

The Evolution of Network Configuration: A Tale of Two Campuses

Hyojoon Kim
Georgia Tech
Atlanta, GA, USA
joonk@gatech.edu

Theophilus Benson
University of Wisconsin
Madison, WI, USA
tbenson@cs.wisc.edu

Aditya Akella
University of Wisconsin
Madison, WI, USA
akella@cs.wisc.edu

Nick Feamster
Georgia Tech
Atlanta, GA, USA
feamster@cc.gatech.edu

ABSTRACT

Studying network configuration evolution can improve our understanding of the evolving complexity of networks and can be helpful in making network configuration less error-prone. Unfortunately, the nature of changes that operators make to network configuration is poorly understood. Towards improving our understanding, we examine and analyze *five years* of router, switch, and firewall configurations from two large campus networks using the logs from version control systems used to store the configurations. We study how network configuration is distributed across different network operations tasks and how the configuration for each task evolves over time, for different types of devices and for different locations in the network. To understand the trends of how configuration evolves over time, we study the extent to which configuration for various tasks are added, modified, or deleted. We also study whether certain devices experience configuration changes more frequently than others, as well as whether configuration changes tend to focus on specific portions of the configuration (or on specific tasks). We also investigate when network operators make configuration changes of various types. Our results concerning configuration changes can help the designers of configuration languages understand which aspects of configuration might be more automated or tested more rigorously and may ultimately help improve configuration languages.

Categories and Subject Descriptors

C.2.3 [Computer-Communication Networks]: Network Operations—*Network management*

General Terms

Management, Measurement

Keywords

Network configuration, Network evolution, Longitudinal analysis

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

IMC'11, November 2–4, 2011, Berlin, Germany.

Copyright 2011 ACM 978-1-4503-1013-0/11/11 ...\$10.00.

1. INTRODUCTION

The behavior of a communications network depends in part on the configuration of thousands of constituent network devices, each of which is configured independently. In this sense, network configuration is a large, distributed program. Despite its importance in dictating the overall behavior of the network, we understand very little about the nature of configuration. Today, network operators implement high-level network tasks with low-level configuration commands; operators frequently make mistakes when making changes to network configuration [6, 14]. Although some studies have examined properties of configuration “snapshots” (*e.g.*, [15]), we have little understanding of how network configuration *evolves* over time.

Studying the evolution of network configuration over time can offer unique insights that cannot be learned from a single static snapshot. First, such a study can shed light on how network functions evolve over time: for example, we can learn more about how network configuration evolves, and which network tasks contribute to the growth in complexity. Second, because configuration changes are the cause of many errors, understanding the nature of how configuration changes over time—and what tasks configuration changes are associated with—can better inform configuration testing by helping create targeted test cases. Third, knowledge about configuration changes also offers valuable information about the parts of network configuration where operators spend time, which may help designers of configuration languages and configuration management systems design a better environment to make these tasks easier.

Towards these goals, this paper presents the first long-term longitudinal study of the evolution of network configuration, for two large campus networks: Georgia Tech and the University of Wisconsin. Comparing the evolution of network configuration across two different large campuses allows us to identify trends that are common across campuses, as well as practices that are specific to a particular network. We study the changes that operators make to the configurations of all routers, switches, and firewalls in these networks over a five-year period. We study the following questions:

- How does network configuration size evolve over time? How many of the changes are additions, modifications, or deletions? Do changes tend to occur at a specific time of day or day of the week?
- Which network-wide factors contribute most to configuration evolution?

- What parts of the configuration change most frequently, and why?
- Are there dependencies and correlations in changes to network configuration, and why?

We perform our analysis on two campus networks. Although the results may not generalize beyond these two campuses, they represent a case study and a first attempt to perform extensive longitudinal analysis on campus networks. Our study also complements previous studies, which have focused on enterprise and backbone networks [7, 18].

We find that configurations continue to grow over time, and a variety of factors—infrequent ones, such as, infrastructure expansion and policy changes, and more frequent ones, such as customer addition/modification—contribute in interesting ways to configuration evolution. We find that routers experience configuration changes more frequently, and that changes to routers involve a broader range of network configuration tasks than the changes that operators make to switch and firewall configurations. Given that there are far fewer routers than firewalls or switches in both networks, the relative frequency of configuration changes to routers highlights that many crucial day-to-day network operations tasks center around router configurations. Third, although there are many similarities in the configurations between the two campuses—both in the amount of configuration devoted to each configuration task and to the nature of the configuration changes—the network operators for each campus do have distinct practices (*e.g.*, the use of static ARP entries) that tend to appear on more general purpose devices, such as core routers. In contrast, devices that are more special-purpose such as firewalls tend to exhibit more common configuration change patterns. Finally, we note that configuration changes to switches and routers are sometimes correlated, suggesting the possibility for better configuration tools that can assist operators by automatically recognizing these dependencies.

Our findings suggest possible areas for improvement in configuration management and testing. For example, our findings concerning correlated changes across configurations can suggest improvements to the configuration process: a configuration management system could observe correlations in configuration changes and suggest changes that operators might make to low-level configuration constructs based on observations of past correlated changes for different high-level tasks. The management system can similarly help with debugging erroneous changes to configuration. Our observations regarding the high frequency of router configuration updates—which are often manual and hence error-prone—suggest the need for automated network-wide configuration management systems targeted specifically toward routers.

The rest of the paper is organized as follows. Section 2 presents background on network configuration, and on the configuration management systems used to make changes to network configuration. Section 3 describes the datasets that we use for this study and our approach. Section 4 describes our results for routers, switches, and firewalls. In each case, we observe characteristics for a snapshot of the network configuration; we also perform a longitudinal analysis of configuration changes over the five years of our study. Section 5 discusses possible future research directions for our results, including suggestions for how they might be used to improve configuration management and testing. Section 6 concludes.

2. BACKGROUND AND RELATED WORK

We use an archived history of network configuration files to analyze the nature and causes of configuration changes to switches, routers, and firewalls in the Georgia Tech and University of Wis-

```

1  ...
2  1.51
3  log
4  @Fri Feb 5 15:04:28 EST 2010
5  @
6  text
7  @a141 1
8  port-object range bootps bootpc
9  a160 4
10 object-group service 12-123-12-13-any-udp udp
11 port-object range bootps bootpc
12 object-group service 12-123-12-14-any-udp udp
13 port-object range bootps bootpc
14 d173 16
15 a188 9
16 object-group service 13-14-15-16-any-udp udp
17 port-object range bootps bootpc
18 object-group service 14-15-16-17-any-udp udp
19 ...
20 @

```

Figure 1: RCS file showing changes to network configuration.

consin campus networks. In this section, we describe the common configuration management systems used by the networks studied.

2.1 Configuration Management

Both campus networks use a configuration management tool called RANCID [17] for tracking and monitoring network device configurations. RANCID’s main function is to construct a catalog of devices, pull the current configurations, and store them in a version control system, which can allow network operators to track changes in the configurations to network devices. In addition to device configurations, RANCID also pulls hardware information (*e.g.*, cards, serial numbers). RANCID operates with devices from many vendors, including Cisco, Juniper, Foundry, and HP. More details can be found on the RANCID Web site [17].

In addition to accepting input from RANCID, the version control system also allows network operators to manually checkpoint and commit changes to the repository. Just as programmers use revision control to track changes to source code, network operators have adopted revision control to manage network devices by tracking changes to device configuration files. The CVS repository tracks changes by giving each revision of the configuration file a unique revision number and by storing metadata along with this revision, such as the time of change, comment, author of the change, and the sets of lines changed.

CVS stores all the metadata along with revisions for each configuration file in a single Revision Control System (RCS) file. Figure 1 shows an excerpt from an RCS file. The current revision number is on the top (line 2), followed by *log* (lines 3–5), which contain comments provided by the operator or RANCID explaining the nature and cause of the change (the revision date in this case). The *text* (lines 6–20) lines document the location, number and nature of the change; these lines allow us to track additions and deletions to the network configuration. We interpret an addition immediately followed by a deletion as a modification if the sum of the line number and the number of line changes of deletion is equal to the line number of addition added by one. For example, in Figure 1, lines 14–15 shows a modification ($173+16 = 188+1$).

2.2 Related Work

We provide a brief overview of related work. We first discuss various prior work on static analysis of network configuration to better understand network properties and to characterize or diagnose network configurations. Although the networking community

has seen only limited work on analyzing configuration *changes*, the software engineering community has performed extensive studies of changes to source code that relate to our study.

Static analysis. Previous work has performed static analysis of network configurations to study network properties and to help network operators diagnose misconfigurations in their networks. Caldwell *et al.* propose a system to automatically discover configuration templates and rules [2]. Maltz *et al.* performed a study of many enterprise network configurations to gain insights about the design and use of network configuration in enterprise networks [15]. The *rcc* tool performs network-wide static analysis of router configurations to detect configuration errors in BGP [6]. There is much other previous work on identifying and detecting misconfiguration [10, 14], but these tools focus on static analysis. Other work has used static analysis of router configuration to understand how operators configure specific functions (*e.g.*, route redistribution [12] and access control [19]), as well as to identify sources of configuration complexity [1]. The NetDB system pulled router configurations from the AT&T backbone network into a database for analysis of configuration snapshots [7]. Gottlieb *et al.* used NetDB to perform static analysis of network configuration to discover and construct configuration templates to help network operators automatically configure new sessions for ISP customers [8]; a follow-up system, Presto, constructs network configurations based on configuration templates [5]. These systems rely on operators to manually specify network configuration templates; ultimately, the type of analysis we perform in this study might help automatically identify common tasks and configuration idioms that could be automated.

Dynamic analysis. Other projects have explored the changes in configuration over time, albeit for smaller sets of network configuration, or over shorter periods of time. Sung *et al.* examined changes in stanzas over time for five enterprises VPN customers. They study which sets of stanzas change most often [18]. Chen *et al.* also examine the dependencies between configuration stanzas by analyzing changes [3]. Le *et al.* [11] aim to automatically generate rules to determine commands that a given stanza should contain. This approach requires pre-processing and domain knowledge to determine how to represent the commands as attributes of stanza; it also does not explore configuration changes over time.

Mining software repositories. The field of software repository mining, a sub-area of software engineering, mines version control systems to discover artifacts that are produced and archived during the evolution of a software program. Several studies have explored the longitudinal evolution of software. Lehman *et al.* studied the evolution of source code in several IBM products between 1969 and 2001 [13]. Eick *et al.* studied the phenomenon of code decay, whereby changes to a software system become increasingly difficult over its lifetime [4]. Those studying changes to software repositories focus on both changes to high-level *properties*, such as software complexity or maintainability; and on changes to *artifacts*, which is typically done by analyzing differences between versions in the version control system itself. A specific approach to mining software repositories called *source code differencing*, pioneered by Raghavan *et al.*, examines difference data in software repositories to look for additions, deletions, and modifications to source code over time [16]. This approach is similar to that which we take in this paper, where we explore additions, deletions, and modifications to network configuration files to understand changes to software artifacts. Kagdi *et al.* provide an excellent survey and taxonomy of the large body of work in mining software repositories [9].

| | Routers | Firewalls | Switches | Total |
|--------------|---------|-----------|----------|-------|
| Georgia Tech | 16 | 365 | 716 | 1097 |
| Wisconsin | 53 | 325 | 1246 | 1624 |

Table 1: Number of devices for each device type.

3. DATA AND ANALYSIS

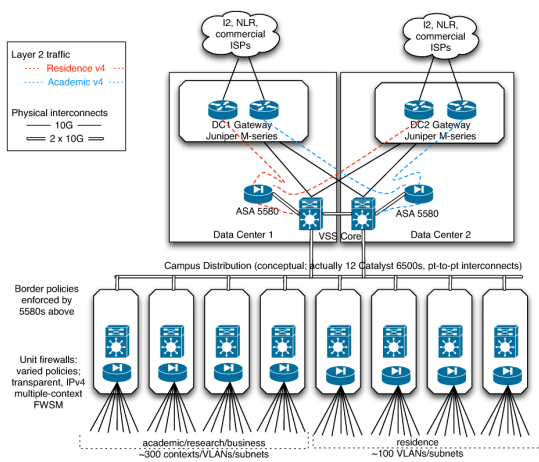
We describe the configuration data that we use in our study and our approach for postprocessing and analyzing the data.

3.1 Data

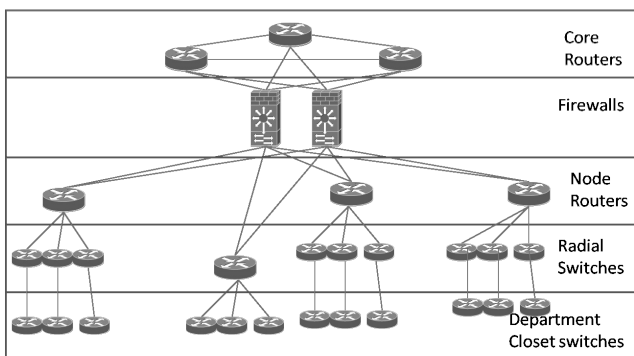
Both Georgia Tech (*GT*) and University of Wisconsin (*UW*) manage their network configurations with RANCID and CVS, the tools described in the previous section. We have collected five years of archived configuration files from all network devices (*i.e.*, routers, firewalls, and switches) in the two networks. Table 1 shows the number of network devices currently deployed in both networks for each device type. We note that both networks comprise mostly Cisco devices; configurations are stored in the RCS format. Our analysis tools thus focus on parsing Cisco’s configuration language and the RCS format. The border routers in both networks are Juniper routers. Because the configuration language syntax differs significantly from Cisco’s configuration language, we omit the border routers from our analysis for both networks. We now describe the design of each network and how they use RANCID and CVS in more detail.

Georgia Tech. Figure 2a shows the topology of the Georgia Tech network. The Internet connection originates from two multi-homed border routers; two physical firewalls are deployed close to the border. Departments, research groups, and residence halls are divided into separate subnets and assigned unique virtual LANs (VLANs). Each subnet has different access control policies, enforced with a separate virtual firewall. The network has hundreds of virtual firewall instances that are not shown in the figure. Hosts are connected to the network via switches at the edge. RANCID pulls the latest configuration files every three hours and saves the snapshot of the files. A CVS “commit” only occurs when the system detects a change in the latest configuration. In this network, the operators rarely make manual backups of the configurations; as such, most revisions in the repository are created by RANCID.

University of Wisconsin. The *UW* network, shown in Figure 2b, is similar to *GT* in many aspects; like the *GT* network, the *UW* network topology is hierarchical, with Internet traffic entering through two border routers and redirected through two major border firewalls for access control. Border routers are distinct from core routers, and they reside one level higher than the core routers. Figure 2b does not show the border routers. The *UW* network contains four tiers: the core and node layers, which primarily comprises routers; and the radial and access layers, primarily comprising switches. The access switches connect computers in the different departments to the network. Unlike *GT*, the RANCID setup at *UW* pulls snapshots for only firewall configurations. Similarly, commits to firewall configuration files only happen when the system detects a change to the latest configuration. Operational policies and tools at *UW* force operators to manually snapshot and commit all other configuration changes before they can be pushed to the devices. As such, most revisions are created by operators. This different approach does not affect our results, as it simply reflects a difference in the mechanism on who pulls snapshots, detects changes and commits to the repository.



(a) Georgia Tech



(b) University of Wisconsin

Figure 2: GT and UW campus networks.

3.2 Analysis

We analyze the evolution of network configuration changes by analyzing the RCS files for changes and extracting information about how each configuration file changes over time. This section describes our analysis, which has two major parts: (1) abstracting higher-level operator tasks from low-level configuration changes; (2) performing change analysis on the RCS files.

3.2.1 Developing abstractions for functionality

The first task in analyzing the network configuration is to derive abstractions of high-level functions and tasks from low-level configuration commands. This helps us more readily identify the types of tasks that network operators most commonly perform when making changes to the network configuration and the amount of configuration devoted to each task. We do so by manually associating each command in the configuration language with a unique function. Table 2 shows the list of the functions we use in our analysis, with several example commands for each of the categories. Given this categorization, we construct a database that maps configuration commands to one of these functions. Due to the abstraction we apply to the configurations, we do lose fine-grain information that can be valuable to our study, but the goal of this work is to offer a high-level analysis of network configuration and its evolution.

Although our categorization is extensive, the mapping does not cover all configuration commands: some commands are difficult to classify or do not map to any particular function (e.g., commit

| | Meaning | Examples |
|------|----------------------------|---------------------------------------------------------------------------------------------------------|
| mgt | device management settings | username, password, telnet, ssh, logging, aaa, clock, console, radius-server |
| I1 | interface/port settings | interface definition, bandwidth, switchport, description, duplex |
| I2 | layer 2 settings | arp x.x.x.x, mac-address, ip proxy-arp, arp timeout, spanning-tree |
| vlan | VLAN settings | vlan definition, switchport mode trunk, switchport mode access vlan, set vtp, set vmps |
| I3 | layer 3 related | ip address x.x.x.x, ip gateway x.x.x.x, ip route x.x.x.x, nat, router bgp/ospf/rip, router-id, neighbor |
| acl | access control | object define, access-list, permit, deny |
| sec | security related | vpn, ipsec, crypto, webvpn, anyconnect, ssl, tunnel-group, flood-guard |
| cflt | control filtering | prefix-list |
| qos | QoS | policy-map, class-map, service-policy, port-channel load-balance, set qos |

Table 2: Functionality map. We map each command to a specific functionality to effectively analyze configuration files.

checksum values, comments, banners, start/end markers, and version info). Although not complete, we believe our functionality map is extensive enough to cover 93% of all configuration commands and 91% of all configuration changes in University of Wisconsin, and 98% of all commands and changes in Georgia Tech, thus complete enough to shed light on how operators devote time to various high-level tasks.

To abstract configuration changes that involve multiple lines of configuration, we group configuration into *stanzas*, each of which is the largest contiguous block of commands that encapsulate a piece of the network functionality. Cisco’s configuration language provides special symbols (“!”) for separating stanzas. We analyze configuration changes according to how stanzas change in total.

3.2.2 Change analysis

After developing abstractions for the configuration tasks and grouping configuration commands by stanza, we perform three types of analysis on the configuration changes to better understand how configuration evolves over time:

- *Snapshot and change analysis.* As a baseline for understanding the composition of network configuration in the two networks, we analyze a snapshot of the network configuration for each network and examine the number of lines of configuration devoted to each function. To understand where operators devote their efforts when modifying network configuration, we then study the extent to which network configuration changes are associated with different functions that the configuration performs. We also explore how many of the changes corresponded to additions, deletions, and modifications. We use a Python library called *diffib* to infer what constitutes an addition, deletion, or modification, by comparing two consecutive revisions and producing a “diff” in a context format. We evaluate the size of the diff in terms of number of lines. We assume the library is able to correctly identify additions, deletions, and modifications. We also correlate the line number difference from the first version to the

| <i>Georgia Tech</i> | <i>add</i> | <i>del</i> | <i>mod</i> | Total |
|---------------------|------------|------------|------------|---------|
| Routers (16) | 31,178 | 27,064 | 262,216 | 326,458 |
| Firewalls (365) | 249,595 | 118,571 | 171,005 | 539,171 |
| Switches (716) | 216,958 | 20,185 | 116,277 | 353,420 |
| Rtr avg. per device | 2,324 | 1,692 | 16,389 | 20,404 |
| FW avg. per device | 684 | 325 | 469 | 1,477 |
| Swt avg. per device | 303 | 28 | 162 | 494 |

Table 3: Number of add/del/mod for all devices and average per device for each device type in *GT*.

| <i>UW-Madison</i> | <i>add</i> | <i>del</i> | <i>mod</i> | Total |
|---------------------|------------|------------|------------|-----------|
| Routers (53) | 79,202 | 38,288 | 154,407 | 271,897 |
| Firewalls (325) | 193,499 | 73,827 | 161,863 | 429,189 |
| Switches (1246) | 1608,512 | 213,910 | 1,768,384 | 3,590,806 |
| Rtr avg. per device | 1,494 | 722 | 2,913 | 5,130 |
| FW avg. per device | 595 | 227 | 498 | 1,321 |
| Swt avg. per device | 1,291 | 172 | 1,419 | 2,882 |

Table 4: Number of add/del/mod for all devices and average per device for each device type in *UW*.

last revision to verify that the numbers of line additions, deletions, and modifications derived are consistent.

- *Longitudinal analysis.* To understand how the network configuration has evolved over time, and the primary factors underlying different evolution patterns, we perform a longitudinal analysis of configuration changes, using five years of RCS files from each network.
- *Correlation analysis.* To understand whether certain aspects of the network configuration frequently change together, we perform a correlation analysis to determine network configuration tasks that operators commonly perform together.

Our toolkit parses each RCS file and aggregates information for the snapshot and basic change analysis, such as how many lines of configuration are devoted to each function, how many changes occur and what types of changes (*i.e.*, addition, deletion, modification) occur, when and where these changes happen. It also parses the data and author information from the RCS files determine the frequency of changes made on certain days, times, and by certain authors. The toolkit then inserts this information into a database that performs the longitudinal and stanza analysis. Correlation analysis requires additional software functionality, beyond gathering first-order statistics. To perform this analysis, the toolkit parses the RCS files and extracts every version of the configuration file. The configuration files are parsed, and stanzas from each configuration file are stored in a database, using which we can study correlated changes.

4. THE EVOLUTION OF CONFIGURATION

We now present results on network configuration evolution in the two campus networks. In the process, we identify change characteristics that the campuses share in common and those where they differ. We augment our analysis with operator discussions to shed light on the impact of operational practices on configuration evolution in the respective campuses.

4.1 Overview

What types of changes occur? We begin by measuring the number of configuration commands changed for each device type and normalize the results by the number of devices, as shown in Tables 3 and 4.

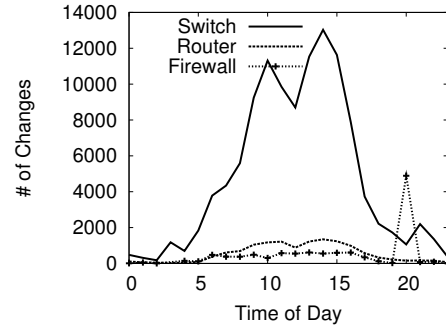


Figure 3: Changes to the *UW* network configuration over a day.

We make the following high-level observations. First, the networks have a similar number of firewalls, but *UW* has nearly four times as many routers and almost twice as many switches. Second, although most changes happen in firewalls and switches (with changes on switches dominating), routers, which are far fewer in number, tend to have many more configuration changes per device (on average) in both the *GT* and *UW* networks. We delve into router configuration changes in Section 4.2. Third, we find interesting differences between the two networks: in *UW*, switches clearly undergo a large number of modifications relative both to other devices in the same network, and to switches in the *GT* network. *GT* has relatively more changes to its firewall devices than *UW*. We delve into firewall and switch configuration evolutions across the two campuses in Sections 4.3 and 4.4, respectively.

When do changes occur? Figure 3 shows how configuration changes are distributed across hours of the day for *UW*. The pattern shows that most changes to switches and routers are made during the work hours. However, many of the configuration changes to firewalls happen later, as shown by a large increase in commits around 8 p.m. In contrast to routers and switches, the firewall configurations are often managed by a department-specific system administrator and then transferred to a member of the campus IT staff, who checks them for consistency and eventually uploads the files to the firewalls and commits them toward, or after the end of, working hours. Figures 4a and 4b show how changes to network configuration are distributed across the months of the year in the two networks. We observe that the number of configuration changes peaks during August. Our conversations with operators revealed that, in both networks, large network-wide changes are put off until the summer when many of the users are away and disruptions might have the least detrimental effect on users of the network.

Table 5 summarizes the main findings from our analysis of these two campus networks. We note several interesting characteristics that may ultimately help understand network configuration and its evolution better, and ultimately devise better methods or tools for device configuration and management. The rest of this section discusses these findings and others in more detail.

4.2 Routers

We first focus our analysis on routers. The main function of routers in the network is to support layer-3 routing, but, interestingly, many of the configuration changes on routers are associated with tasks other than layer-3 routing.

4.2.1 Snapshot and change analysis

Commands other than those involving routing change frequently over time. Figures 5a and 5b present the results from

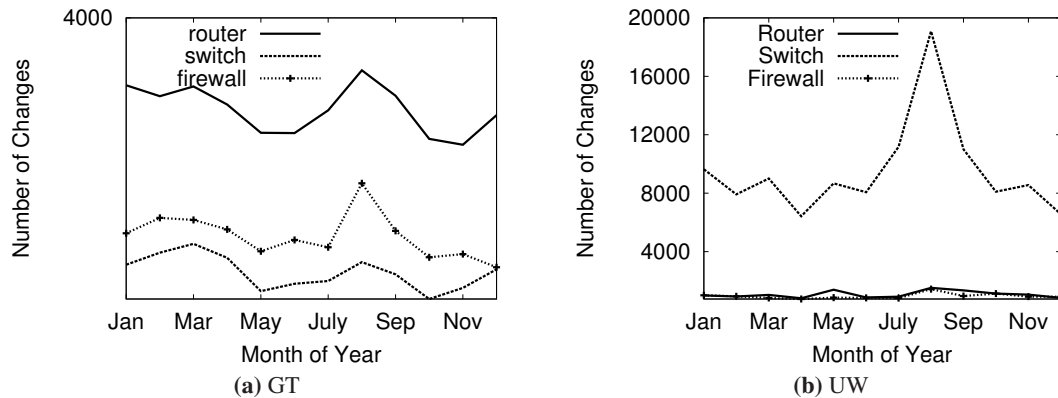


Figure 4: Changes to the GT and UW configurations over one year. Changes to the configuration increase dramatically before the start of the academic year.

| | |
|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------|
| Many changes in routers are security-related. Bulk changes in routers come from security-related tasks, and different campuses use different mechanisms, thus showing dissimilar patterns of changes. | §4.2.1, §4.2.2, Figure 5, Figure 6 |
| Deployment location matters. Where the device is placed within the network, e.g., border vs edge, have significant influence on the size of change and what changes. Intuitively, devices at the border undergo a lot more changes than edge devices. In addition, the changes are more restrictive at borders and are more closely monitored. | §4.2.1: Figure 6, §4.3.1: Figure 9 |
| Variety of network-wide tasks contribute to changes in devices. This includes network expansion, change of policy, addition of new features or modification of existing, and provisioning new customers or deleting subnetworks. | §4.2, §4.3, §4.4 |
| Specialized devices are similar across different networks. Highly specialized network appliances, like firewalls, tend to have very similar traits when it comes to configuration and its evolution. | §4.3.1, §4.3.2 |
| Correlated changes in firewall devices occur within the <i>acl</i> command. In firewalls, commands are highly concentrated on <i>acl</i> stanzas. However, we found highly correlated behavior within the <i>acl</i> stanza, namely object definition and actual access lists. | §4.3.3 |
| Switch configuration is highly port-centered. Switches are configured port-by-port, and batch changes on interfaces is frequent. <i>I1</i> has a variety of subcommands, hence large correlation occur among commands in switches, although they differ by network and tasks assigned. | §4.4, §4.4.3 |

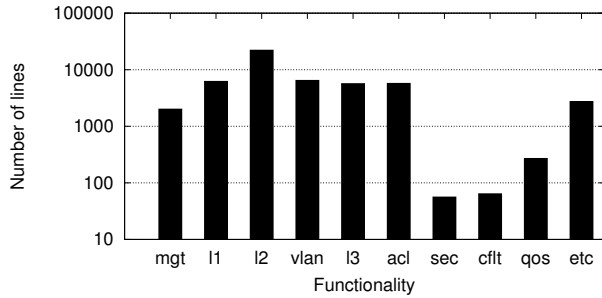
Table 5: Main results, and where they are described.

analyzing a snapshot of recent configuration files from all routers from each campus. Figures 5c and 5d present the changes occurring in both networks over five years for the routers in each campus. In both campuses, the *I1* and *I3* commands are added, deleted, or modified most frequently. In the GT network, *I2* changes are extremely frequent, due to changes to static ARP entries, which we will explain in more detail later. There are many more modifications to the UW network in general, many of which result from modifications to the existing *I1*–*I3* and *vlan* commands. Further investigation shows that VLAN interface definitions in UW router configurations are constantly changing due to addition, update and removal of “customers” of the UW network (departments or groups of users with

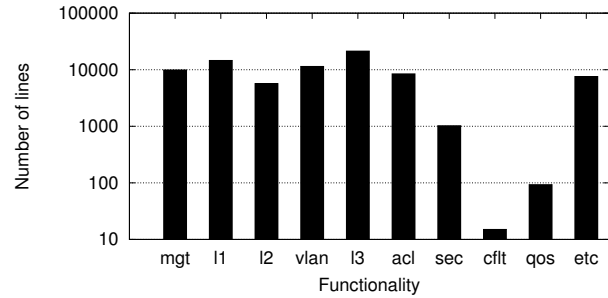
specific needs, each group with its own VLAN). This explains the large amount of the *I1*, *I2* and *vlan* configuration changes. As we show later, the changes in *I3* are due to security related changes to a particular router.

Distinct security-related practices are found. Notably, many configuration changes to routers involve security-related tasks, although the mechanisms that each network uses differ. Each campus has security practices that use specific router configuration commands. The significant amount of the *I2* command changes in Figure 5c shows one example. Further inspection reveals that this is due to *static ARP* entries in routers deployed in residence halls in the GT network. The GT network maps each end host machine’s MAC address to a static IP address in residence halls. The main goal is to provide better security and more fine-grained monitoring. This part of the configuration constantly undergoes changes as residents come and go. We examine changes to the stanzas in the routers for the different networks in Figures 6a and 6b. Our analysis shows that, on average, most routers in the UW network have configuration changes that reflect a consistent breakdown across functions, regardless of the router. The exception to this pattern is device 6, which has significantly more layer-3 changes than other devices. Upon further examination, we discovered that router is a *blackhole* device. Blackhole filtering, or null routing, provides a firewalling service with minimal impact on performance, by routing certain traffic to nowhere, and not providing any information regarding what happened to the traffic sources (no “no route to destination” reported back; hence the term blackhole). This practice uses “static routes” to manipulate the blackhole behavior; thus, we see constant changes to *I3* stanzas in this router device. This is another example of using specific router configuration commands to achieve certain security goals. The GT network also has routers with changes that are distinct from one another. Devices 5, 6, 7, and 8 are “resnet” routers in the campus dormitories with static ARPs.

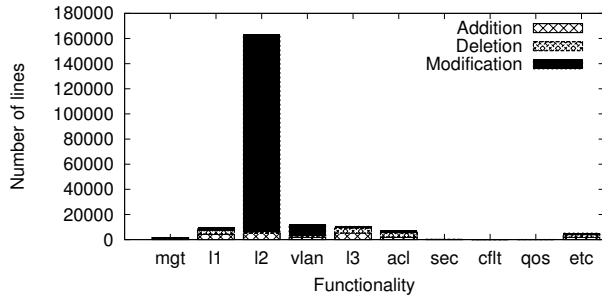
Number of changes in core routers are noticeably larger than non-core routers. In Figure 6a, devices 1, 2, and 3 in the GT network have many *I1* and *vlan* changes, which are distinct from other routers. These three routers are core routers through which many VLANs are trunked. Core routers are responsible for routing traffic between the upper border gateway and subnets at the edge. Many VLANs for edge subnets are aggregated (or trunked) at these routers, so changes in each subnet can result in substantial changes



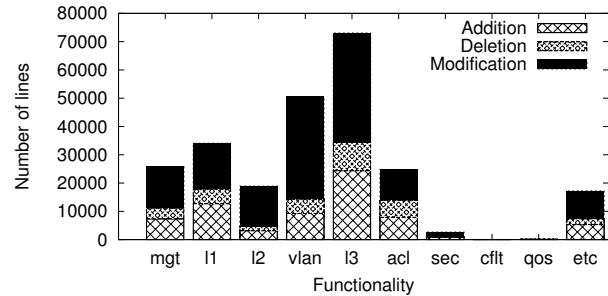
(a) Static analysis of latest snapshot (logscale) - *GT*



(b) Static analysis of latest snapshot (logscale) - *UW*

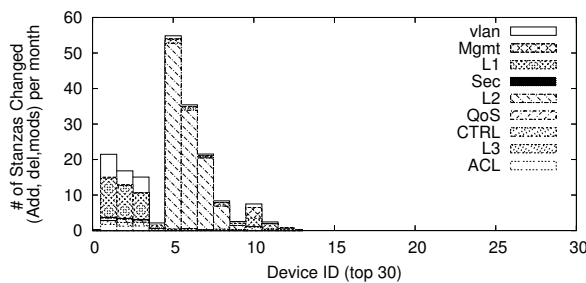


(c) Change characteristic over five years - *GT*

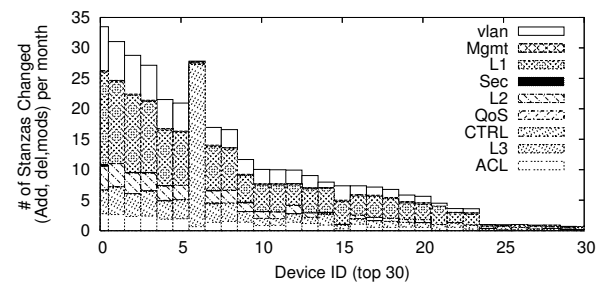


(d) Change characteristic over five years - *UW*

Figure 5: Static analysis and change analysis for routers. (a) and (b) shows a static analysis of the latest router configurations all combined for each campus. (c) and (d) shows the number of additions, deletions and modifications of each command in a row-stacked fashion, all devices combined over 5 years.



(a) *GT*



(b) *UW*

Figure 6: Number of changes to each router per month for the *GT* and *UW* networks.

at the core router. This characteristic is apparent in the *UW* network as well, as shown in Figure 6b. Devices aligned on the left of the figure (device 0-5) are the core routers, which are identified by their device names. The number of changes per month for these routers are significantly larger than other non-core routers. The changes are concentrated on *l1* and *vlan* stanzas.

This analysis shows that, depending on the function of the router and its placement in the network, the changes to its configuration may differ significantly. Different design and operational practices in two networks result in distinct change patterns in routers.

4.2.2 Longitudinal analysis

Network expansion causes steady increase in configuration size. Figures 7a and 7d show how the number of lines changes, for each router function. The figures account for all routers that were deployed in the network at a single time epoch. There is a gradual

increase in overall number of lines in *UW* routers while it is quite steady for the *GT* routers. This growth appears to be caused by devices being added to the network, as shown in Figures 7c and 7f. The number of *GT* routers has grown relatively slowly over the past five years, while the *UW* network has added many routers over the time period. In 2007, there are 7 cores, 10 nodes, and 6 devices playing miscellaneous roles. By 2010, however, there are 9 cores, 12 nodes, 4 radial and hot spares, 1 VPN router, and several access routers. There is one considerable period of growth in routers in the *GT* campus network around July 2008 (Figure 7c), which caused a significant increase in *l2*, as shown in Figure 7a. This growth in routers corresponded to the expansion of the “resnet” network in *GT*.

Adding features, modifying existing commands and provisioning new customers using VLANs cause a lot of changes. We now examine the evolution of a typical router configuration. We calculate the median number of lines of each function among all

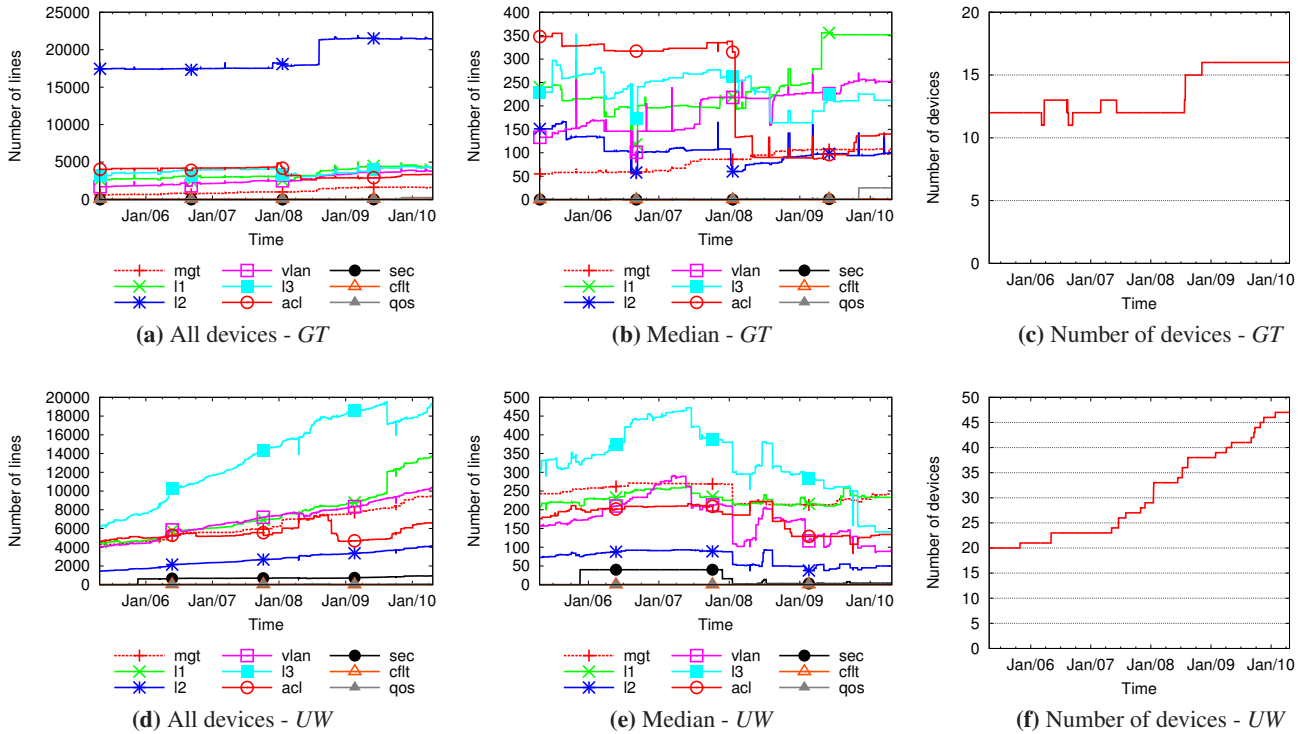


Figure 7: Longitudinal analysis of routers over five years. The graphs show the total number of lines of configuration for all devices, the median number of lines of configuration per device, and the total number of devices over time.

routers at every time epoch. Results are shown in Figure 7b and 7e. For the *GT* network, *l1* is increasing over time, while *l3* stays relatively steady. The sudden increase of *l1* seems to be caused by new routers added to the network, again around July 2008. This is understandable, as new router devices would have a bulk of ports defined and opened when first deployed. A sudden drop in *acl* around January 2008 was caused by a major configuration clean-up: several core routers purged hundreds lines of access control lists, and disabled *AppleTalk* routing and related instances. Several subnets were removed from the network at this point, causing hundreds of *acl* to be obsolete. For the *UW* network, the number of *vlan* and *l3* lines for the median device generally decreases over time as new devices are added, mainly because the newly added devices have far fewer lines with these functionalities. The exception to this is the period from from Jan 2008 to Jun 2008, where there is growth in the configuration constructs for these two functions. There are two reasons for this: first, the median device remains the same over this period; as more VLANs are added, there is a steady growth in *vlan* statements; and second, the introduction of multicast related configuration and new filters *acl* result in a sharp jump in *l3* and *acl* configuration lines.

On the whole, we find that a variety of network-wide factors contribute to configuration changes: network expansion, addition of new features or modification of existing ones, and provisioning of new customers. No one factor appears to be the dominant reason for configuration changes, although the impact of the first two factors, and the configuration changes that result from them, appear less often.

| UW | | GT | |
|--------------------|-----|--------------------|-----|
| Correlated Stanzas | % | Correlated stanzas | % |
| l1, vlan | 35% | l1, vlan | 26% |
| l1, l2, l3 | 13% | l1, l2 | 18% |
| l1, l3 | 14% | acl, l1 | 10% |
| acl, l1, l2, l3 | 7% | l1, l3 | 9% |
| acl, l1 | 5% | l1, l3, acl | 5% |

Table 6: Correlated stanza changes for routers.

4.2.3 Correlation analysis

We first examine the extent to which changes to certain configuration constructs are correlated with others. To perform a certain high-level operational task, an operator has to change (add, delete, or modify) many different stanzas simultaneously. Indeed, we found that a fraction of changes—13% and 10% of all change events in *UW* and *GT*, respectively—are indeed performed in a correlated fashion, involving two or more stanza types. Such correlation has implications for the design of configuration management and debugging tools, as we discuss in Section 5.

Table 6 enumerates the five most common types of correlated stanza changes. The two campuses agree in three of the five: (1) *l1* and *vlan*: Although routers provide routing capabilities, they also have to control reachability (i.e., offer basic connectivity between, as well as ensure separation of, multiple subnets), which is often achieved through the use of VLANs. Thus, a considerable amount of change to *l1* and *vlan* occurs; (2) *l1* and *acl*: These change together because adding, deleting, or modifying networks (an *l1* change) often causes access control lists corresponding to the networks to be added, deleted, or modified to ensure consistency; and

(3) *l1* and *l3*: These correlated changes occur because modifying interfaces and subnets in the routing stanzas permits redistribution of reachability information.

Correlated *l1* and *l2* changes (the second most common in the *GT* network) occur when defining interfaces and setting a spanning-tree protocol for that interface for which each port has a variety of options. Similarly, correlated *l1*, *l2* and *l3* changes in *UW* occur when interfaces are added, the spanning tree is modified for the interfaces, and the routing protocol is changed to include the subnet or to setup routing protocol/peering adjacency over the interface.

4.3 Firewalls

A firewall's main function is to perform access control. As expected, we find that configuration changes are concentrated on *acl* statements (Figures 8c and 8d). Therefore, we focus most of our analysis instead on how firewalls are used in the respective campuses and the similarities and differences in how they are used. We find that, despite the highly specialized nature of firewalls, their usage across the two campuses is similar (see Figures 8 and 10).

4.3.1 Snapshot and change analysis

The change characteristics are similar between two campuses.

Figures 8a and 8b show the static analysis of the latest firewall configuration snapshot. As expected, most of the configuration is dedicated to access control (note that graphs are in log scale). This characteristic holds for changes as well, as shown in Figure 8c and 8d. Most changes are dedicated to access control while other types are mostly insignificant. High similarity in snapshot and change analysis between two different campus networks is intriguing. These similarities also appear in the longitudinal analysis shown in Figure 10. This similarity seems to result from the fact that devices like firewalls are designed to provide specific functionality to the network. Most changes involve *acl* for both campus networks. The actual content of commands will still differ, since network policies from two distinct networks are far from similar.

Deployment location matters. Figure 9 shows the top 30 firewall devices sorted in terms of number of stanza changes per month. Again, the graph shows that changes are heavily concentrated on *acl* stanzas. The number of changes per firewall is skewed, with a few firewalls experiencing many more changes and a large set of remaining firewalls experiencing minimal changes per month. Upon closer examination, we find that the top firewalls in the *GT* network (Figure 9a) are close to the border, and are physical devices where multiple virtual firewall instances actually spawn from. All changes to the virtual firewalls are reflected on these border firewalls, hence showing a lot of changes compared to others. This is true for the *UW* network as well. We also found a practice in the *GT* network that close-to-border firewalls are periodically backed up and commit to the CVS every six hours, while unit firewalls which are deployed closer to the edge are not managed that intensively. We assume this is due to the higher reliability requirement for the border firewalls compared to the unit firewalls; failure or misconfiguration on border firewalls can bring destructive consequences to the whole network.

Distinct practice using *sec* stanzas is captured. There are some noticeable *sec* stanzas on the *UW* network in Figure 9b. Inspection of the actual commands changed show that operators are turning off failover for the interfaces (internal and external). This prevents the virtual firewalls from failing over when the interfaces or connected devices are taken down. The anomalies occur in the firewalls used primarily by the campus IT department.

In summary, the two campus networks show very similar config-

uration change and usage patterns when it comes to firewalls. The majority of the changes occur at the border firewalls, where the changes on virtual firewalls are aggregated to. Border firewalls are also managed more intensively compared to simple unit firewalls at the edge.

4.3.2 Longitudinal analysis

Figure 10 shows the longitudinal view of firewall configuration evolution over a span of five years. The number of lines for *acl* commands increases sharply over five years. For the *GT* network, there was a factor of six increase. In the case of the *UW* network, firewalls were deployed starting in January 2006, and around 260 firewalls were added through April 2010.

Figures 10b and 10e show how the configuration of a typical firewall evolves over time. The small spike in the *GT* network around July 2007 is caused by adding new firewall machines and moving a substantial amount of access control lists to those firewalls. ACL configurations in previous firewalls are removed several days after the migration occurs. A slight increase of *qos* commands occurs around July 2008. Further inspection in the configuration shows a global policy map, which includes default inspection ports and Session Initiation Protocol (SIP) ports, is defined and inserted into around 40 firewalls. This policy map enforces further inspection on the specified services. In the *UW* network (Figure 10e), we observe a similar increase in *qos* commands around January 2008. These define two class maps: one for SIP ports as with the *GT* network, and one for a specific list of management services, ranging from DNS and rsh to SunRPC.

The decrease of *mgt* commands around January 2008 is due to the removal of the "pdm" (PIX Device Manager) commands from all firewalls. This command allows the device to determine who will have access to the management GUI for the firewalls. The removal of these commands prevents access to the GUI and forces the departments to submit all changes to the campus IT team, reflecting a new campus-wide security policy instituted in *UW* in 2008.

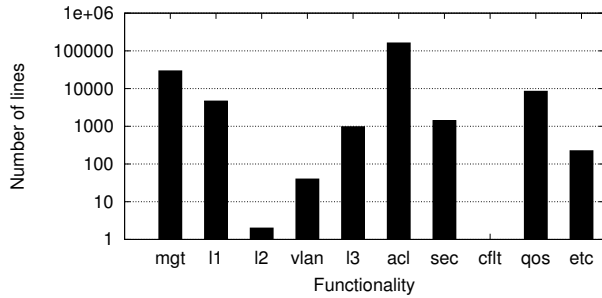
As with routers, we observe that a variety of network-wide factors, ranging from infrastructure expansion, to policy changes, and addition/modification of features, contribute to firewall configuration evolution.

4.3.3 Correlation analysis

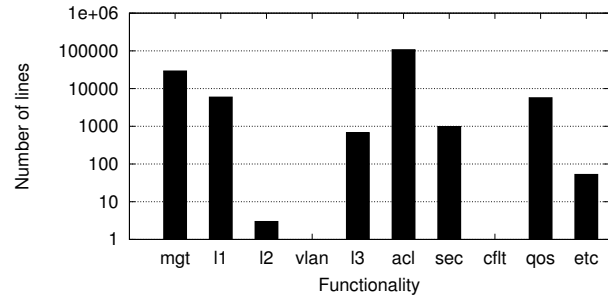
Correlation analysis on firewall devices has not shown any specific correlated change behavior on stanzas, as most changes are solely made to *acl* stanzas. Clearly, the functionality abstraction we employ in this paper is too high-level for correlation analysis on firewalls. We do witness many correlated changes within the *acl* commands. Specifically, if there is a change in the network object group definition (where operators can combine multiple IP addresses, ports, subnets to separate groups), there are also changes in the actual access control lists which reference that particular network object. More fine-grained classification of commands within the *acl* stanza may show interesting correlated change behavior. For example, there might be correlated changes within the access control lists; such an exploration would make a good direction for future work.

4.4 Switches

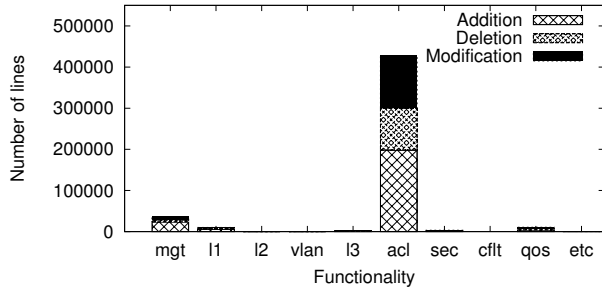
In this section, we study changes in switch configurations. We find: (i) interface definition and related subcommands dominate the changes in switches (Figures 11c and 11d); and (ii) extensive correlated changes (Table 7).



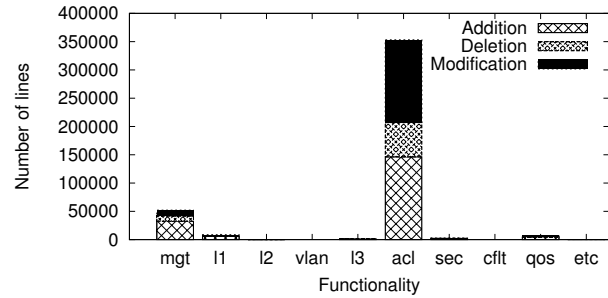
(a) Static analysis of latest snapshot (logscale) - *GT*



(b) Static analysis of latest snapshot (logscale) - *UW*

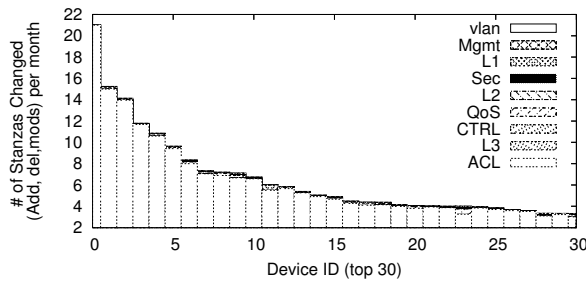


(c) Change characteristic over five years- *GT*

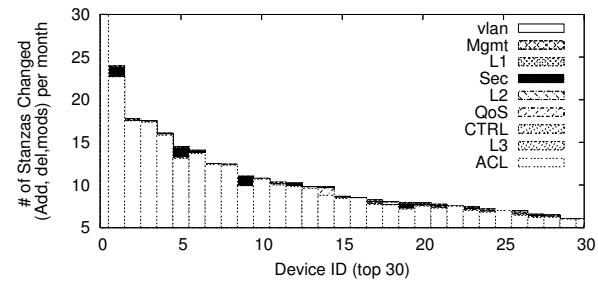


(d) Change characteristic over five years- *UW*

Figure 8: Static analysis and change analysis for firewalls. (a) and (b) shows a static analysis of the latest firewall configurations all combined for each campus. (c) and (d) shows the number of additions, deletions and modifications of each functionalities in a row-stacked fashion, all devices combined over 5 years.



(a) *GT*



(b) *UW*

Figure 9: Number of changes to each firewall device per month for the *GT* and *UW* networks.

4.4.1 Snapshot and change analysis

Many changes involve management, layer 1, layer 2 and VLAN commands. Figure 11 shows the static snapshot and change analysis of switches from both campuses. If we focus on changes of the configuration, Figures 11c and 11d show that there are many *mgt*, *l1*, *l2* and *vlan* command changes. Note from Figures 11a and 11b that these lines also constitute a majority of any given switch configuration snapshot. As seen previously, *l1* and *vlan* changes occur together naturally. *l2* changes are understandable as switches can offer layer 2 connectivity. The abnormality seems to be the significant number of changes in *mgt* commands. A majority of *mgt* changes in switch devices are in fact *snmp trap* and *logging* commands; *snmp trap* is for sending events to the SNMP server when particular conditions are met, and *logging* configures the logging behavior for the interface. The default is to define SNMP trap and logging commands along with the port configuration in the *UW*

network. This induces the bulk of the *mgt* and *l1* changes to the network.

Certain practices shape how switch configuration evolves. The *GT* network has many additions, while the *UW* network experiences many modifications. Most of the changes in the *GT* network are actually due to additions in *l1* and *vlan*, followed by *mgt* and *l2* commands. However, the *UW* network looks different; there are a lot of modifications in general, with changes in *mgt* and *l1* followed by *vlan* appearing to dominate the changes. We found that this resulted from a large number of modifications that operators made to a majority of switches in the *UW* network, including interface and their subcommands, *snmp trap* and *logging*. From this behavior, it appears as though operators of the *UW* network have decided to uniformly enable or disable some functions on ports across most of its switches.

Specialized switch devices can be identified by their distinct

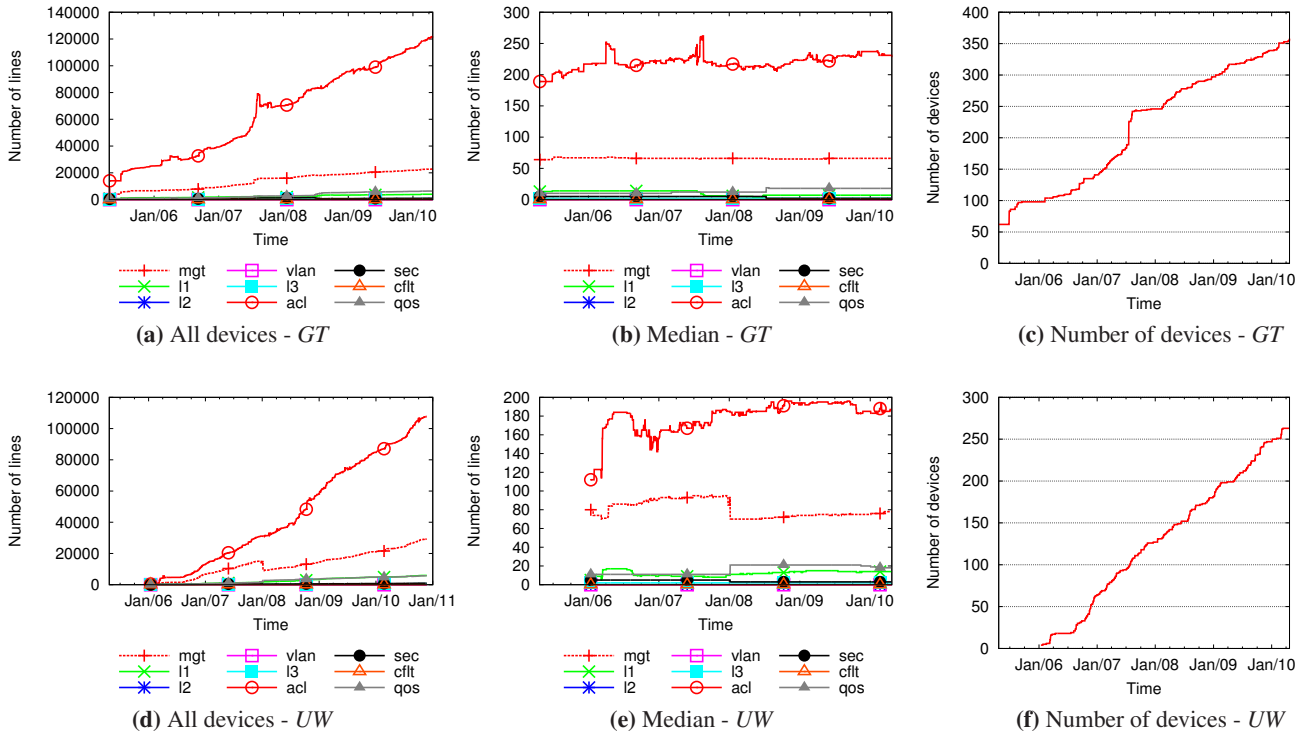


Figure 10: Longitudinal analysis of firewalls over five years.

patterns. Figure 12 presents the number of changes per stanza type, for each of the top 30 switches in both networks. The switches in *UW* are similar, in that most of the changes are in the interface commands, with a few devices having a significant amount of VLAN additions. Unlike *UW*, *GT* switches have several different patterns. *mgt* accounts for a significant amount for Device 1. This device is a switch for the SoX (Southern Crossroads) deployed within the Georgia Tech campus, and the changes to *mgt* are due to changes to the device authentication configuration, namely *aaa*, *radius-server*, and *file* stanzas. This can be caused by the fact that multiple users manage this device. There is also a group of devices (devices 25–29) that have a lot of *sec* stanzas. Further inspection shows these devices are *VPN* boxes with many *ipsec* changes. These devices can be classified as specialized switches that provide unique functions of secure connection establishment between the internal and external network.

4.4.2 Longitudinal analysis

Evolution trend is heavily influenced by how switches are added, removed or swapped, as well as their initial configuration. Figures 13a and 13d show a gradual increase in the total number of configuration lines, for both networks. The increasing trend is mainly due to the increase in number of switch devices, as shown in Figures 13c and 13f. An abnormal spike appears around September 2010 in Figure 13a, when 150 new switches were added to the *GT* network. In Figure 13b, the *l2* command experiences several decreases and increases. Inspection of the switch configurations reveals that around February 2006, a set of new switches with 24 ports was added to the network; older switches had more ports and hence had more verbose configuration. The other increases and decreases are of the same nature: the trend depends on what type of switches are added to the network with specific initial configu-

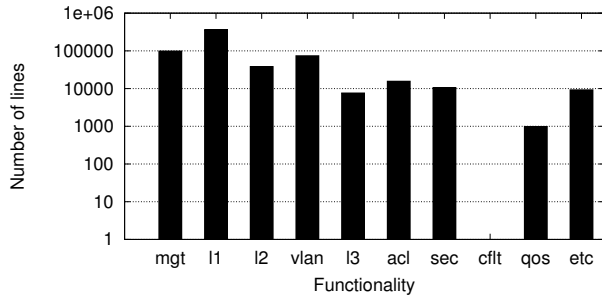
rations. This phenomenon is similar to the decreasing number of *vlan* commands in Figure 13b; switches with small *vlan* instances are being added to the network continuously.

Distinct practices and decisions in *UW*. Turning to *UW*, in Figures 13d and 13e, we see that in January 2008, the number of lines for *sec* doubles and *mgt* decreases. Additions to *sec* commands occurred due to changes to all interfaces on all switches to make the network more resilient to bursty traffic, by installing a limiting mechanism. The *mgt* drop in Jan, 2008 is due to operators removing the *SNMP* feature from all interfaces. In both cases, the large changes reflect updates to all switches in the network.

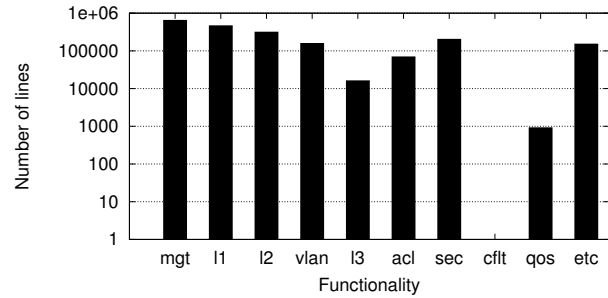
We find that operational and network expansion practices of individual networks play an important role in shaping how the switch configuration evolves; understanding this evolution can facilitate the design of better network management tools.

4.4.3 Correlation analysis

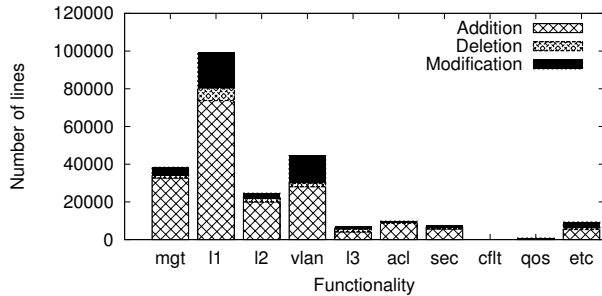
In Table 7, we examine the set of correlated stanzas for the switches across the two campuses. Most changes in *UW* include either *l1* or *mgt* whereas most changes in *GT* include either *vlan* or *l1*. However, both networks agree on a set of correlated changes: (1) *vlan* and *l1*, (2) *acl* and *l1*, and (3) *mgt* and *l1*. As with other devices, interface definition and VLAN configuration commands have high correlation, as do ACLs that reference certain interfaces. *l1* and *mgt* correlates well in switches, because many default interface configuration in switches have *snmp trap* and *logging*. The *UW* network has an additional *l2* correlation with *l1* and *mgt*. This is due to heavily using spanning-tree related subcommands on interfaces. Although not shown, *UW* does in fact contain *mgt* and *vlan* correlations; *UW* contains half as many correlated occurrences of *mgt* and *vlan*. In addition to those, *GT* includes a unique combi-



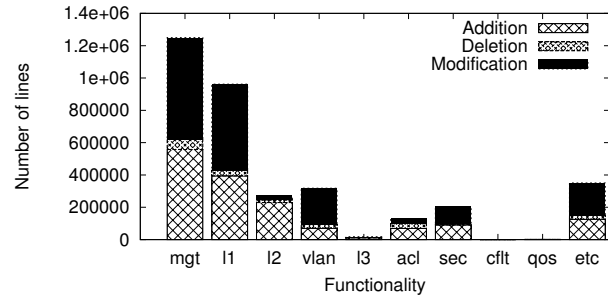
(a) Static analysis of latest snapshot (logscale) - *GT*



(b) Static analysis of latest snapshot (logscale) - *UW*

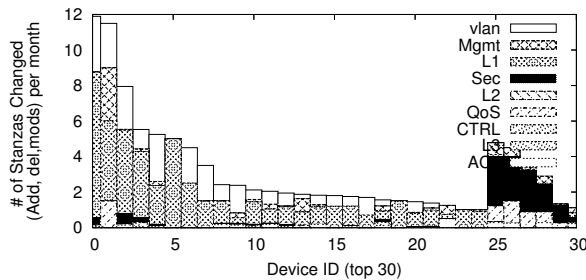


(c) Change characteristic over five years - *GT*

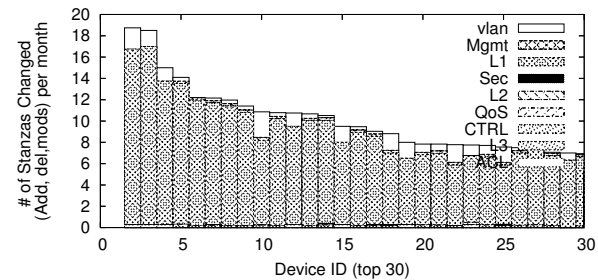


(d) Change characteristic over five years - *UW*

Figure 11: Static analysis and change analysis for switches. (a) and (b) shows a static analysis of the latest configurations all combined for each campus. (c) and (d) shows the number of additions, deletions and modifications of each functionalities in a row-stacked fashion, all devices combined over 5 years.



(a) *GT*



(b) *UW*

Figure 12: Number of changes to each switch per month for the *GT* and *UW* networks.

nation *sec* and *qos*, this combination can be attributed to the usage of VPN devices, which are unique to *GT*. The *l1*, *l2*, and *mgt* combinations are unique to the *UW* campus; similar commands change as with the *l1* and *mgt* combination.

On the whole, we find significant evidence of correlated changes to stanzas, although the specific sets of stanzas that change depend on the network in question. As such, when using correlations to drive configuration management (*e.g.*, in providing guidance on configuration changes to install, as discussed in Section 5), we must first design techniques that learn the correlations prevalent in a given network.

5. DISCUSSION

We now discuss how our observations can help augment existing and proposed configuration tools and mechanisms.

First, it may be possible to provide recommendations or suggestions to the operator concerning changes, based on the previous

| UW | | GT | |
|--------------------|-----|--------------------|-----|
| Correlated Stanzas | % | Correlated Stanzas | % |
| <i>acl, l1</i> | 24% | <i>mgt, l1</i> | 15% |
| <i>l1, vlan</i> | 11% | <i>l1, vlan</i> | 13% |
| <i>l1, l2, mgt</i> | 11% | <i>acl, l1</i> | 11% |
| <i>mgt, l1</i> | 10% | <i>mgt, vlan</i> | 6% |
| <i>vlan, mgt</i> | 9% | <i>sec, qos</i> | 6% |

Table 7: Correlated stanza changes for switches.

historical change logs. For example, we have found that *l1* and *l3* stanzas change the most in routers, while *l1* and *vlan* stanzas changes together frequently in switches. Based on the what the operator is changing, a system could provide feedback or possible changes associated with currently updated stanzas. For example, in firewalls, if there is a change in network object group defini-

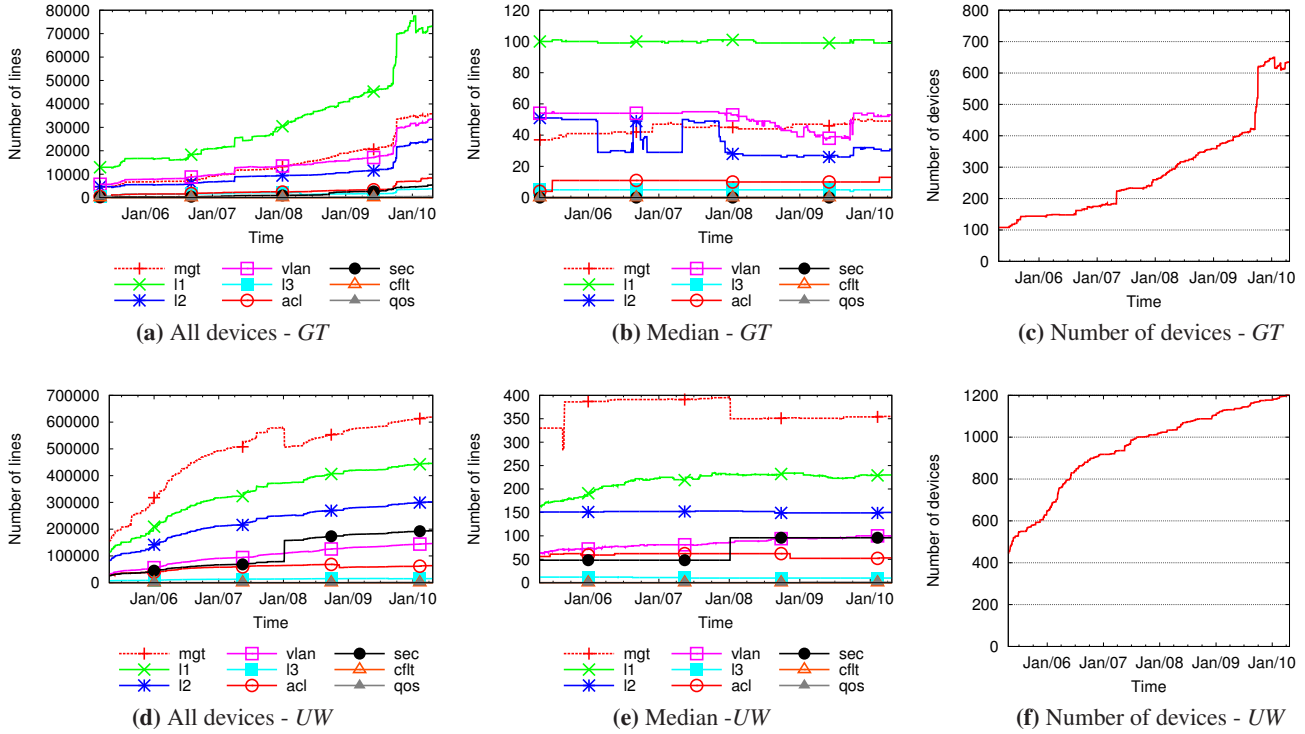


Figure 13: Longitudinal analysis of switches over five years.

tions, the system can recommend the operator to also consider the ACLs where that group is used. A tool could also proactively offer *negative* recommendations. If the operator attempts to change a line or stanza of the network configuration that is not frequently changed, the configuration system could discourage the modification by raising an alert, forcing the operator to go through multiple confirmation steps, or blocking a low-rank operator from making changes.

Knowledge of network configuration changes also helps point designers of configuration automation tools to parts of the configuration that could be automatically generated, for various types of devices. Our results reveal that network operators make different types of configuration changes, depending on the network device type and, hence, point to possibilities for customizing these configuration change templates based on the type of device. Highly specialized devices like firewalls are distinct, where most configuration is dedicated to access control, and the changes mostly happen on that specific area as well. For switches, a template that enables batch-modification on ports and VLAN configurations would be helpful.

Our work also opens many possible avenues for studying configuration changes in more detail. One possibility would be to look for changes to network configuration that were later reverted or changed again. Such changes might point to cases where operators made errors when configuring the network; knowledge about where in the configuration these errors occur could improve configuration testing methods. Another possibility would be to analyze the configuration for “copy-paste” behavior, as is done in software engineering studies of “code clones”. Identifying parts of configuration that are commonly copied might also inspire design templates for configuration update.

6. CONCLUSION

We presented a longitudinal analysis of network configuration and its evolution over five years for two large campus networks. We focused on capturing configuration characteristics and common practices by examining change patterns from CVS commit log data, in contrast to earlier studies, which have mostly examined a single configuration snapshot. In contrast to earlier studies, which have focused mostly on router configuration, we also study configuration changes for firewalls and switches, which comprise a significant portion of the networks. Our study unearths a variety of important aspects regarding configuration evolution: we study the frequency and extent of configuration updates, showing the similarities and differences across networks; identify the prevalence of correlated changes; and shed light on how various network-wide factors affect evolution.

Our work is a first step toward understanding configuration evolution in more depth. Just as mining software repositories has emerged into a mature field in software engineering for assisting programmers understand software engineering better, we believe that mining network repositories for configuration changes can help deepen our understanding of configuration and design practices. A better understanding of this process is imperative, as it is a cornerstone for improving current methods of configuring and managing network devices, easing the pain of network operators by providing recommendations, offering better templates, reducing misconfiguration and, ultimately, enabling automation.

Acknowledgements

We thank the campus network operators who provided us with network configuration data. We especially thank Dan Forsyth and Dale Carder for answering many questions about network management and configuration practices. We should also like to thank Lixia Zhang (our shepherd) and the anonymous reviewers for their insightful feedback. This work is supported in part by NSF grants CNS-0643974, CNS-0746531, CNS-1017545, CNS-1018021, and CNS-1050170.

REFERENCES

- [1] T. Benson, A. Akella, and D. Maltz. Unraveling Complexity in Network Management. In *Proc. 6th USENIX NSDI*, Boston, MA, Apr. 2009.
- [2] D. Caldwell, A. Gilbert, J. Gottlieb, A. Greenberg, G. Hjálmtýsson, and J. Rexford. The cutting edge of IP router configuration. In *Hotnets-II*, Cambridge, MA, Nov. 2003.
- [3] X. Chen, Z. Mao, and J. Van der Merwe. Towards automated network management: network operations using dynamic views. In *INM 07*.
- [4] S. Eick, T. Graves, A. Karr, J. Marron, and A. Mockus. Does code decay? assessing the evidence from change management data. *IEEE Transactions on Software Engineering*, 27(1):1–12, 2001.
- [5] W. Enck, T. Moyer, P. McDaniel, S. Sen, P. Sebos, S. Spoerel, A. Greenberg, Y. Sung, S. Rao, and W. Aiello. Configuration management at massive scale: system design and experience. *IEEE Journal on Selected Areas in Communications*, 27(3):323–335, 2009.
- [6] N. Feamster and H. Balakrishnan. Detecting BGP Configuration Faults with Static Analysis. In *Proc. 2nd USENIX NSDI*, Boston, MA, May 2005.
- [7] A. Feldmann and J. Rexford. IP network configuration for intradomain traffic engineering. *IEEE Network*, Sept. 2001.
- [8] J. Gottlieb, A. Greenberg, J. Rexford, and J. Wang. Automated Provisioning of BGP Customers. *IEEE Network*, 2003.
- [9] H. Kagdi, M. Collard, and J. Maletic. A survey and taxonomy of approaches for mining software repositories in the context of software evolution. *Journal of Software Maintenance and Evolution: Research and Practice*, 19(2):77–131, 2007.
- [10] S. D. Krothapalli, X. Sun, Y.-W. E. Sung, S. A. Yeo, and S. G. Rao. A toolkit for automating and visualizing vlan configuration. *SafeConfig '09*, pages 63–70, New York, NY, USA, 2009. ACM.
- [11] F. Le, S. Lee, T. Wong, H. Kim, and D. Newcomb. Minerals: using data mining to detect router misconfigurations. In *Proc. MineNets '06*, pages 293–298, Pisa, Italy, Sept. 2006.
- [12] F. Le, G. Xie, D. Pei, J. Wang, and H. Zhang. Shedding Light on the Glue Logic of the Internet Routing Architecture. Aug. 2008.
- [13] M. Lehman. Programs, life cycles, and laws of software evolution. *Proceedings of the IEEE*, 68(9):1060–1076, 1980.
- [14] R. Mahajan, D. Wetherall, and T. Anderson. Understanding BGP misconfiguration. In *Proc. ACM SIGCOMM*, pages 3–17, Pittsburgh, PA, Aug. 2002.
- [15] D. Maltz, G. Xie, J. Zhan, H. Zhang, G. Hjálmtýsson, and A. Greenberg. Routing Design in Operational Networks: A Look from the Inside. In *Proc. ACM SIGCOMM*, Portland, OR, Aug. 2004.
- [16] S. Raghavan, R. Rohana, D. Leon, A. Podgurski, and V. Augustine. Dex: A semantic-graph differencing tool for studying changes in large code bases. 2004.
- [17] Really Awesome New Cisco Config Differ (RANCID). <http://www.shrubbery.net/rancid/>, 2004.
- [18] Y. Sung, S. Rao, S. Sen, and S. Leggett. Extracting Network-Wide Correlated Changes from Longitudinal Configuration Data. In *Proc. PAM*, Seoul, South Korea, Apr. 2009.
- [19] G. Xie, J. Zhan, D. Maltz, H. Zhang, A. Greenberg, G. Hjálmtýsson, and J. Rexford. On static reachability analysis of IP networks. In *IEEE INFOCOM*, volume 3, pages 2170–2183, 2005.