Mosaic: Policy Homomorphic Network Extension

Li Erran Li[⋆] Michael F. Nowlan[†] Yang Richard Yang[†] Ming Zhang[‡]

Bell Labs[⋆] Microsoft Research[‡] Yale University[†]
erranlli@research.bell-labs.com {michael.nowlan,yang.r.yang}@yale.edu
mzh@microsoft.com

ABSTRACT

With the advent of large-scale cloud computing infrastructures, network extension has emerged as a major challenge in the management of modern enterprise networks. Many enterprises are considering extending or relocating their network components, in whole or in part, to remote, private and public data centers, in order to attain scalability, failure resilience, and cost savings for their network applications. In this paper, we conduct a first study on the extension of an enterprise network while preserving its performance and security requirements, such as layer 2/layer 3 reachability, and middle-box traversal through load balancer, intrusion detection and ACLs. We formulate this increasingly important problem, present preliminary designs, and conduct experiments to validate the feasibility of our design.

Categories and Subject Descriptors

C.2.1 [**Computer Communication Networks**]: Network Architecture and Design—*Network communications*; C.2.3 [**Computer Communication Networks**]: Network Operations

General Terms

Algorithms, Performance, Reliability.

Keywords

Network Extension, Network Migration, Network Policy.

1. INTRODUCTION

Nowadays, there is an increasing demand on network extension. By network extension, we mean both scaling-out and migration of components of an existing enterprise network to provide infrastructure as a service (IaaS). Many factors drive the demand for network extension, including enterprise dynamics (*e.g.*, expansion into a new site), hardware consolidation, and the emergence of cloud computing infrastructures. As a recent survey [9] has shown, many enterprises are running out of space in their existing data centers, and thus need to extend or relocate their networks to new data centers. Recent emergence of public cloud computing infrastructures (*e.g.*,) provide enormous opportunities for an enterprise to either replace or complement its existing infrastructure with computing resources in the cloud (*e.g.*, [6]), in order to take advantage of improved efficiency and reliability. We refer to the private or public data centers that an enterprise extends to as the *remote data centers*; the original enterprise network as the *base network*.

Despite the potential business benefits and needs, network extension can be technically quite complex and thus pose substantial challenges to the management of modern enterprise networks. In particular, such extensions often have to be incremental instead of a complete restructuring of the existing network infrastructure. Thus, a seemingly small extension can be quite challenging to handle in practice.

Consider a simple example of relocating a set of application servers from one data center of an enterprise to a remote data center (*e.g.*, another private or public cloud data center). This seemingly simple task can be quite challenging due to multiple reasons. First, these servers usually have complex communication patterns governed by network policies (*e.g.*, [7]), such as traversal of firewalls and intrusion detection systems before being reached. Second, the enterprise network may enforce network policies using a variety of techniques including routing design, topology design, and deployment of policy boxes at strategic locations. Some of such techniques, such as deployment at topology cuts, can be implicit without any explicit representation. Third, the remote data center may present a different design with different topology and different placement of middleboxes. Specifically, there are two common ways to extend an enterprise network to a remote data center. In one extreme, a remote data center may belong to the same enterprise, allowing plenty of flexibility in constructing network topology and placing policy boxes inside the remote data center. In the other extreme, a remote data center may belong to a public cloud provider, imposing substantial restrictions on the network layout and customized policy enforcement. Given the preceding challenges, it can be extremely difficult to take these servers out of their current "context" and place them into another "context" while preserving existing network policies. Manual reconfiguration, although maybe feasible for small networks, cannot satisfy the need to scale to large enterprise networks.

We present Mosaic, a first framework for network extension while preserving enterprise network policies. We refer to policy-preserving network extension as *policy homomorphic network extension*. Mosaic introduces two key notions — way-points and scopes — to capture network policy constraints during network extension. Moreover, Mosaic includes simple and yet powerful primitives (*e.g.*, proxy and mirror) to implement network extension. Guided by the policy constraints and utilizing the primitives, a Mosaic extension algorithm computes an efficient network extension strategy.

The rest of the paper is organized as follows. In Section 2, we present a motivating example. In Section 3, we give an overview

LADIS '10 Zürich, Switzerland

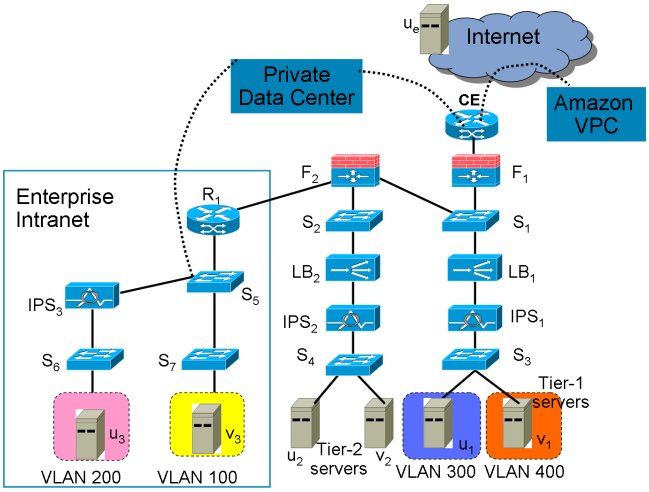

Figure 1: The configuration of a real enterprise network hosting multi-tiered applications. The network implements many policies through middlebox placement, topology design and routing design.

on the Mosaic framework. Section 4 presents how Mosaic specifies the requirements and constraints of network extension, and Section 5 describes the network transformation algorithms for network extension. We then evaluate our network extension algorithms in a large campus network setting. Our preliminary results indicate that the Mosaic extension algorithms perform better than a naive server relocation algorithm in terms of number of policy violations. We conclude and give future work in Section 7.

2. MOTIVATING EXAMPLE

We start with a motivating example for the extension of an enterprise network into either a public data center such as Amazon's EC2 or a private data center.

Figure 1 shows part of a real network without equipment model names or numbers. The network is a relatively standard three-tier design, hosting multiple applications of the organization. For reliability, each logical network device (*e.g.*, firewall F_i, load balancer LB_i, intrusion prevention system IPS_i, and switch S_j, $i = 1,2$, $j = 1,\cdots,5$) represents two identical physical devices. One is active, and the other standby. To make the figure easy to read, we draw only one such device. Note that LB_1, LB_2 and CE are layer 3 (L3) devices; servers are endpoints; and the rest are layer 2 (L2) devices.

Specifically, the tier-1 servers are the front ends of multiple network applications. The tier-1 servers of a given application are configured to belong to an IP subnet with private IP addresses. Each application is also assigned a public IP address to allow external access. Public IP addresses are assigned to the two load balancers represented by LB_1. The tier-1 servers communicate with the tier-2 servers, which are located behind the two load balancers represented by LB_2. The tier-2 servers and LB_2 are configured with private IP addresses for security protection. The LB_1 balancers are configured with static routes to reach LB_2. The network border gateway CE has no knowledge about the routes to the tier-2 servers.

Extension to Public Cloud

As an example, let us first consider the possibility of relocating the tier-1 servers to a public cloud such as Amazon's EC2. One might consider this a trivial task. Specifically, after relocating the tier-1 servers to EC2, the operator simply updates LB_1 with the new IP addresses of the servers, if their IP addresses change. However, this simple solution can be broken in multiple aspects:

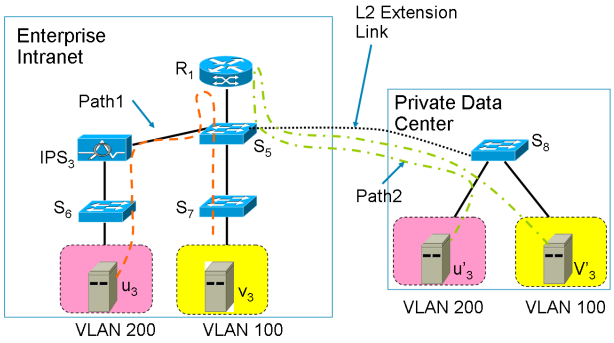

Figure 2: Layer 2 extension may introduce policy violations.

- *Violation of security policies:* The tier-1 servers are configured with multiple subnets, and the two boxes represented by IPS_1 monitor cross-subnet traffic. By simply relocating the tier-1 servers without relocating IPS_1, the solution bypasses the protection provided by IPS_1, violating the security policies of the organization.
- *Broken external client sessions:* Consider that an external Internet client establishes a connection with a public IP address of LB_1. The load balancer directs the request from the client to one of the relocated tier-1 servers. The tier-1 server in the cloud processes the request and sends back a reply, with the client's address as the destination and the server's address as the source. However, the client is expecting a reply with a source IP address of the load balancer, not the server. The client machine is likely to drop the reply, thus breaking the client's session.
- *Disconnection from Tier-2 servers:* Recall that only the two load balancers represented by LB_1 have routes to the tier-2 servers. Thus, when packets sent by the relocated tier-1 servers to tier-2 reach CE (the customer gateway), say via an Amazon VPC tunnel, CE will drop these packets because it does not know how to forward them. In this case, we see internal connection failure, as opposed to the preceding issue of broken external client sessions.

Extension to Private Cloud with L2 Extension

In light of the preceding issues, one might think that L2 extensions [3] into a private data center may not generate policy violations. Specifically, L2 extensions are the techniques to enable a LAN to be extended to a remote site, in order to reduce network and application changes needed to support live server migration. Previous work has focused on transparency in terms of L2 connectivity [3, 10].

However, L2 extension still does not address policy homomorphism (defined in §1). In the preceding example, consider the case of extending both VLAN 100 and VLAN 200 into a remote data center in order to scale-out the intranet servers in the enterprise. Figure 2 zooms in on the left portion of Figure 1.

Specifically, assume that an L2 extension link is created between S_5 and S_8 and VLAN 100 and VLAN 200 are logically connected to S_8 in the remote data center. When a server v_3 in VLAN 100 communicates with u_3 in VLAN 200 in the enterprise network, the packet traverses: $v_3 \rightarrow S_7 \rightarrow S_3 \rightarrow S_5 \rightarrow R_1 \rightarrow S_5 \rightarrow IPS_3 \rightarrow S_6 \rightarrow u_3$. However, when v_3' in VLAN 100 communicates with u_3' in VLAN 200 in the private data center, it will not go through IPS_3; similarly the path from v_3 to u_3' will not traverse IPS_3. Thus, L2 extension will not satisfy policy constraints automatically.

3. MOSAIC OVERVIEW

The motivating example reveals potential issues facing the extension of an enterprise network (*i.e.*, the base network) into a remote

data center. Mosaic is a framework to address these issues. Mosaic consists of two major components: policy specification and network transformation.

Policy specification: To systematically investigate and solve the problems raised in the preceding section, we need to explicitly define the policies that an enterprise network intends to enforce so that one can validate any given solution. Policies capture the "invariants" that network extension should preserve. Since network extension alters an existing network topology (*e.g.*, by adding new nodes or relocating existing nodes), the *traversal* and *scope* of a packet (or frame if we talk about layer 2) can deviate from those in the base network. Thus, policy specification is crucial for policy enforcement, which will be discussed in Section 4.

Network transformation: Bounded by policy specification, network transformation computes the configuration at the remote data centers as well as at the base network. In addition to policies, multiple other factors, including objectives and constraints on application performance and migration costs, contribute to the complexity and effectiveness of network transformation. For example, if the objective is to achieve rapid network application deployment (*e.g.*, dynamic addition/removal of servers), then the solution is likely to be software based, without involving hardware rearrangement.

In particular, the capabilities of network devices influence what transformation techniques may be used. In this paper, we do not assume the availability of mechanisms such as pswitches [7] and OpenFlow [8]. While these mechanisms can simplify our solutions, they have not been widely adopted. Instead, we focus on the traditional mechanisms that are readily available in today's enterprise networks. In Section 5, we will discuss the primitives and algorithmic framework of Mosaic.

4. POLICY SPECIFICATION

We start with the policy specification. The capability to specify policies for enterprise networks can be a highly valuable tool (*e.g.*, [2]). In Mosaic, we represent the topology of the original enterprise network G using V, the set of nodes consisting of end hosts (servers, virtual machines), switches, routers and middleboxes; and E, the set of connections among network nodes.

An enterprise network operator defines policies P on packets and frames, based on topology, as we have seen in the motivating example. Since we treat L3 packets and L2 frames uniformly in our framework, we use packet as a general term. For a given packet, policies specify additional information beyond what is already contained in the packet. Specifically, for a given packet pkt_i, policy $\texttt{Policy}_\texttt{i}$ consists of not only destination(s) $\texttt{Destination}_\texttt{i}$ but also two additional perspectives: waypoints $\texttt{Waypoints}_\texttt{i}$ and scope $\texttt{Scope}_\texttt{i}$.

By default, packets not associated with any policy are unwanted. These packets must be filtered before reaching their destinations. This default policy captures un-reachability policies which are typically enforced by limiting route redistributions and specifying access control lists (ACLs) in routers.

Waypoints: The waypoints of a packet are the network nodes in addition to the destination(s) that should receive the packet. An enterprise may design its network such that a packet should pass through a particular set of network nodes. In the motivating example, we see that packets from the Internet should visit an intrusion prevention box before reaching a tier-1 server. As another example, an enterprise network may deploy a sniffer that is connected to the mirror port of a switch to receive a copy of a given packet for logging purpose. In this case, the sniffer also belongs to the waypoints of the packet. Let $\texttt{Waypoints}_\texttt{i}$ be the waypoints of packet pkt_i.

Waypoints are specified by using the *ordering* and *occurrence* constraints. Ordering specifies if there are any constraints on the order to visit the waypoints. For example, an enterprise network may require a packet to visit one middlebox before visiting another one. Occurrence specifies the number of times that a middlebox should be visited. For example, a packet may visit a middlebox only once, or none at all. When we want to emphasize that $\texttt{Waypoints}_\texttt{i}$ requires the ordering and occurrence constraints that are not included in other policy specification systems (*e.g.*, [2]), we write $\texttt{Waypoints}_\texttt{i}(\texttt{Order}_\texttt{i}, \texttt{Occurrence}_\texttt{i})$.

It is important to realize that we use network nodes in a generic sense when specifying waypoints. We can view each network node, in particular, a middlebox, as the member of a function class (*e.g.*, firewall, intrusion prevention, or sniffer) with a specific configuration state. Formally, we denote the function class of the middlebox node v_j as $\texttt{class}(\texttt{v}_\texttt{j})$; and its configuration state as $\texttt{conf}(\texttt{v}_\texttt{j})$.

As an example, consider the network in Figure 1. The tier-1 and tier-2 firewalls have the same function class: $\texttt{class}(\texttt{F}_1) = \texttt{class}(\texttt{F}_2) = \texttt{Firewall}$. But their configuration states are different: the tier-1 firewall is in charge of the first line of defense and thus is configured to allow only HTTP traffic; the tier-2 firewall handles traffic from the tier-1 servers and intranet and thus may allow more protocols.

Scope: Destinations and waypoints capture the nodes that a packet *must* visit. However, a packet *may* reach other nodes in an enterprise network. For example, a modern switch may flood a given packet to a layer 2 domain if a forwarding entry is not present in its layer 2 FIB (forwarding information base); routers and switches along the path from the source to the destinations will see the packet (if unencrypted); due to routing changes, some routers not on the normal forwarding path may also see the packet. We introduce a concept called *scope* with each packet, which defines the security zone of the packet. The scope is the maximum set of nodes that a packet can reach. Let $\texttt{Scope}_\texttt{i}$ be the scope of pkt_i.

Example Policies: We now illustrate the preceding concepts using the example shown in Figure 1. Figure 3 specifies six policies for the network. In each of the six policies, σ represents the set of valid policy preserving paths.

Policy $\texttt{Policy}_1$ specifies that any HTTP request packet pkt_1 to a tier-1 application server from an Internet client must traverse tier-1 firewall, tier-1 load balancer (the tier-1 application's public IP address is configured at the load balancer LB_1). The packet's destination is changed to a tier-1 server u_1 by LB_1. We treat this as a new packet pkt_2. This packet with source u_e and destination u_1 which originates from L_1 needs to traverse IPS_1. The scope of pkt_1 $\texttt{Scope}_1 = \{\texttt{LB}_1, F_1, CE, S_1, u_e\}$. The scope of pkt_2 $\texttt{Scope}_2 = \{\texttt{LB}_1, \texttt{IPS}_1, \texttt{S}_3, \texttt{u}_1\}$.

Policy $\texttt{Policy}_3$ says that, any reply packet pkt_3 from a tier-1 server to an Internet client must be sent to the load balancer first. It should be checked by IPS_1.

Policy $\texttt{Policy}_4$ says that, for any packet pkt_4 with source LB_1 originating from u_1, destined to an Internet client needs no further checks. $\texttt{Scope}_3 = \texttt{Scope}_2$ and $\texttt{Scope}_4 = \texttt{Scope}_1$.

Policy $\texttt{Policy}_5$ states that a tier-1 server's packet pkt_5 must traverse tier-2 firewall and load balancer LB_2. The scope $\texttt{Scope}_5 = \{\texttt{u}_1, \texttt{u}_2, \texttt{F}_2, \texttt{LB}_2, \texttt{IPS}_2, S_1, S_2, S_3, S_4, IPS_1, LB_1\}$.

Policy $\texttt{Policy}_6$ states that cross-traffic between tier-1 servers in different subnet must be checked by IPS_1. The scope $\texttt{Scope}_6 = \{\texttt{u}_1, \texttt{v}_1, \texttt{IPS}_1, \texttt{S}_3\}$.

5. NETWORK TRANSFORMATION

5.1 Overview

We consider network transformation algorithms that take as input the base network, its policy specification, the set U of servers

// 1. Internet client u_e to a tier-1 application
$\texttt{Policy}_1 = ([\texttt{u}_e, \texttt{L}_1, *, 80, \texttt{TCP}], \texttt{Scope}_1, \texttt{Waypoints}_1(\{\texttt{F}_1\texttt{LB}_1\}, \{\sigma | \texttt{Ocurr}(\sigma, \texttt{F}_1) = 1, \texttt{Ocurr}(\sigma, \texttt{LB}_1) = 1\})$
$\texttt{Policy}_2 = ([\texttt{u}_e, \texttt{u}_1, *, 80, \texttt{TCP}], \texttt{Scope}_2, \texttt{Waypoints}_2(\{\texttt{IPS}_1\}, \{\sigma | \texttt{Ocurr}(\sigma, \texttt{IPS}_1) > 0\})$

// 2. Tier-1 application server u_1's reply to Internet client u_e
$\texttt{Policy}_3 = ([\texttt{u}_1, \texttt{u}_e, 80, *, \texttt{TCP}], \texttt{Scope}_3, \{\sigma | \texttt{Ocurr}(\sigma, \texttt{LB}_1) = 1, \texttt{Ocurr}(\sigma, \texttt{IPS}_1) > 0\})$
$\texttt{Policy}_4 = ([\texttt{u}_1, \texttt{u}_e, 80, *, \texttt{TCP}], \texttt{Scope}_4, \texttt{Waypoints}_2(\{\}, \{\}))$

// 3. Tier-1 application server u_1 communicates with tier-2 server u_2
$\texttt{Policy}_5 = ([\texttt{u}_1, \texttt{u}_2, *, *, \texttt{TCP}], \texttt{Scope}_5, \{\texttt{F}_2\texttt{LB}_2\texttt{IPS}_2\}, \{\sigma | \texttt{Ocurr}(\sigma, \texttt{F}_2) = 1, \texttt{Ocurr}(\sigma, \texttt{LB}_2) = 1, \texttt{Ocurr}(\sigma, \texttt{IPS}_2) > 0\})$

// 4. Tier-1 application server u_1 in subnet 1 communicates with tier-1
// application server v_1 in subnet 2
$\texttt{Policy}_6 = ([\texttt{u}_1, \texttt{v}_1, *, *, \texttt{TCP}], \texttt{Scope}_6, \texttt{Waypoints}_6(\{\texttt{IPS}_2\}, \{\sigma | \texttt{Ocurr}(\sigma, \texttt{IPS}_2) > 0\})$

Figure 3: Policies for enterprise network in Figure 1.

to be extended or relocated to a remote data center, reconfiguration constraints at the base network and the remote data center, the cost model of network equipment and traffic, and performance constraints on applications. The set U can be manually given or computed by another algorithm.

The outputs of network transformation include:

- The connectivity from the base network to the remote data center;
- The reconfiguration of the base network, including addition and deletion of nodes, as well as reconfiguration of existing nodes;
- The configuration of the remote data center.

Note that the capabilities supported at the remote data center can place substantial constraints on the outputs of the network transformation algorithm. Consider Amazon's VPC as an example remote data center. VPC makes public cloud resources appear the same as internal enterprise resources. However, VPC imposes specific constraints on the connections from the enterprise network. First, VPC specifies L3 connectivity. Second, inside VPC, the enterprise can construct only a logical star topology connecting multiple subnets. On the other hand, a private data center, for instance, a new data center owned by the same enterprise, may allow more flexibility. In this case, the remote data center may allow both L2 and L3 connectivity from the base enterprise network to the remote data center. Also, the enterprise can have flexibility in constructing a topology and placing policy devices inside the remote data center.

To be concrete, we present a two-stage transformation algorithm.

Stage 1: The algorithm computes, for each policy, whether to enforce it at the base network or the remote data center. The computation is based on the constraints on application performance (*e.g.*, delay constraints), enterprise costs (*e.g.*, cross data-center traffic and equipment replication cost), and the availability of policy classes at the base network and the remote data center.

Stage 2: The algorithm constructs detailed configurations at the base network and the remote data center.

Instead of going over all steps of complete algorithms, we present three key primitives used at Stage 2:

- Mosaic proxy: this primitive allows enforcement of a policy at the base network. The primitive is driven by the principle of least-disruption and greatest re-use. It enforces a policy by traversing the original middleboxes in the base network. This can be a quite useful primitive when there are constraints on deploying middleboxes in the remote data center, due to constraints on cost (*e.g.*, cannot duplicate a middlebox) or availability (*e.g.*, limited placement feasibility inside a public cloud data center). The primitive tries to avoid policy omissions such as those discussed at the end of Section 2.
- Mosaic mirror: this primitive enforces policies at the remote data center by replicating a minimal set of middleboxes in the remote data center. The replicated set is determined by computing an edge-cut-set surrounding relocated nodes to ensure robust policy enforcement, even in the presence of failures. Enforcing policies at the remote data center reduces latency, in particular, for traffic among relocated servers.
- Mosaic policy relocation: this primitive optimizes specific classes of policies (*e.g.*, firewall) by relocating them from one device to another existing network device (*e.g.*, as a different firewall context) to enforce policy without introducing any new devices.

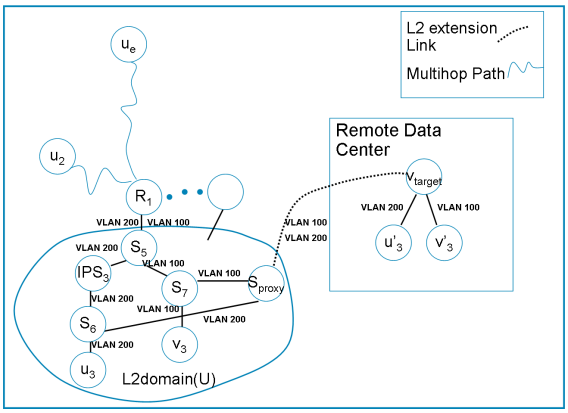

Figure 4: Mosaic proxy resolves policy violation of Fig 2.

Next we present more details on the proxy and mirror primitives.

5.2 Mosaic Proxy

The objective of Mosaic proxy is to force packets to go back to the base network to satisfy the policies. A key challenge, as we discussed, is to not introduce unintended paths that may introduce policy violations.

We illustrate the basic idea of Mosaic proxy using an example. In particular, we show how Mosaic proxy addresses the issue shown in Figure 2.

Specifically, by using Mosaic proxy, we have the context (determined by a higher-level algorithm) that the network operator targets to still use IPS_3 in the base network to conduct policy check for communications between VLAN 100 and VLAN 200. In other words, the communications between u'_3 and v'_3 should still go through IPS_3. Thus, packets from u'_3 in VLAN 200 at the remote data center should be routed back to the base network first (and visit IPS_3), before reaching v'_3 in VLAN 100 at the remote data center.

The issue of the solution shown in Section 2, however, is that although the traditional layer-2 extension sends the packet from u'_3 to v'_3 back to the base network, it creates a new path connecting the two VLANS, bypassing IPS_3 and thus causing a policy violation.

Figure 4 illustrates the introduction of a Mosaic proxy to fix the issue. Let v_{target} denote the entrance to the remote data center. Mosaic proxy introduces a switch S_{proxy} with L2 connectivity to v_{target}. Since u_3 and v_3 will migrate to the remote data center, Mosaic proxy connects their corresponding switches S_6 and S_7 to S_{proxy} with VLAN configurations shown in the figure.

Now, consider the policy that communications between u'_3 and v'_3 be checked by IPS_3 in the network after migration. Specifically, since v'_3 and u'_3 are in different subnets, a packet from one to the

other will be routed to R_1 in VLAN 100. In particular, the path from v'_3 to R_1 is $v'_3 \rightarrow v_{target} \rightarrow S_{proxy} \rightarrow S_7 \rightarrow S_5 \rightarrow R_1$. The path from R_1 to u'_3 in VLAN 200 is $S_5 \rightarrow IPS_3 \rightarrow S_6 \rightarrow S_{proxy} \rightarrow v_{target} \rightarrow u'_3$. Thus, any packet from v'_3 to u'_3 traverses the policy box IPS_3, satisfying the policy requirement.

We can prove that Mosaic proxy can satisfy basic network policies for an extended server set U with the `Scope` extended to include relevant nodes at the remote data center.

Note that S_{proxy} does not have to be a new device. It can be any L2 switch that can connect to V_{target}. The logical links connecting S_6, S_7 to S_{proxy} can be implemented using existing mechanisms such as 802.1Q (double VLAN tagging) and MACinMAC (MAC encapsulation). With 802.1Q, the straightforward solution is to construct an outer VLAN for each logical link. Frames are tagged with the outer VLAN ID and carried in the outer VLAN at one end of the logical link. The outer VLAN tag is removed at the other end of the logical link. With MACinMAC, frames are encapsulated at one end of the logical link and decapsulated at the other end of the logical link. These solutions can be further optimized to reduce the number of VLANs or MACinMAC tunnels needed.

5.3 Mosaic Mirror

One basic function of the Mosaic mirror primitive is to achieve network extension by enforcing policies at the remote data center (locally) for communications among relocated (or newly added) end points at the remote data center, without the need of going back to the base network. The mirror primitive complements the Mosaic proxy primitive and is necessary in settings with constraints such as low-latency communications among relocated end points.

The mirror primitive consists of two algorithmic components: (1) it computes the locations and reconfiguration at the base network to enforce policies between those relocated (newly added) and those not; and (2) it computes a minimal network configuration at the remote data center that can enforce policies involving those relocated end points.

Specifically, consider the case that the policies for the communications within a set of relocated end points U are enforced by L2 devices. Then the mirror primitive computes the remote data center configuration in the following way. First, it computes `L2Domain(U)`, by starting at the set U, and recursively adding necessary layer 2 nodes. The primitive stops when all outgoing edges from `L2domain(U)` are connected to nodes with layer 3 labels. Denote $L3Cutset(U)$ as the set of layer 3 nodes that connect to at least one node in `L2domain(U)` as the cutset routers. The primitive uses `L2domain(U)` as a basis to construct a topology at the remote data center, and reconfigures routers in $L3Cutset(U)$ to create tunnels and routing to robustly enforce policies.

Figure 5 shows an example of how Mosaic mirror enforces the policy of the example shown in Figure 2. In this example, the primitive computes a minimal topology at the remote data center to enforce policies among the communications of the relocated u_3 and v_3. We can see that the remote topology removes S'_7. The primitive also computes a network cut, establishes tunnels to enforce policies between those relocated and those not (*e.g.*, u_e).

One can prove that Mosaic mirror can satisfy basic network policies for an extended server set U with `Scope` extended to include relevant nodes at the remote data center.

6. EVALUATION

We conduct preliminary evaluation on the effectiveness of Mosaic.

6.1 Methodology

Specifically, we obtain router, middlebox and switch configuration files of a campus network with more than 50 routers and more than 1000 switches. We extract route distribution graph, and L3 topology using a tool in [1]. We then insert the L2 topology into L3 topology due to the fact that switch configurations are not adequate. We infer the middlebox traversal policy based on the topology properties and route distribution graph. We examine the possible paths between two endpoints (represented as two subnets or two VLANs). From the path, we determine the middleboxes traversed and store this sequence as the way-points for this particular path. For scope, if both endpoints are in the same VLAN, then the scope is all nodes in the broadcast domain. If they are not in the same VLAN, we use all reachable nodes (based on route distribution graph and ACLs) in the security zone as the scope.

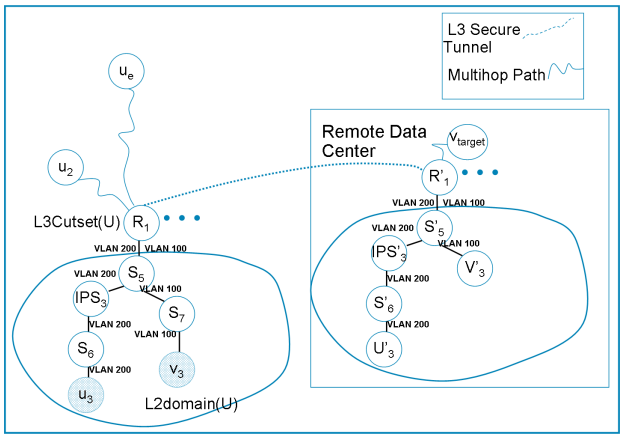

Figure 5: An example illustrating the Mosaic primitive.

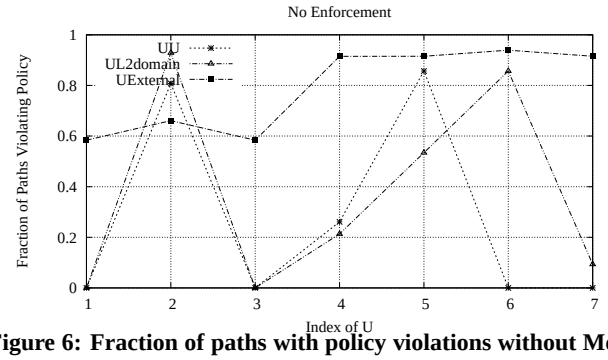

Figure 6: Fraction of paths with policy violations without Mosaic.

6.2 Results

Figure 6 shows the results of no policy enforcement extension, when we pick a set U of servers to relocate. The x-axis is the index of the collection of Us (7 of them) to relocate. Specifically, the algorithm operates in the following way: Routes in the remote data center are advertised through BGP and are redistributed to the enterprise network. As such, the design is the most efficient solution in terms of performance. However, it is not a viable option to enforce policy requirements due to its volatile tendency to violate policy. We break the communication into three types: `UU` sessions (among relocated servers), `UL2domain` (between relocated and those remaining at the same L2 domain), and `UExternal`(between relocated and those outside the campus network). We see that there can be significant policy violations. The `UExternal` policy violations can occur on as many as 80% of the paths due to traversal of several security zones. Conversely, Mosaic sustains no policy violations.

Although the no-policy-enforcement approach is not feasible in

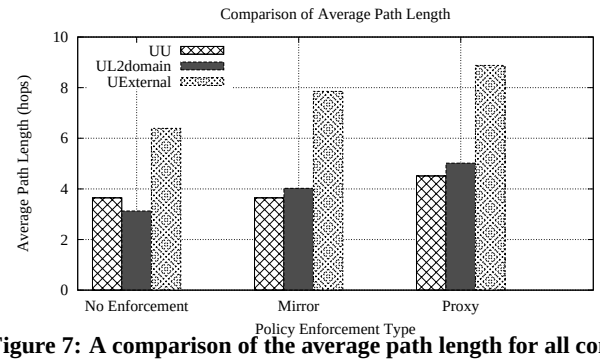

Figure 7: A comparison of the average path length for all communication sessions.

practice, it is a baseline comparison for measuring the cost of enforcing policy. In the next experiment, we evaluate the cost of enforcing policies by considering the average path length for communication between points in relocated U and other endpoints in the network. The path length is defined as the number of network devices a packet must traverse from source to destination (*i.e.*, network hops).

Figure 7 shows the results. The distances shown in these results only include the enterprise portion and count both L2 and L3 hops. A tunnel hop is counted by the number of hops it traverses. It is important to recall that Mosaic-Mirror enforces policies remotely, whereas Mosaic-Proxy enforces locally. We observe that the price of enforcing policies measured by hop counts may not be significant, shown by the non-significant increase of hop counts compared with no policy enforcement.

7. CONCLUSION AND FUTURE WORK

Network extension and migration are now a major challenge for large-scale networks, attracting industrial attention (*e.g.*, [3, 4, 5, 10]). Ad-hoc methods of network extension and migration can result in serious policy violations. In this paper, we present a framework for network policy specification. Furthermore, we evaluate the feasibility of policy homomorphic network extension and migration to remote data centers.

There are many avenues for future work. In particular, we plan to conduct large-scale tests to evaluate the scalability of our algorithms.

8. ACKNOWLEDGMENTS

The research of Y.R. Yang is supported in part by NSF. We are grateful to the contributions of Xuan Zhang, Andreas Voellmy, Dave Maltz, and Chen Tian. We thank Thomas Woo for helpful discussions. The comments of the anonymous reviewers helped to improve the paper.

9. REFERENCES

[1] T. Benson, A. Akella, and D. A. Maltz. Mining policies from enterprise network configuration. In *Internet Measurement Conference*, Chicago, Illinois, Nov 2009.

[2] M. Casado, M. Freedman, J. Pettit, N. McKeown, and S. Shenker. Ethane: Taking control of the enterprise. In *Proc. of SIGCOMM*, Kyoto, Japan, Aug. 2010.

[3] Cisco White Paper. Data center interconnect: Layer 2 extension between remote data centers. Available at `http://www.cisco.com/en/US/prod/collateral/switches/ps5718/ps708/white_%paper_c11_493718.html`, 2010.

[4] Cohesive Flexible Technologies Corp. VPN-Cubed: customer controlled security for the Cloud. Available at `http://www.cohesiveft.com/Cube/VPN/VPN-Cubed_Custom_Enterprise_Configur%ations/`, 2010.

[5] Computer World. IBM, Juniper join in Cloud strategy: Technology would reallocate computing resources between private, public Cloud. available at `http://www.computerworld.com/s/article/9127702/IBM_Juniper_join_in_clou%d_strategy`, Feb 2009.

[6] M. Hajjat, X. Sun, Y.-W. E. Sung, D. Maltz, S. Rao, K. Sripanidkulchai, and M. Tawarmalani. Cloudward bound: Planning for beneficial migration of enterprise applications to the cloud. In *Proc. of SIGCOMM*, Delhi, India, Aug. 2010.

[7] D. A. Joseph, A. Tavakoli, and I. Stoica. A policy-aware switching layer for data centers. In *ACM SIGCOMM*, Seattle, WA, Aug 2008.

[8] Openflow. The OpenFlow switch specification. `http://OpenFlowSwitch.org`.

[9] Viridity Software. E-Guide: An expert guide to data center trends, energy regulations, and more. Available at `http://www.viridity.com/`, Apr. 2010.

[10] T. Wood, A. Gerber, K. Ramakrishnan, and J. Van Der Merwe. The case for enterprise-ready virtual private clouds. In *USENIX HotCloud*, San Diego, CA, Jun 2009.