Demystifying the Dark Side of the Middle: A Field Study of Middlebox Failures in Datacenters

Rahul Potharaju
Purdue University
rpothara@purdue.edu

Navendu Jain
Microsoft Research
navendu@microsoft.com

ABSTRACT

Network appliances or *middleboxes* such as firewalls, intrusion detection and prevention systems (IDPS), load balancers, and VPNs form an integral part of datacenters and enterprise networks. Realizing their importance and shortcomings, the research community has proposed software implementations, policy-aware switching, consolidation appliances, moving middlebox processing to VMs, end hosts, and even offloading it to the cloud. While such efforts can use middlebox failure characteristics to improve their reliability, management, and cost-effectiveness, little has been reported on these failures in the field.

In this paper, we make one of the first attempts to perform a large-scale empirical study of middlebox failures over two years in a service provider network comprising thousands of middleboxes across tens of datacenters. We find that middlebox failures are prevalent and they can significantly impact hosted services. Several of our findings differ in key aspects from commonly held views: (1) Most failures are grey dominated by connectivity errors and link flaps that exhibit intermittent connectivity, (2) Hardware faults and overload problems are present but they are not in majority, (3) Middleboxes experience a variety of misconfigurations such as incorrect rules, VLAN misallocation and mismatched keys, and (4) Middlebox failover is ineffective in about 33% of the cases for load balancers and firewalls due to configuration bugs, faulty failovers and software version mismatch. Finally, we analyze current middlebox proposals based on our study and discuss directions for future research.

Categories and Subject Descriptors

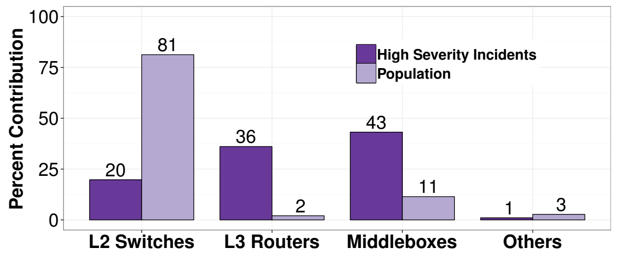
C.2.3 [**Computer-Communication Network**]: Network Operations—*network management*

General Terms

Network management; Reliability

Keywords

Datacenters; Network reliability; Middleboxes



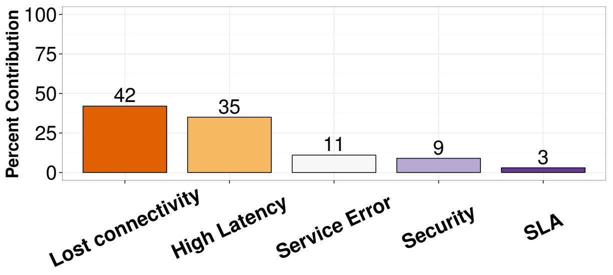


Figure 1: Middleboxes contribute to 43% of high-severity incidents despite being 11% of the population (top). The top-5 categories of service impact in these incidents caused by middleboxes (bottom).

1. INTRODUCTION

Today's datacenters and enterprises deploy a variety of intermediary network devices or *middleboxes* to distribute load (e.g., load balancers), enable remote connectivity (e.g., VPNs), improve performance (e.g., proxies) and security (e.g., firewalls, IDPS), as well as to support new traffic classes and applications [1–4]. Given these valuable benefits, the market for middleboxes is estimated to exceed $10B by 2016 [5], and their number is becoming comparable to that of routers in enterprise networks [1, 6].

These benefits, however, come at a high cost: middleboxes constitute a significant fraction of the network capital and operational expenses [1]; they are complex to manage and expensive to troubleshoot [6]; and their outages can greatly impact service performance and availability [7]. For instance, in December 2012, a load balancing misconfiguration affected multiple Google services including Chrome and Gmail [8]. In a 2011 survey of 1,000 organizations [9], 36% and 42% of the respondents indicated failure of a firewall due to DDoS attacks at the application layer and network layer, respectively; the very attack firewalls are deployed to protect against.

Table 1: The major findings of our study of middlebox failures and their implications.

Major findings	Implications
(1) Middlebox failures are prominent and can greatly degrade service performance and availability.	(1) The problem of middlebox failures is significant and worth special attention from the research community.
(2) Most failures are grey dominated by connectivity errors and link flaps that exhibit intermittent connectivity; fail-stop device failures occur, but they are not as often.	(2) Connectivity errors degrade service performance and availability e.g., due to timeouts, delayed protocol convergence, packet retransmissions, and parity errors. Robust network protocols (e.g., MP-TCP [10]) need to be developed to route traffic around interfaces exhibiting intermittent connectivity.
(3) Hardware faults, misconfigurations and overload problems are present, but they are not in majority in contrast to common perception [6].	(3) Hardware repairs risk long periods during which the network is operating at reduced or no redundancy. Load testing of middleboxes is needed to measure the maximum operating capacity under peak traffic.
(4) There are a variety of misconfigurations such as incorrect rules, VLAN misallocation and mismatched keys.	(4) Software-defined networks (SDNs) based configuration and management tools need to be developed to handle a wide range of middlebox misconfigurations.
(5) Middlebox redundancy is ineffective in about 33% cases for load balancers and firewalls due to configuration bugs, faulty failovers and software version mismatch.	(5) Middlebox failovers need rigorous testing and ensuring software version compatibility in redundant units. Explore commoditizing middleboxes to dynamically scale-out/in and to avoid single points of failure.
(6) A family of middleboxes having faulty PSUs and another exhibiting the few bad apples effect, could have been detected as early problem indicators.	(6) Trend analysis can be a useful tool in identifying failure trends in network datasets e.g., compare device categories/models to find the ones less reliable than others.

Our analysis, Figure 1 (top) of *high-severity* incidents (signifying high customer or business impact), over four years in the network we studied indicates that middleboxes contributed to about 43% of the incidents despite being 11% of the population. Figure 1 (bottom) shows the top-5 categories of service impact caused by middleboxes in this dataset. Operators tagged each incident to a problem category based on the primary issue experienced by impacted customers. We observe that loss of connectivity and high latency issues dominate, with a long tail of other issues such as service errors, security problems and SLA violations. Thus, understanding the characteristics and the impact of middlebox failures is important to improve performance and availability of hosted services.

Recent research proposals on middleboxes [2, 3, 6, 11–13] aim to alleviate many of their shortcomings, but they do not address how to improve middlebox reliability in existing deployments. These efforts, in turn, would benefit from a field study on middlebox failures as understanding their failures, causes and impact may help guide researchers to improve their reliability, cost-effectiveness and management, in spirit of recent efforts [14, 15].

Unfortunately, very few empirical studies have been conducted on middlebox failures and even they have a limited focus. For example, Allman et al. [16] perform active measurements of TCP traffic across two firewalls and load balancers to quantify their impact on end-to-end performance [16]. Sekar et al. [1, 11] presented anecdotal data on middleboxes from a large enterprise network. Sherry et al. [6] provide a useful survey of 57 network administrators to study enterprise middlebox deployments. They, however, do not report any empirical failure data, but rather the fraction of administrators who *estimated* one of three pre-defined issues to be the most common root cause. Our prior work on datacenter network failures observed that load balancers exhibit many short-term, transient faults [17]. However, it did not analyze their root causes, service impact, effectiveness of middlebox redundancy or study other types of middleboxes; we provide a detailed comparison to related work in Section 8.

1.1 Our contributions

In this paper, we present a large-scale empirical study of middleboxes in datacenters to (a) understand their failure characteristics, (b) analyze current middlebox proposals through the lens of reliability, and (c) derive implications for research on middleboxes. Specifically, our study focuses on answering the following key questions:

1. What are the failure characteristics of middleboxes?
2. What are the root causes and what actions are taken for resolution? How prevalent are misconfigurations and what are their different types?
3. Does reliability improve with new generations? Can we find *early problem symptoms* to enable proactive repair?
4. Is middlebox redundancy effective in masking failures?

To answer these questions, we collect network event logs over two years (July 24, 2010-12) across 10+ datacenters hosting 100k+ servers and 2k+ middleboxes; absolute counts omitted due to confidentiality reasons. Our measurement data covers a wide range of network event sources, including syslog and SNMP alerts, configuration files, a maintenance tracking system, trouble tickets, and traffic data. The details of our datasets and methodology are described in Section 2. Our major findings are summarized in Table 1.

To identify failure trends in network datasets, we propose using a simple technique of trend analysis which can uncover global problem trends rather than analyzing incidents in isolation. For instance, we observed that only a few devices contributed to a significant number of failures in one device family exhibiting the *few bad apples* effect, and a family of load balancers having faulty PSUs could have been detected as early problem indicators by aggregating failure root causes across that type, but they kept being repaired in isolation as observed in our failure dataset.

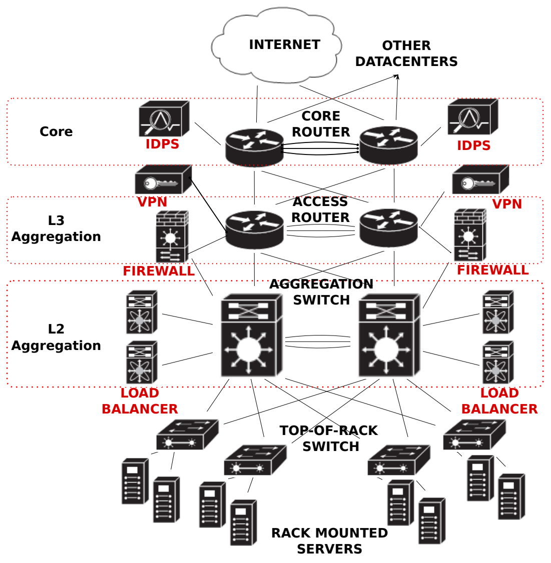


Figure 2: Example of a conventional datacenter network comprising middleboxes, switches and routers.

While we studied only our dataset, we believe it is fairly representative of a larger class of datacenters based on the scale (2k+ middleboxes across 10+ datacenters), diversity (multiple middlebox types and platforms), measurement duration (over two years), and validation with operators having high experience with middleboxes and some having previously worked at other sites. Further, the middleboxes we studied are from multiple leading network vendors having a significant share of the middlebox market.

2. BACKGROUND AND METHODOLOGY

In this section we provide a brief overview of the datacenter network architecture, and then describe the challenges and methodology for analyzing middlebox failures.

2.1 Datacenter network architecture

Figure 2 illustrates a topology[1] of a conventional datacenter network [18] comprising a variety of L2 switches, L3 routers and middleboxes such as firewalls, IDPS, load balancers, and VPNs.

Firewalls (FW) protect applications from unwanted traffic (e.g., DoS attacks) by examining packet fields at IP, transport and application layers against a specified set of rules. Network-based IDPS devices analyze traffic to safeguard against attacks such as malware and intrusions. Typically, hardware FWs and IDPSes are deployed to handle high traffic rates [19]. Load Balancers (LB) distribute application requests across hosted servers. Redundant pairs of LBs typically connect to each aggregation switch (AGG) and perform mapping between static (exposed to clients through DNS) and dynamic IP addresses of the servers. Recent hardware LBs also provide other features such as NAT, SSL acceleration, and data caching. VPNs are typically used to facilitate secure remote access for web and client/server applications. Some commercial offerings such as Cisco Catalyst 6509-E provide VPN functionality through IPSec and SSL. Due to the low population of other middleboxes such as NATs and media converters, our study did not include them.

[1]Other configurations are possible e.g., firewalls at the network edge.

2.2 Network Datasets

We examined multiple sources of network failure data for middleboxes over two years logged in the monitoring servers of a large service provider comprising 100k+ servers and 2k+ middleboxes across 10+ datacenters. The service provider uses middleboxes to deliver a variety of applications e.g., load balancers for web services, search, email, cloud computing and video streaming; firewalls and IDPS to protect confidential data (e.g., high business impact applications), and VPNs to enable remote access (e.g., to office network).

- **Event Logs**: Operators typically detect network failures from monitoring alarms such as syslog and SNMP traps, and track device/link status via ping and SNMP polling. These logs contain information about what type of network element experienced the event, event type, the other end-point of the interface, and a short machine-generated description.
- **Configuration data**: A configuration database stores and tracks changes to device configurations using a revision control system. It also provides a 'diff'-like functionality.
- **Maintenance Tracking System**: This system tracks network changes such as device provisioning, repairs, and code updates. Network engineers use this system to check for on-going and planned maintenance during failure triage.
- **Link-level traffic**: Traffic carried on each network interface is logged using SNMP [20] polling which averages packet and byte counts observed every five minutes. Most traffic monitoring systems use the MIB [21] format to store the data that includes fields such as the interface type (e.g., token ring, ethernet), the other endpoint, interface status (up/down), and the number of bytes sent/received. We correlate this data with network events to extract failures impacting network traffic, and to reverse-engineer the topology.
- **Trouble Tickets**: Network tickets record the activities of engineers while troubleshooting a problem [22]. Each ticket is assigned a unique identifier (TicketID) and contains multiple fields such as when and how a failure was discovered, and a diary of free-form text written by engineers describing the steps taken to mitigate the problem.

Note that since the datacenters we studied are managed by a central operations team, their failure characteristics are similar and thus we do not compare between datacenters.

2.3 Methodology

Failure definition. Intuitively, a failure can be defined as an event that causes a network device or link to be *unavailable* to carry traffic. However, precisely defining a middlebox failure is complicated because its impact is dependent on the device type. For instance, a faulty load balancer may cause a loss of throughput while a misconfigured firewall may incorrectly forward or drop legitimate traffic. Therefore, we consider all logged events as failures which cause a traffic impact (in terms of loss in traffic volume) or cause a device to function incorrectly in routing or processing traffic (based on problems observed in trouble tickets).

Challenges in analyzing data. There are several challenges in utilizing the network event logs to derive an accurate set of failure events and to characterize their impact:

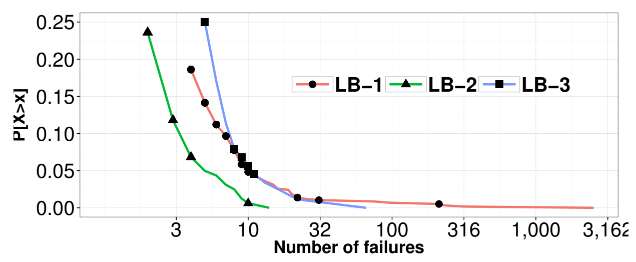


Figure 3: CCDF plot showing that a small fraction of devices log ≫ 100 failure events.

1. How can we differentiate noisy events from *meaningful* failures? For instance, syslog messages can be spurious where a device may log multiple 'down' events (even when it is operational) or when neighbors may send redundant notifications for the same device. Similarly, how to handle link flaps generating multiple and possibly overlapping down/up messages which get logged as separate events?

2. Is it accurate to consider all failures to characterize reliability? In particular, some down events are expected due to routine and scheduled maintenance e.g., code update. Operators aim to limit the planned downtime, and at the same time, prioritize detection and mitigation of unexpected failures. While prior efforts [16, 17] did not differentiate them, "lumping" all events together risks inflating failure rates.

3. How to handle *redundant failures* from devices that keep logging errors while undergoing repairs or being scheduled for replacement? Figure 3 shows the CCDF for different types of LBs in our dataset with some devices logging more than 1000 down messages over a few hours as the monitoring system did not suppress them during triage. Such events may bias measurement of device failures.

Event Filters: To address these challenges, we apply a pipeline of four event filters (Figure 4): (1) timing filter, (2) maintenance filter, (3) redundant failures filter, and (4) impact filter.

1. **Timing Filter**: This filter fixes various timing inconsistencies. First, it groups all events with the same start and end time originating on the same interface to remove duplicate events. Second, to avoid any problems due to clock drifts (typically small) and log buffering, multiple events within a 60 second window on the same interface are grouped into a single event by picking the earliest start and end times. If two events on the same interface have the same start time but different end times, we group them into a single event and assign the earliest end time as events may not always be quickly cleared after being fixed.

2. **Maintenance Filter**: We use the maintenance tracking system to filter failure events due to planned maintenance. As operators likely have a good understanding of scheduled repairs, this filtering enables computing device reliability only due to unexpected or unplanned problems.

3. **Redundant Failures Filter**: To filter redundant failures, we group events based on their unique trouble ticket identifiers as events within the same ticket are typically part of the common problem. We validated these events with operators assigned to trouble tickets for these failures.

4. **Impact Filter**: We measure the impact of a failure in two ways based on our definition of a middlebox failure: (a) traffic loss and (b) device malfunctioning.

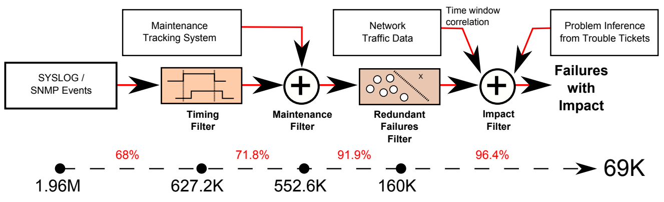


Figure 4: A pipeline of event filters applied to extract meaningful failures from the network data sources. The cumulative noise reduction at each step is shown below.

A. Traffic Impact: Since the network is shared and we did not have access to application logs, we estimate the failure impact by leveraging network traffic data and computing the ratio of the median traffic on a failed device/link during a failure and its value in the recent past (preceding a two hour window): a failure has traffic impact if this ratio is less than one [17]. Note that this filter is applied *after* a network alarm has been logged and a ticket opened to resolve the issue. Hence, it will not misclassify failures due to the inherent time-varying nature of traffic e.g., time of day effects. We perform hypothesis testing to validate this approach. Specifically, we use the Mann-Whitney-Wilcoxon (MWW) [23] test for validation, which is a non-parametric significance test to compare two populations. The *null hypothesis* to test is that the traffic values observed before a failure and during a failure have identical distribution functions. We randomly sampled 7k middlebox failure events and 7k time points from periods during which no known failures occurred. For the former, we obtained traffic values up to eight hours before the failure and during the failure, and up to eight hours before the event and up to two hours afterwards for the latter.

Figure 5 plots the distribution of the p-value to quantify the fraction of cases where we can reject the null hypothesis at a significance level of 0.05. The top figure shows that in 99.3% of the failures, p-value is less than 0.05; the remaining cases had insufficient traffic data points (e.g., due to best-effort logging). Hence the null hypothesis can be rejected in 99.3% of the cases. Similarly, the bottom figure shows that the distributions are identical during periods when no failures occurred. Further, a time window size of two hours and eight hours of traffic before the event yields the highest and lowest accuracy, respectively. Indeed, under more strict assumptions, a significant MWW test can be interpreted as showing a difference in medians [24]. Our prior work [17] also correlated failures with traffic deviations, but it did not evaluate its accuracy or specify the time window size before a failure to compute the traffic ratio with highest accuracy. As Figure 5 (top) shows, the accuracy is clearly dependent on the time window size.

B. Device Malfunction Impact: To measure impact due to incorrect device function, we leverage the information logged by operators in trouble tickets to determine the problem root causes when middleboxes fail. Specifically, we apply NetSieve [25], an automated problem inference system that analyzes the free-form text in a trouble ticket to generate its synopsis: (1) the problems observed e.g., link down, misconfigured rule, (2) troubleshooting performed e.g., check configuration, verify routes, and (3) actions taken for resolution e.g., replaced line card, cleaned fiber.

Table 2: Annual number of failures per middlebox and annualized failure rate (AFR) for devices that were present for at least a month in production.

Type	Mean	Stddev	25P	50P	75P	95P	COV	AFR% (July 2010-11)	AFR% (July 2011-12)
FW	2.1	2	1	2	3	5	0.9	19	13
IDPS	3.5	2.9	1	1	6	8.3	0.8	23	18
LB	1.5	0.9	1	1	2	3	0.6	31	19
VPN	3.4	2.2	2	3	5	7	0.7	7	12

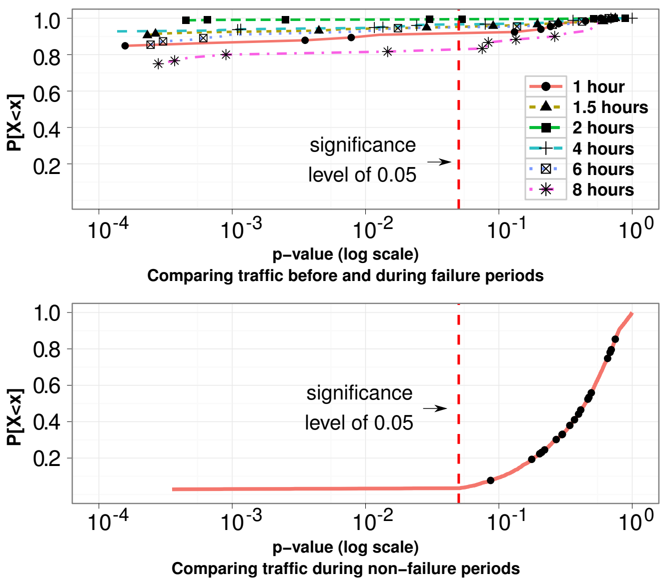


Figure 5: MWW test: p-value < 0.05 in 99.3% of the failures (top), and p-value > 0.05 in 99.2% of the events during no failure periods (bottom). Hence the null hypothesis can be rejected.

Our dataset comprised 1.96M raw events collected over two years for FWs, IDPSes, LBs and VPNs. Note that the high-severity incidents described in Figure 1 is a very small fraction compared to this dataset; we use the latter for analysis throughout the rest of this paper. Applying the pipeline of filters on this data resulted in about 17k events from the traffic filter and about 52k events from the device malfunction filter. To ensure 100% coverage of impactful failure events, we verified that our filtered dataset includes all the events recorded in tickets deemed 'actionable' by operators. Since typically multiple events logged close in time for a device attribute to a common problem and hence are likely to be merged in the same ticket, the number of tickets is significantly smaller than the number of events.

3. FAILURE CHARACTERIZATION

In this section we characterize reliability of the four types of middleboxes in our dataset: firewalls, IDPS, VPNs each spanning two generations (FW[1-2], IDPS[1-2], and VPN[1-2], respectively), and load balancers spanning three of them (LB[1-3]). For a middlebox type, its different generations are from the same network vendor, and they are ordered by their release date (1 is oldest). Where applicable, we first provide an inter-type comparison and then compare devices across different generations within a type.

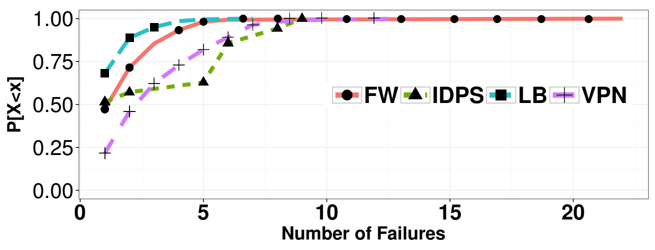


Figure 6: Failures per device per year for middleboxes.

3.1 How reliable are middleboxes?

Figure 6 shows the CDF of the number of failures per device per year for devices with at least a month in operation. Most devices experience few failures per year with a median value of at most three (Table 2), and they occur with low variability (COV < 1: COV [26] is defined as the ratio of standard deviation to the sample mean where a COV $>> 1$ indicates high variability) as expected in a service provider network carrying customer traffic. Across middlebox types, IDPS devices show relatively highest annual failures per device with a 95th percentile value of 8.3 while LBs exhibit lowest with a 95th percentile value of 3. The distribution for firewalls have a relatively long tail of up to 24 annual failures per device compared to other types.

We next compute the annualized failure rate (AFR) by dividing the number of devices of a given type that observe at least one failure by the population of that type.

Inter-type: Table 2 shows the AFR for different device types during July 2010-11 and July 2011-12. We observe that LBs are the least reliable with about a 1 in 3 chance of failure during July 2010-11 which improved to about a 1 in 5 chance in the next year. The reason for this improvement is due to on-boarding of the more reliable LB-3 platform in the deployment and the end-of-life retirement of some of the older LB-1 devices. FWs and IDPSes also exhibit improvement in AFR over the two years. VPNs show relatively the lowest AFR amongst all middleboxes.

Intra-type: Figure 7 shows the AFR and Figure 8 shows the fraction of failures and downtime, respectively, across different generations of each middlebox type. We find that during the first year, FW-2 exhibits relatively low failure rate of 14% whereas FW-1 exhibits 2x higher failure rate at 32%. Through trouble tickets, we found that the reliability of FW-2 started improving when the vendor released an end-of-life notice for FW-1 and provided faster bug fixes and better technical support to handle FW-2 failures. Figure 8 shows that FW-2 contributed majority to the total failures due to its dominant population while FW-1 devices decreased due to being decommissioned. This observation

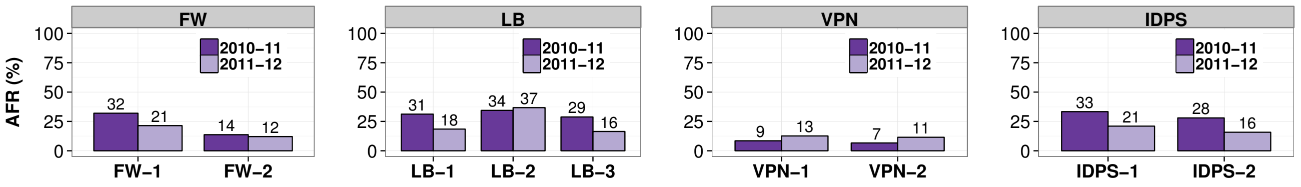


Figure 7: Comparing annualized failure rate across middleboxes over the two years measurement period.

combined with the relatively low AFR of FW-2 (Figure 7) indicates that a few devices contributed the most, due to limited device familiarity of operators with the new platform and unexpected hardware problems. Further, the number of failures contributed by FW-2 is roughly proportional to its downtime while they are relatively higher for FW-1. This indicates that most FW-1 failures seem to be relatively short-lived and resolved through reboot or rule reconfiguration.

In the case of load balancers, LB-1 exhibits a large AFR and contributes significantly (Figure 8) toward the total number of failures and downtime. However, note that in comparison to other LBs, the fraction of failures contributed is higher than the downtime indicating that most problems are short-lived. We validate this observation by using the time-to-repair (TTR) plots (Figure 9(c)) where a short TTR indicates transient problems. We find that a majority of these problems were due to link flapping i.e., interface continually goes up and down due to a bad cable, faulty line card, etc. In most cases, the network engineer adjusts/reseats the cable/network card or the problem gets self-corrected. LB-2, on the other hand, exhibited an increased AFR indicating that this generation experienced new types of problems such as unexpected reboots. However, these problems seem to have been resolved in its successor, LB-3, which exhibits relatively the highest reliability. Similar to the case of firewalls, note that the significant improvement in LB-3 reliability occurred due to the vendor releasing an end-of-life notice for older generations, as we observed through tickets.

Across the four middlebox types, VPN devices exhibit relatively higher reliability with failure rates between 7%-9% which increased slightly year over year to 11%-13%. The increase in AFR over time is due to (a) the older generation VPN-1 which was being decommissioned exhibiting an increasing number of connectivity errors, and (b) the newer generation VPN-2 exhibiting the infant mortality effect e.g., memory errors. Further, the failures contributed is roughly proportional to the downtime caused by both VPN-1 and VPN-2 suggesting that each failure roughly causes the same downtime on average. We find a common root cause of VPN failures to be VLAN misconfigurations. Similar to other middleboxes, we also observed an end-of-life announcement for VPN-1 recently.

Findings (1): (1) Firewalls and IDPS devices exhibit annualized failure rate of 19% and 23%, respectively, which improved to about 13% and 18% year over year due to timely bug fixes and decommissioning of older generation of devices. (2) About 1 in 3 load balancers experience a failure and exhibit the highest annualized failure rate during July 2010-11, which improved next year to about a 1 in 5 chance of failure due to new generations exhibiting relatively higher reliability. (3) VPN devices exhibit the relatively lowest annualized failure rate across middleboxes. (4) When vendors release an end-of-life notice for an old device generation, it often results in improved reliability of newer generations because of faster bug fixes and better technical support.

3.2 How often do middleboxes fail?

We define the time to failure (TTF) of a device as the time elapsed between two consecutive failures. As this metric requires that a device fail at least twice, we exclude devices having a single failure during our measurement period.

Inter-type: Figure 9(a) shows the TTF across middleboxes. We observe that the lines of FWs and LBs nearly overlap exhibiting a high similarity with a median TTF of 7.5 hours and 5.2 hours, respectively. Many FW failures are caused due to connectivity errors, hardware faults and overload conditions (Figure 11). Similarly, LB failures are mostly due to short-term transient faults, hardware and parity errors e.g., due to incompatible protocol versions between devices. For IDPS, the median is about 20 minutes while the 95th percentile is around 40 days. This indicates that even amongst devices that fail multiple times, there are two types: (a) robust ones that fail at most once in one or two months and (b) failure prone ones that experience numerous failures (the few bad apples effect), mostly within a few hours. In comparison, VPN devices show relatively the smallest median TTF of about 10 minutes due to connectivity errors (Figure 11) and 95th percentile of about three days.

In terms of availability, Figure 9(b) shows that FWs exhibit a median of four 9's and a 95th percentile of two 9's. VPNs perform slightly better with their 95th percentile above two 9's. While the median value for LBs is above four 9's, their 95th percentile is two 9's due to all three LB[1-3] generations exhibiting high AFR over the two year measurement period as observed in Figure 7.

3.3 How long do repairs take?

We define the time to repair (TTR) for a device as the time between a down notification for a device and when it is next reported as being back online. There are two main types of failures: *short-lived transient faults* where an engineer may not always intervene to resolve them and *long-term failures* where the device or its component is usually replaced or decommissioned. Note that for long-term failures, the failure durations may be skewed (up to days) by when the trouble ticket got closed e.g., till a spare is on-site. Figure 9(c) shows the TTR results. The 95th percentile values show that the distribution has a long tail for both LBs and FWs. Using problems inferred from trouble tickets, we found several incidents for device replacements and ambiguous customer complaints which resulted in an increased TTR. For instance, in several cases, after receiving vague complaints from many customers about occasional slow file-transfer rates between two datacenters, determining the rootcause was challenging as the problem could not be easily reproduced. The root-

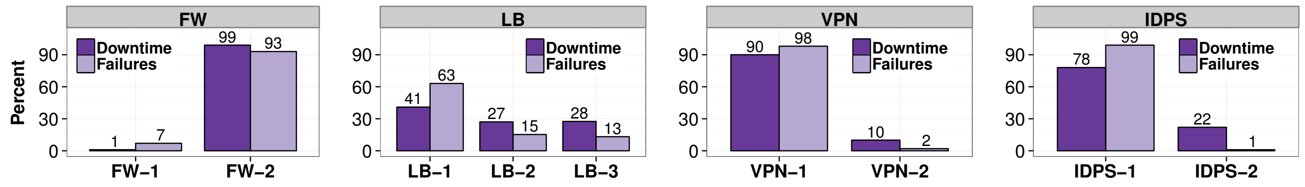
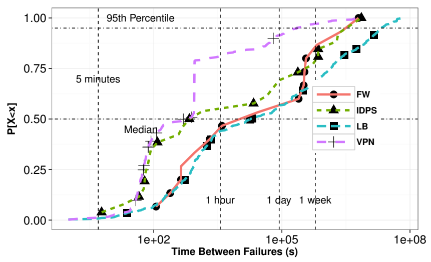
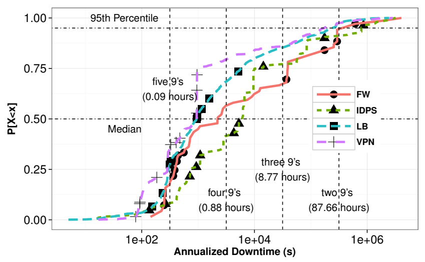


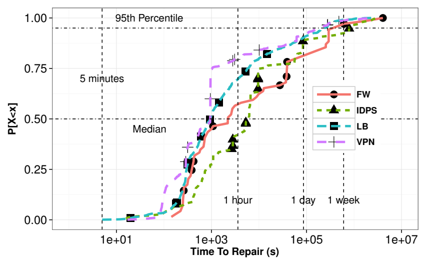
Figure 8: Comparing fraction of failures and downtime across middleboxes.



(a)



(b)



(c)

Figure 9: Comparing (a) Time to Failure, (b) Annualized downtime, and (c) Time to Repair across middleboxes.

cause, determined after three weeks, was a software bug that prevented some FWs from recognizing large flows.

Across device types, middleboxes exhibit typically a low median TTR, indicating short-lived failures that get resolved within a few minutes to at most a few hours in most cases. The short-lived failures for LBs are due to interface flaps and software problems (e.g, OS watchdog timer triggering a reboot) and the long-lived failures to delays in equipment replacement (due to spare unavailability), faulty spares and problem misdiagnosis. VPNs exhibited mostly link flapping and connectivity errors which get resolved quickly (e.g., protocol convergence, port disabled) resulting in a small TTR.

Findings (2): Middleboxes exhibit a low median TTR, suggesting most failures are short-lived due to connectivity errors and link flaps that get resolved within a few minutes to a few hours.

3.4 How to model failures and repairs?

Building statistical models of middlebox failures is important as researchers can use them to understand their failure modes, analyze a large number of network design solutions [14, 16, 27], and derive implications to improve fault detection and troubleshooting operations. Although this section discusses the specific case of LBs, note that our analysis can be extended to other types of middleboxes.

Before modeling the failure distribution, we need to first characterize their statistical properties. As Figures 9(a) and 9(c) show, the range of TTF and TTR durations can be several orders of magnitude across devices. To reduce this high variance, we use the Box-Cox transformation [28], a family of parametric power transformation techniques that transform data such that it approximately follows the normal distribution. Using this technique, we found that the logarithmic power transform yielded the best normal fit for our failure data which we verified using a Q-Q plot (not shown).

Figures 10(a) and 10(d) show the kernel density plot of the log-transformed TTF and TTR respectively. We observe that the distributions are right skewed and are clearly not unimodal (observe the peaks). To find these peaks automatically, we use a local peak finding algorithm where a peak is defined as the locally highest value in a time series, before a change in signal direction has occurred. Observe that the main peaks occur at 15 minutes for TTF (Figure 10(a)) and 5.6 minutes for TTR (Figure 10(d)). Further, there are secondary peaks, which are relatively small, at 14 days for TTF and 24 hours and 2.5 days for TTR.

These findings indicate that there are two or three qualitatively different *types* of failures and repairs in effect, respectively. For TTF, the peak at 15 minutes indicates that connection errors and interface flaps dominate middlebox failures, and the secondary peak at 14 days indicates problems due to both hardware faults (e.g., line card, memory errors) and software bugs. For TTR, most repairs take a short time (< 6 minutes), but once the repair window exceeds a day (the 24 hours peak), it will likely take longer e.g., hardware replacement.

Failure Modeling: Based on prior work [14, 29], we first tried to fit existing heavy-tailed distributions to our failure data. However, they did not capture the multi-mode properties of our dataset. For instance, in the case of lognormal distribution, the Q-Q plots in Figures 10(b) and 10(e) show that the data does not follow the absolute line.

Intuitively, the presence of multiple peaks indicates that a two or three-component mixture of right-skewed distributions might be able to provide a good approximation of the process generating the data. We next present the mixture model framework [30] we used. Suppose the real-valued variables $\mathbf{X}_1, ..., \mathbf{X}_n$ are a random sample of time periods from a finite mixture of $m(> 1)$ arbitrary distributions, called *components*. The density of each $\mathbf{X}_i$ may then be written as:

$$h_\theta(\mathbf{x}_i) = \sum_{j=1}^{m} \lambda_j \phi_j(\mathbf{x}_i), \mathbf{x}_i \in \mathbf{R}^r \quad (1)$$

where $\boldsymbol{\theta} = (\boldsymbol{\lambda}, \boldsymbol{\phi}) = (\lambda_1, ...\lambda_m, \phi_1, ..., \phi_m)$ denotes the model parameter and $\sum_{j=1}^{m} \lambda_m = 1$. We further assume that ϕ_j

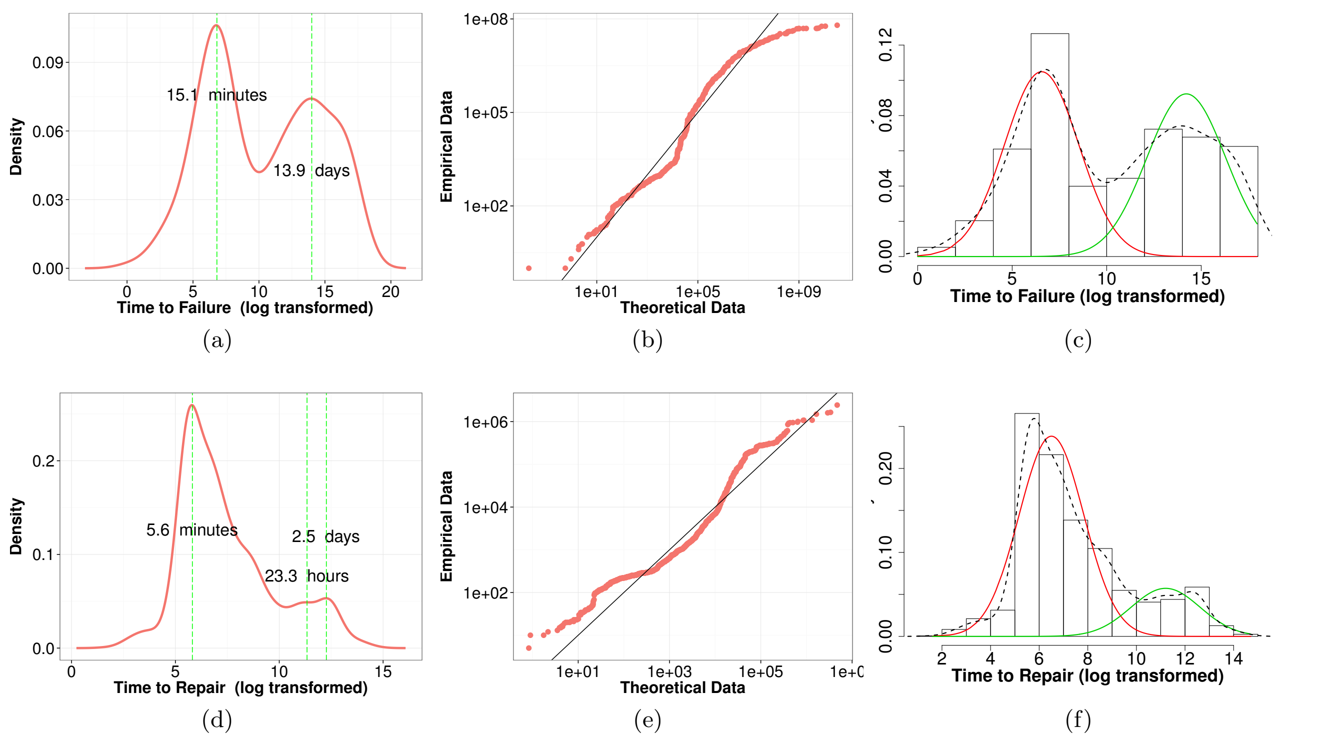


Figure 10: Modeling Time to Failure [a-c] and Time to Repair [d-f] for load balancers. Figures (a, d) show the kernel density plot of the log transformed TTF and TTR. Figures (b, e) show fit of a single log-normal distribution. Figures (c, f) show fit of a mixture of log-normal distributions.

are drawn from some family $\mathcal{F}$ of univariate density functions on $\mathbf{R}$. We consider a univariate lognormal family $\mathcal{F} = \{\phi(\cdot|\mu_{ln}, \sigma_{ln}^2\}$, in which μ_{ln} and σ_{ln} denote the mean and standard deviation in log scale. Thus, the model parameter reduces to $\boldsymbol{\theta} = (\boldsymbol{\lambda}, (\mu_{ln_1}, \sigma_{ln_1}^2), ..., (\mu_{ln_m}, \sigma_{ln_m}^2))$. By substituting these parameters, Equation (1) can be written as:

$$h_\theta(\mathbf{x}_i) = \sum_{j=1}^{m} \lambda_j \frac{1}{\sigma_{ln_j} \mathbf{x}_i \sqrt{2\pi}} e^{-\frac{(ln(\mathbf{x}_i) - \mu_{ln_j})^2}{2\sigma_{ln_j}^2}}, \mathbf{x}_i \in \mathbf{R}^r \qquad (2)$$

We obtained a reasonable fit by choosing a two-component ($m = 2$) lognormal mixture, where our model from Equation (1) becomes: $\lambda f(\mu_{ln_1}, \sigma_{ln_1}) + (1 - \lambda) f(\mu_{ln_2}, \sigma_{ln_2})$. To obtain the model parameter λ, we use the well-known Expectation Maximization(EM) [31] algorithm. Table 3 gives the values of the parameters for the two-component lognormal mixture distribution. We use the two-sample Kolmogorov-Smirnov (KS) test [32] as a goodness-of-fit test to compare data generated from the two-component lognormal mixture distribution with our failure data. We obtain a p-value of 0.111 indicating that we cannot reject the *null hypothesis* that the samples are drawn from the same distribution.

Figures 10(c) and 10(f) show how this model fits our data; the dotted-line is the kernel density curve of our data and the solid lines are the individual mixture components. We observe that our model closely approximates the real-world data of load balancer failures in our study.

4. ROOT CAUSE ANALYSIS

In this section, we analyze the root causes of middlebox problems and the actions taken to mitigate them.

Table 3: Parameters for the two-component lognormal mixture distribution

Type	λ	μ_{ln_1}	σ_{ln_1}	μ_{ln_2}	σ_{ln_2}
TTF	0.516284	6.557005	1.963986	14.210978	2.090600
TTR	0.428609	6.045474	0.690533	8.524983	2.541691

4.1 What are the dominant problems?

As described in Section 2.3, we apply NetSieve to do automated problem inference on network trouble tickets [25] for determining the problem root causes when middleboxes fail. Table 4 shows the classification of the problems into five categories based on the ticket data and their example causes. Note that there are two potential issues to be considered in interpreting these categories. First, a problem observed in one category can be due to issues in another one. For instance, parity errors in VPNs indicate failed sanity checks which may be caused due to misconfigured pre-shared keys. Another example is that a malformed message could be due to incompatible protocol versions between devices. Second, in some cases, there may be multiple problems pertaining to the same failure. In these cases, we take a conservative approach and attribute each observed problem to its respective category.

Inter-type: Figure 11 shows the breakdown of problems observed across the four types of middleboxes. We find that connectivity errors dominate failures across all middleboxes as also validated from the results in Section 3. Connectivity errors are mainly interface/link flaps which can be due to many factors such as protocol convergence, line card errors, cpu overload, bit corruption and cable problems. As a

Table 4: Classification and examples of problem types observed across middleboxes.

Problem Classification	Example causes or explanation
Connectivity errors	Device/neighbor unreachable, link flaps, ARP conflicts, address not found, message/frame parity errors, checksum failed, malformed message, port errors
Hardware	faulty PSU/ASIC, memory error, defective chassis/disk/fan, failed SUP engine/line card
Misconfiguration	bad rule, VLAN misallocation, configuration conflict/out-of-sync/corrupted/missing, mismatched crypto keys, expired security association, incorrect clock causing certificate expiry, incorrect root password
Software	reboot/reload, firmware bug, OS errors
Overload	High utilization exceeding a specified load threshold

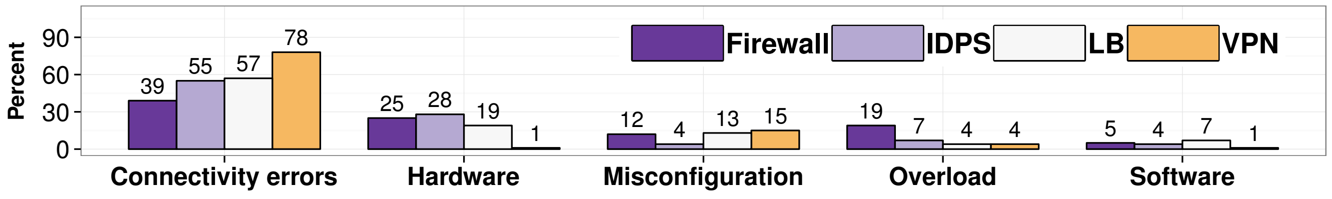


Figure 11: Problem types observed across middleboxes.

result, these errors can significantly degrade service performance and availability due to TCP timeouts, convergence delays, and retransmissions e.g., due to malformed packets or parity errors. The second main root cause is hardware problems except for VPNs where it accounts for only 1% of the problems. One implication of high percentage of hardware problems is long repair times as the faulty component or the device gets replaced. Another one is that this observation is in direct contrast to the industry view of equating hardware boxes with high reliability.

VPN problems are dominated by parity errors which may be attributed to faulty cables (e.g., copper, SFP fiber optics) and their common root causes are power surge, bad connectors, or a short circuit. For these problems, we observed mainly two repair actions of either cleaning the fiber or replacing them. In comparison, misconfigurations and overload problems are relatively in minority, contrary to the common view of being a dominant contributor [6].

Table 4 (row 3) gives the common categories of misconfigurations across device types. While bad rules as a common root cause (≈70% cases) are expected, there are a wide variety of other problems such as VLAN misallocation, corrupted or missing configuration files, mismatched cryptographic keys, and expired certificates.

Across middleboxes, misconfigurations contribute lowest to IDPS problems due to the fact that centralized management in a datacenter continuously monitors and immediately enforces policy changes (e.g., security access) upon violation. Software problems contribute the minimum percentage to problems indicating that the software running on these devices exhibits reasonable stability. Finally, software problems are fixed relatively quickly e.g., via code update or power cycling the device.

Intra-type: Figure 12 shows the problems observed split by different generations for each of the four types of middleboxes. For Firewalls, FW-1 exhibits lower hardware and software problems compared to FW-2. However, the fraction of misconfigurations and connectivity errors is higher, due to better configuration tools and more reliable port connectors and cables for the new platform FW-2. Surprisingly, even though FW-2 has a higher capacity compared to FW-1, it experiences a high fraction of overload induced failures which are not observed in FW-1. Unlike firewalls, for IDPS devices, the port errors increased with the newer device IDPS-2, while contribution of the other problem types reduced. For load balancers, LB-3 exhibits fewer problems due to hardware and software improvements compared to its predecessors, LB-1 and LB-2. In sharp contrast to trends observed across FWs, IDPSes and LBs, VPN-2 exhibits a higher percentage of hardware faults and overload problems compared to VPN-1 due to the instability of the newer platform.

Overall, these trends provide a fine-grained analysis of what aspects of a new platform are improving over its predecessor and vice versa, to guide researchers and vendors in developing better designs and tools to improve middleboxes.

Findings (3): (1) Connection errors and interface problems dominate causes of middlebox failures, and (2) Hardware, misconfigurations and overload problems are present, but they are not in majority.

4.2 What are the main resolution actions?

Figure 13 shows the breakdown of actions taken to mitigate middlebox failures. The nine categories show the resolutions applied to faulty components at a fine granularity. For instance, the cable category implies that the cable was either cleaned or replaced. The software category denotes that either a patch was applied or the code was upgraded to the latest stable version. Similarly, the categories of different hardware components such as chassis, disk, fan, and PSU denote that they were replaced using on-site spare or sent back to the vendor for repairs, or warranty purposes.

We observe that across middleboxes, the most prevalent actions taken are replacing the cable and disks, and reboot to fix software-related issues. The former can be attributed to the relatively low-cost and high failure rates of these commodity components as operators tend to replace them quickly after observing a problem. In comparison, the decisions to replace specialized components such as chassis, supervisor engines, or even the device, requires a detailed

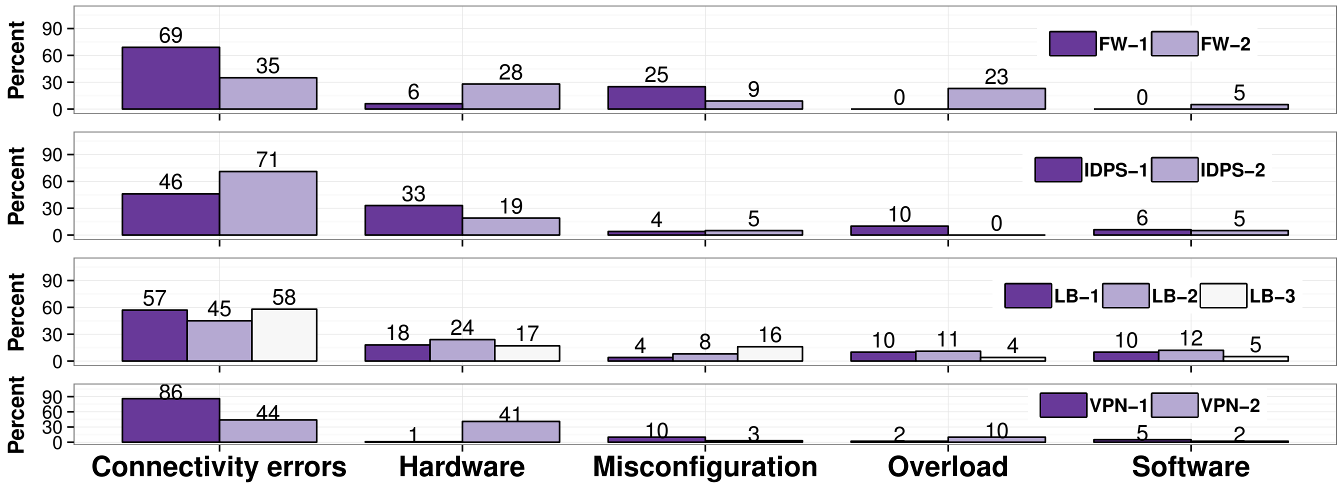


Figure 12: Comparison of problem types across different generations of middleboxes.

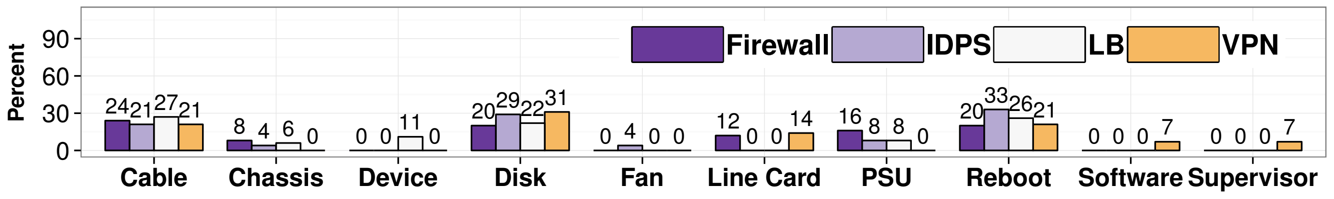


Figure 13: Resolution actions observed across middleboxes.

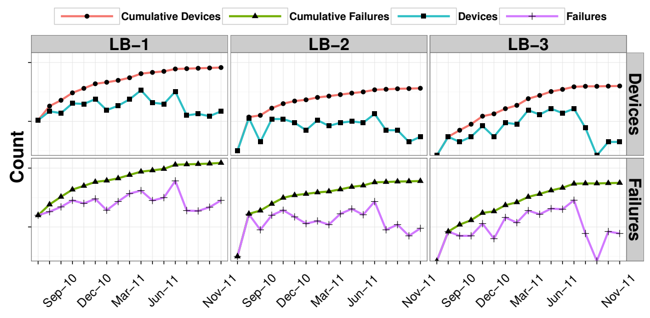


Figure 14: Failure trends across different generations of load balancers. The y-axes are unlabeled for confidentiality.

investigation due to their high cost and long turnaround time from their vendor. The reason for a high percentage of reboots is that it is a quick-fix solution in many cases and hence run as a popular first attempt to mitigate a problem to reduce downtime.

Findings (4): (1) Cables and disks are the most failing components, and (2) Device reboot is a popular quick-fix solution to mitigate middlebox problems.

5. FAILURE TREND ANALYSIS

In this section we analyze failure trends across different device generations of middleboxes. We use trend analysis as it is a useful tool to uncover outliers i.e., set of devices which are more failure-prone than the rest of the population. We present detailed case studies on LB[1-3].

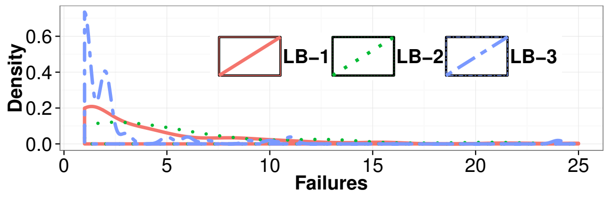


Figure 15: Distribution of load balancer failures.

Figure 14 shows the timeline for the number of failure events and the number of devices that failed; the Y-axis is on log scale. Each subplot shows two curves: a cumulative curve that sums up the number of failures (or devices that failed) across time and the other showing the actual number of failures (or devices that failed). For LB-1, the number of failure events far exceeds the number of devices that failed indicating that a few bad devices are responsible for a majority of their failures.

Validation. We validate this inference by computing two statistical metrics from the failure distribution (Figure 15). (1) skewness [33], an indicator used in distribution analysis as a sign of asymmetry where positive values indicate a right-skewed distribution and vice versa, and (2) COV. The observed skewness values are 2.2 (LB-1), 1.5 (LB-2) and 3.9 (LB-3), indicating that the distributions for LB-1 and LB-3 are right-skewed (tail on the right). Though the COV across all LBs is 0.6 (Table 2), the COV for LB-1 is 5.8 whereas it is 1.3 for LB-3 indicating that even though both have a right-skewed distribution, the “few bad devices” effect is higher in LB-1. Further, in Figure 14, the slope of the cumulative curve indicates the rate of increase of the observed value. Sudden increase in the slope indicates major problems during that period of observation. A key root cause of this behavior is connectivity problems as observed in Figure 12.

Combining failure rates with trouble tickets, trend analysis revealed two key issues:

1. **Early problem symptoms:** Inference on the problems observed in trouble tickets for LB-1 devices revealed a recurring PSU problem. Aggregating problem statistics across LB-1 tickets indicated that 46% of the tickets called for a device replacement due to a narrow focus of operators in mitigating a problem (compared to its diagnosis) and limited tools at their disposal in checking vendor-specific problems. In retrospect, a simple approach of aggregating root causes across a device generation can identify the problem patterns. In this case, failing PSUs could have been used as early indicators of problems with the product line. A *Cost of Ownership* analysis (§7) can then help decide whether to continue using LB-1 or gracefully upgrade it to a newer, more stable generation.
2. **The few bad apples affect:** LB-1 suffers from the *few bad apples* effect i.e., the rate of increase of the slope of the cumulative count of failures curve is higher than the rate of increase of the slope of the cumulative count of devices curve. This indicates that only a few devices are contributing to a majority of failures and is validated by high COV value for LB-1 of 5.8. There were several reasons why this happened. First, we observed a re-occurring fault using LB-1 trouble tickets with the regulated PSU, despite being redundant, that frequently caused power fluctuations leading to damaged parts. Even if the damaged parts are replaced, unless the fault is localized to the power supply unit, the problem likely will repeat itself. Second, a memory bit corruption (e.g., SDRAM ECC error) kept being masked temporarily by the quick-fix solution of reboots as observed in Figure 13.

These observations can be leveraged to (a) prioritize replacement of frequently failing devices and scheduling of repair actions based on the top-k observed problems, and (b) aid repair vs. replace analysis by combining early problem symptoms with a cost of ownership metric. Finally, frequently occurring hardware problems can aid in detecting faults at the vendor.

Findings (5): (1) A significant number of failures in one *generation* of load balancers exhibited the few bad apples effect, and (2) A family of load balancers having faulty PSUs could have been detected as early problem indicators using trend analysis.

6. REDUNDANCY ANALYSIS

In this section we first analyze the loss of traffic due to middlebox failures and then evaluate the effectiveness of redundancy in masking them.

6.1 How much traffic loss do failures cause?

Quantifying the impact of a failure is difficult as it requires attributing discrete outage levels to annotations used by network operators such as *severe, mild*, or *some impact*. As described in Section 2.3, we estimate the failure impact in terms of lost network traffic that would have been routed on a failed device in the absence of failure. Specifically, we first compute the median traffic (bytes/sec) on the device two hours before the failure and during the failure. The traffic loss is estimated as product of the failure duration and the difference between the two median traffic values.

Figure 16(a) shows the estimated traffic loss across the three LB generations. We observe the estimated median number of bytes lost during failures is about 1 GB for LB-1 (exhibiting a relatively high failure rate as seen in §3.1 and §5) while it is close to 5 MB for newer LB-3 generation devices, which have a low failure rate. Notice that LB-2, which also exhibited high failure rates compared to LB-3, shows a median traffic loss close to LB-1.

6.2 How effective is middlebox redundancy?

We next analyze how effective is middlebox redundancy, a *de-facto* technique for fault tolerance, in masking failures. Typically, middleboxes are deployed in 1:1 redundant groups, where one device is designated the primary and the other as backup. Based on the failure definition in Section 2.3, we evaluate redundancy effectiveness for LBs based on the traffic impact, and based on device malfunctioning for the other middlebox types.

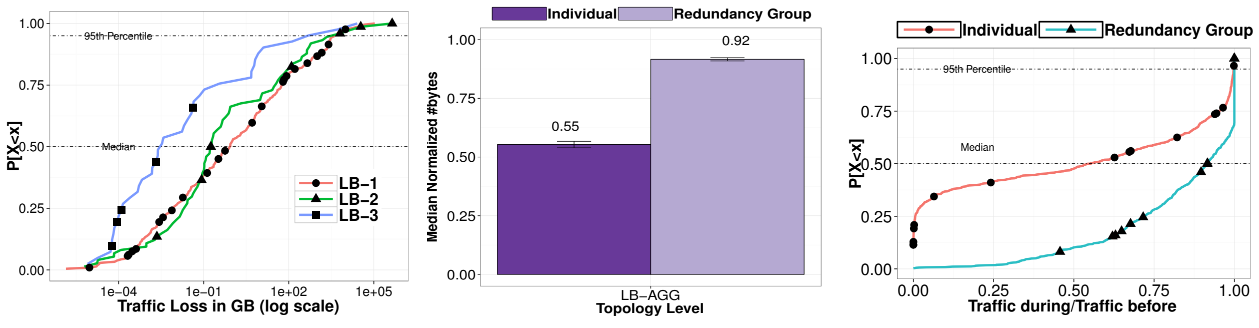
Evaluating LB redundancy. To estimate the effectiveness of LB redundancy, we first compute the ratio of median traffic on a device during a failure and its median traffic before the failure, and then compute this ratio across all devices in the redundancy group where the failure occurred. Network redundancy is considered 100% effective if this ratio is close to one across a redundancy group. We refer to this ratio as *normalized traffic* [17].

Figures 16(b) and 16(c) show the distribution of normalized traffic for individual LBs and their redundancy groups. We observe that the redundancy groups are effective at moving the ratio close to one with 40% of the events experiencing no failure impact at the redundancy group level. The median traffic carried at the redundancy group level is 92% compared with 55% at the individual level, a relative improvement of 67.2% in median traffic as a result of middlebox redundancy.

Evaluating redundancy for other middleboxes. To analyze redundancy effectiveness for FWs, IDPSes, and VPNs, we measure the fraction of overlapping failure events where both the redundant pairs were down simultaneously. In particular, after one of the pair fails, we check if the other device also undergoes a failure before the former becomes operational. Our analysis of redundancy effectiveness across FWs, IDPSes and VPNs revealed that both the redundant pairs failed in 33% and 1% of the failure events for FWs and IDPS, respectively. In comparison, VPN redundancy exhibits 100% effectiveness in mitigating failures.

Analyzing unsuccessful redundancy failover. Based on applying NetSieve [25] on trouble tickets, we observed two main reasons why the redundancy failover was not successful for LBs and FWs in masking the failure impact:

- **Faulty Failovers**: The primary device failed when the backup was experiencing an unrelated problem and hence led to a *failed failover*. In other cases, protocol bugs, bad cables, and software issues such as version incompatibility caused an unsuccessful failover, often sending devices into a deadlock state.



(a) (b) (c)

Figure 16: (a) Estimated traffic loss during failure events for LBs, (b) Normalized traffic (bytes) during failure events for a device as well as within a redundancy group and (c) Normalized bytes (quartiles) during failure events per device and across redundancy group compared across LBs.

- **Misconfigurations**: The same configuration error was made on both primary and backup devices. Primary reasons were due to copy-paste of the configuration files, mismatched cryptographic keys, and expired certificates.

Findings (6): (1) Middlebox redundancy is ineffective in 33% of the cases for load balancers and firewalls, and (2) Unsuccessful redundancy failover is mainly due to misconfigurations, faulty failovers and software version mismatch.

7. DISCUSSION

In this section, we discuss the research implications based on our middlebox reliability study.

Commoditize middleboxes: §1 and §3 indicated that hardware middleboxes contribute significantly to network failures, and that their failovers are not 100% effective (§6.2). Therefore, a natural alternative is to commoditize them [11–13] as they can enable a *scale-out* design and provide n:1 redundancy. While there has been promising work on software routers [34], achieving high performance, scalability and fault tolerance simultaneously poses several challenges. First, achieving high performance and programmability are often competing goals. For instance, the communication overhead introduced by software LBs may impact performance of latency-sensitive applications. Second, well-known software issues such as code bugs, frequent updates, and misconfigurations, may cause a commodity middlebox to be less reliable than hardware boxes. Finally, running a large number of commodity servers or VMs as middleboxes risks high operational costs such as power and cooling. Efforts such as [35, 36] to reduce network energy footprint can be explored to address them. In some cases, strict application requirements for hardware accelerators such as hardware-based deep packet inspection (DPI) for IDPS and deep-packet modification for WAN optimizers, may become a limiting factor.

Middlebox verification: §4 indicated that connection errors and interface problems dominate middlebox failures, and that there is a wide range of misconfigurations observed on middleboxes. As a result, a key requirement for network operations is to automate management and debugging of network problems e.g., using SDN [37, 38]. To improve network debugging, one approach is to do static and runtime verification of desired properties e.g., to check configurations, verify reachability, detect routing loops, guarantee isolation [37, 39, 40]. However, there are several challenges that need to be addressed. First, current tools lack support for common middlebox features such as tunneling and load balancing. Second, the task of parsing policy specifications and configurations is non-trivial and error-prone, as vendors use varying semantics which in-turn may change with device generations. Third, checking for correctness of policies involves designing a set of constraints that can be tested against the device configuration. However, defining a complete constraint set is difficult and evaluating it, computationally expensive [41]. Finally, the large-scale of data center networks makes it challenging to apply formal verification techniques due to the well-known state space explosion problem.

Detect early faulty product-line: In §5, we uncovered using trend analysis that an LB generation had lower reliability compared to other generations. This raises a serious dilemma: keep replacing faulty devices in existing deployment or upgrade to a new product line? While capital and operating costs dominate the cost vs. benefit analysis, answering this question is challenging for two additional reasons:

- Device familiarity: Upgrading devices with their high-speed, feature-rich counterparts will require additional training for network engineers (and has the undesirable consequence of increasing TTR due to unfamiliarity).
- Active/Standby Compatibility: Where 1:1 redundancy is dominant, both devices should be simultaneously upgraded to avoid unsuccessful redundancy failovers.

By computing a Cost of Ownership (COO) [42] metric, decisions such as whether to buy more spares for a faulty product line or to gracefully replace a product line can be made. The primary cost elements of the COO model are: (i) initial investment in purchasing the product, (ii) depreciation, (iii) cost of operation that includes space, personnel, and power expenses, (iv) cost of maintenance operations including downtime, repair cost, failure rates, and cost of spares, and (v) enhancement costs that include hardware/software upgrades and personnel training. Our reliability analysis attempts to provide some of these metrics as described in §3.1 and §3.3.

Feasibility of making middlebox processing as a cloud service: A recent proposal is to offload middlebox processing to the cloud [6]. Similar to CDNs, the idea is to use DNS-based redirection to route requests across multiple cloud PoPs and deploy middlebox instances in the cloud to process them based on defined policies. While this approach shifts the burden of owning and managing middleboxes from enterprises to the cloud, our study indicates that it risks making the problem worse as middlebox failures are prevalent in datacenters (§3.1) and redundancy is not 100% effective (§6.2) . Several other challenges also need to addressed before pushing middleboxes as a cloud hosted service: (a) how to guarantee correct middlebox traversal and coordinate state during churn? (b) how to manage the state and complexity for multiple tenants sharing the same set of middlebox instances? (c) how to address grey failures (§4.1) that can degrade service performance and availability? and (d) how to elastically scale-out and scale-in the processing capabilities (e.g., for stateful services) as demand changes? For instance, the stateful session nature of many middleboxes (e.g., load balancers) can limit how quickly instances can be (de)instantiated or risk a traffic spike as disconnected customers attempt to simultaneously rejoin when an instance is shutdown. To deliver traffic reliably in the presence of grey failures, multi-path protocols such as MP-TCP [10] can be explored.

Make the cost-metric of middlebox hardware reliability aware: To compare network hardware costs across platforms and vendors, the conventional metric is to use \$/Gb/s as the unit-cost basis e.g., \$ per 10 Gbps port. For instance, using this metric, we find that hardware LBs are about an order of magnitude more expensive than commodity Layer 3 switches. As §3 shows, middleboxes experience a significant fraction of failures, and incur substantial operating costs for repair and upgrade [6]. Thus, to balance the trade-off between improving service availability while cutting costs down, the cost metric should comprise both capital and operational expenses, e.g., \$/Gbps/99.9% availability, so that administrators can compute the total cost of ownership (TCO) and determine the network costs to provide a target SLA (e.g., 99.9%), to hosted services. Note that the accuracy of this metric needs to be continuously validated and improved by leveraging data from middlebox deployments in production.

8. RELATED WORK

Broadly, the research community has pursued two key directions to build better middleboxes:

1. **Middlebox Architectures**: Research in this direction proposes techniques for explicit control of the middleboxes e.g., middlebox communication (MIDCOM) [4, 43–45] and IETF Next Steps in Signaling (NSIS) [46]. The motivation behind these efforts draws from the fact that while operators increasingly require support for new applications (e.g., teleconferencing, cloud and mobile applications), they are unable to use third-party software/hardware specific to the application and are tied to their middlebox vendor to make the necessary upgrade [4, 6]. There has been some recent work [47, 48] in separating policy from reachability and centralized management of networks. Joseph et al. [2] proposed a policy-aware switching layer for data centers comprising inter-connected policy-aware switches that enable forwarding of different types of traffic through a specified sequence of middleboxes. dFence proposes adding DoS mitigation middleboxes on-demand on the data path to servers under attack [27].

2. **Middlebox Processing Analysis**: Research in this direction largely focused on either characterizing NAT properties such as mapping type and filtering rules [49, 50] or quantifying the end-to-end performance degradation induced by middleboxes [16]. Our work falls into the latter category of middlebox reliability analysis. We do not argue the benefits or shortcomings of current middlebox architectures but instead focus on analyzing their performance.

Failures in datacenters have received significant attention in the recent years [51–56]. Wang et al. measure the impact of middlebox policies on performance, energy and security in cellular networks [15]. However, they did not study middlebox failures. Our work is complementary to these efforts in that we focus on characterizing the device reliability across middleboxes and understanding the causes and impact of their failures.

Markopoulou *et al.* [54] studied failure in the Sprint backbone using passive optical taps and high-speed packet capture hardware. Their results indicated that 20% of all failures is due to planned maintenance and that maintenance is usually scheduled during periods of low network usage to minimize the impact on performance. As operators likely have a good understanding of problems under scheduled maintenance, we apply a planned maintenance filter on network events to analyze device-level reliability due to unexpected outages.

Our work draws some similarity with the analysis carried out by Turner et al. [56] and Gill et al. [17]. Turner et al. [56] used router configurations, syslog and email records to analyze network failures in the CENIC network that serves educational and research institutions in California. Gill et al. [17] study network failures in datacenters. Similar to us, they observe that LBs exhibit high annualized failure rates.

While we share the broader goal of studying network failures with these efforts, this work differs in three important ways driven by our goal of understanding the causes and impact of middlebox failures. First, we focus on several new questions to characterize middlebox reliability e.g., What are the key problems observed and their resolution actions? How prevalent are misconfigurations and what are their different types? Does reliability improve with new generations? and How effective is middlebox redundancy? that have not been addressed in the past. Second, our methodology filters maintenance events to identify "unexpected/unplanned" outages, filters events for devices with redundant failures, and performs hypothesis testing to validate traffic based impact filtering. Finally, our results provide useful guidelines to improve middlebox reliability.

9. CONCLUSION

Middleboxes form a critical component of the network infrastructure in datacenters and enterprises, yet, there have been few real-world studies on understanding the characteristics of middlebox failures. In this paper, we make one of the first attempts to do a large-scale characteristic study of real-world middleboxes to understand the causes and impact of their failures in datacenters. Our study reveals several surprising findings that differ from some commonly held views. We hope that this study motivates further research to improve middlebox reliability. In future work, we aim to leverage the middlebox failure models to predict their failures, develop techniques to improve middlebox redundancy, and build better tools for configuration management.

10. ACKNOWLEDGEMENTS

We thank our shepherd Z. Morley Mao and the anonymous reviewers for their feedback. Moises Goldszmidt and Nachiappan Nagappan provided invaluable help in verifying the failure analysis methodology. We are grateful to David St. Pierre for helping us understand the network data sets.

11. REFERENCES

[1] V. Sekar, S. Ratnasamy, M. Reiter, N. Egi, and G. Shi, "The Middlebox Manifesto: Enabling Innovation in Middlebox Deployment," in *HotNets*, 2011.

[2] D. Joseph, A. Tavakoli, and I. Stoica, "A Policy-Aware Switching Layer for Data Centers," in *SIGCOMM CCR*, 2008.

[3] M. Walfish, J. Stribling, M. Krohn, H. Balakrishnan, R. Morris, and S. Shenker, "Middleboxes No Longer Considered Harmful," in *OSDI*, 2004.

[4] P. Srisuresh, J. Kuthan, J. Rosenberg, A. Molitor, and A. Rayhan, "Middlebox Communication Architecture and Framework," *RFC 3303*, 2002.

[5] ABI, "Enterprise Network and Data Security Spending Shows Remarkable Resilience," http://goo.gl/t43ax, January 2011.

[6] J. Sherry, S. Hasan, C. Scott, A. Krishnamurthy, S. Ratnasamy, and V. Sekar, "Making Middleboxes Someone Else's Problem: Network Processing As a Cloud Service," in *SIGCOMM*, 2012.

[7] I. Week, "Data Center Outages Generate Big Losses," http://goo.gl/zBOfv, May 2011.

[8] "Why Gmail went down: Google misconfigured load balancing servers," http://goo.gl/NXTeW.

[9] "2011 ADC Security Survey Global Findings," http://goo.gl/A3b2Q.

[10] M. Scharf and A. Ford, "MP-TCP Application Interface Considerations," *draft-ietf-mptcp-api-00*, 2010.

[11] V. Sekar, N. Egi, S. Ratnasamy, M. Reiter, and G. Shi, "Design and Implementation of a Consolidated Middlebox Architecture," in *NSDI*, 2012.

[12] H. Uppal, V. Brajkovic, D. Brandon, T. Anderson, and A. Krishnamurthy, "ETTM: A Scalable Fault Tolerant Network Manager," in *NSDI*, 2011.

[13] A. Greenhalgh, F. Huici, M. Hoerdt, P. Papadimitriou, M. Handley, and L. Mathy, "Flow Processing and the Rise of Commodity Network Hardware," *SIGCOMM CCR*, 2009.

[14] V. Liu, A. Krishnamurthy, and T. Anderson, "F10: Fault-Tolerant Engineered Networks," in *NSDI*, 2013.

[15] Z. Wang, Z. Qian, Q. Xu, Z. Mao, and M. Zhang, "An Untold Story of Middleboxes in Cellular Networks," *SIGCOMM CCR*, 2011.

[16] M. Allman, "On Performance of Middleboxes," in *IMC*, 2003.

[17] P. Gill, N. Jain, and N. Nagappan, "Understanding Network Failures in Data Centers: Measurement, Analysis, and Implications," in *SIGCOMM*, 2011.

[18] "Cisco Data Center Network Architecture," http://goo.gl/kP28P.

[19] J. Lockwood, "Open Platform for Development of Network Processing Modules in Reprogrammable Hardware," in *DesignCon*, 2001.

[20] J. Case, M. Fedor, M. Schoffstall, and J. Davin, "Simple Network Management Protocol," SRI International Network Information Center, May 1990.

[21] M. McCloghrie, K. ad Rose, "Management Information Base for Network Management of TCP/IP-based internets," RFC 1213.

[22] D. Johnson, "NOC Internal Integrated Trouble Ticket System," RFC 1297, 1992.

[23] H. Mann and D. Whitney, "On a Test of Whether One of Two Random Variables is Stochastically Larger than the Other," in *The Annals of Mathematical Statistics*, 1947.

[24] A. Hart, "Mann-Whitney test is not just a test of medians: differences in spread can be important," 2001.

[25] R. Potharaju, N. Jain, and C. Nita-Rotaru, "Juggling the Jigsaw: Towards Automated Problem Inference from Network Trouble Tickets," in *NSDI*, 2013.

[26] C. E. Brown, "Coefficient of Variation," in *AMSGRS*, 1998.

[27] A. Mahimkar, J. Dange, V. Shmatikov, H. Vin, and Y. Zhang, "dFence: Transparent network-based denial of service mitigation," in *NSDI*, 2007.

[28] R. Sakia, "The Box-Cox Transformation Technique: A Review," in *JSTOR Statistician*, 1992.

[29] W. E. Leland, M. S. Taqqu, W. Willinger, and D. V. Wilson, "On the Self-Similar Nature of Ethernet Traffic (extended version)," in *IEEE ToN*, 1994.

[30] T. L. Bailey and C. Elkan, "Fitting a Mixture Model by Expectation Maximization to Discover Motifs in Bipolymers," in *ISMB*, 1994.

[31] T. K. Moon, "The Expectation-Maximization Algorithm," 1996.

[32] H. W. Lilliefors, "On the Kolmogorov-Smirnov Test for Normality with Mean and Variance Unknown," *JASA*, 1967.

[33] R. Bendel, S. Higgins, J. Teberg, and D. Pyke, "Comparison of Skewness Coefficient, Coefficient of Variation, and Gini Coefficient as Inequality Measures within Populations," in *Oecologia*, 1989.

[34] K. Argyraki, S. Baset, B. Chun, K. Fall, G. Iannaccone, A. Knies, E. Kohler, M. Manesh, S. Nedevschi, and S. Ratnasamy, "Can Software Routers Scale?" in *PRESTO*, 2008.

[35] S. Nedevschi, L. Popa, G. Iannaccone, S. Ratnasamy, and D. Wetherall, "Reducing Network Energy Consumption via Sleeping and Rate-adaptation," in *NSDI*, 2008.

[36] A. Greenberg, P. Lahiri, D. Maltz, P. Patel, and S. Sengupta, "Towards a Next Generation Data Center Architecture: Scalability and Commoditization," in *PRESTO*. ACM, 2008.

[37] P. Kazemian, G. Varghese, and N. McKeown, "Header space analysis: Static checking for networks," in *NSDI*, 2012.

[38] N. Handigol, B. Heller, V. Jeyakumar, D. Mazières, and N. McKeown, "Where is the debugger for my software-defined network?" in *Proceedings of the first workshop on Hot topics in software defined networks*. ACM, 2012, pp. 55–60.

[39] H. Mai, A. Khurshid, R. Agarwal, M. Caesar, P. Godfrey, and S. King, "Debugging the Data Plane with Anteater," *SIGCOMM CCR*, 2011.

[40] N. Feamster and H. Balakrishnan, "Detecting BGP configuration Faults with Static Analysis," in *NSDI*, 2005.

[41] A. Feldmann and J. Rexford, "IP network Configuration for Intra-domain Traffic Engineering," *Network, IEEE*, 2001.

[42] L. Ellram, "Total Cost of Ownership: An Analysis Approach for Purchasing," in *Journal of PDLM*, 1995.

[43] R. Swale, P. Mart, P. Sijben, S. Brim, and M. Shore, "Middlebox Communications Protocol Requirements," *RFC 3304*, 2002.

[44] B. Carpenter and S. Brim, "Middleboxes: Taxonomy and Issues," *RFC 3234*, 2002.

[45] M. Stiemerling and J. Quittek, "Middlebox communication (MIDCOM) protocol semantics," *RFC 4097*, 2008.

[46] R. Hancock, S. Bosch, G. Karagiannis, and J. Loughney, "Next steps in signaling (NSIS): Framework," in *IETF RFC 4080*, 2005.

[47] A. Greenberg, G. Hjalmtysson, D. Maltz, A. Myers, J. Rexford, G. Xie, H. Yan, J. Zhan, and H. Zhang, "Clean slate 4D approach to Network Control and Management," in *SIGCOMM*, 2005.

[48] M. Casado, M. Freedman, J. Pettit, J. Luo, N. McKeown, and S. Shenker, "Ethane: Taking Control of the Enterprise," in *SIGCOMM CCR*, 2007.

[49] J. Eppinger, "TCP connections for P2P Apps: Software Approach to Solving the NAT Problem," in *ISR*, 2005.

[50] A. Biggadike, D. Ferullo, G. Wilson, and A. Perrig, "NATBLASTER: Establishing TCP Connections Between Hosts Behind NATs," in *ACM SIGCOMM Workshop*, 2005.

[51] S. Kandula, R. Mahajan, P. Verkaik, S. Agarwal, J. Padhye, and P. Bahl, "Detailed Diagnosis in Enterprise Networks," in *SIGCOMM*, 2009.

[52] V. Padmanabhan, S. Ramabhadran, S. Agarwal, and J. Padhye, "A Study of End-to-End Web Access Failures," in *CoNext*, 2006.

[53] C. Labovitz, A. Ahuja, and F. Jahanian, "Experimental Study of Internet Stability and Backbone Failures," in *FTC*, 1999.

[54] A. Markopoulou, G. Iannaccone, S. Bhattacharyya, C. Chuah, Y. Ganjali, and C. Diot, "Characterization of Failures in an Operational IP Backbone Network," in *IEEE/ACM TON*, 2008.

[55] A. Shaikh, C. Isett, A. Greenberg, M. Roughan, and J. Gottlieb, "A Case Study of OSPF Behavior in a Large Enterprise Network," in *ACM SIGCOMM WIM*, 2002.

[56] D. Turner, K. Levchenko, A. Snoeren, and S. Savage, "California Fault Lines: Understanding the Causes and Impact of Network Failures," in *ACM SIGCOMM CCR*, 2010.