula, and Syam Appala. 2020. *Scalable-Group Tag eXchange Protocol (SXP).* Internet-Draft draft-smith-kandula-sxp-10. Internet Engineering Task Force. https://datatracker.ietf.org/doc/html/draft-smith-kandula-sxp-10 Work in Progress.

[40] Michael Smith and Larry Kreeger. 2018. *VXLAN Group Policy Option.* Internet-Draft draft-smith-vxlan-group-policy-05. Internet Engineering Task Force. https://datatracker.ietf.org/doc/html/draft-smith-vxlan-group-policy-05 Work in Progress.

[41] Gary N Stone, Bert Lundy, and Geoffrey G Xie. 2001. Network policy languages: a survey and a new approach. *IEEE network* 15, 1 (2001), 10–21.

[42] Yu-Wei Eric Sung, Xin Sun, Sanjay G Rao, Geoffrey G Xie, and David A Maltz. 2011. Towards systematic design of enterprise networks. *IEEE/ACM Transactions on Networking (TON)* 19, 3 (2011), 695–708.

[43] Z. Teo, K. Birman, and R. Van Renesse. 2016. Experience with 3 SDN Controllers in an Enterprise Setting. In *2016 46th Annual IEEE/IFIP International Conference on Dependable Systems and Networks Workshop (DSN-W)*. 97–104. https://doi.org/10.1109/DSN-W.2016.20

[44] Bingchuan Tian, Xinyi Zhang, Ennan Zhai, Hongqiang Harry Liu, Qiaobo Ye, Chunsheng Wang, Xin Wu, Zhiming Ji, Yihong Sang, Ming Zhang, and et al. 2019. Safely and Automatically Updating In-Network ACL Configurations with Intent Language. In *Proceedings of the ACM Special Interest Group on Data Communication* (Beijing, China) *(SIGCOMM 2019)*. Association for Computing Machinery, New York, NY, USA, 214–226. https://doi.org/10.1145/3341302.3342088

[45] Rory Ward and Betsy Beyer. 2014. BeyondCorp: A New Approach to Enterprise Security. *;login:* Vol. 39, No. 6 (2014), 6–11.