A Large Scale Study of Data Center Network Reliability

Justin Meza
Carnegie Mellon University
Facebook, Inc.
jjm@fb.com

Tianyin Xu
Facebook, Inc.
tianyin@fb.com

Kaushik Veeraraghavan
Facebook, Inc.
kaushikv@fb.com

Onur Mutlu
ETH Zürich
Carnegie Mellon University
omutlu@gmail.com

ABSTRACT

The ability to tolerate, remediate, and recover from network incidents (e.g., caused by device failures and fiber cuts) is critical for building and operating highly-available web services. Achieving fault tolerance and failure preparedness requires system architects, software developers, and site operators to have a deep understanding of network reliability at scale, along with its implications to data center systems. Unfortunately, little has been reported on the reliability characteristics of large-scale data center network infrastructure, let alone its impact on the availability of services powered by software running on that network infrastructure (service-level availability).

This paper fills the gap by presenting a large-scale, longitudinal study of data center network reliability based on operational data collected from the production network infrastructure at Facebook, one of the largest web service providers in the world. The study covers reliability characteristics of both intra and inter data center networks. For intra data center networks, we study seven years of operation data comprising thousands of network incidents across two different data center network designs – a classic cluster-based architecture and a state-of-the-art fabric-based topology. For inter data center networks, we study eighteen months of recent repair tickets in the field to understand reliability of WAN backbones. In contrast to prior work, we study the effects of network reliability on web services, and how these reliability characteristics evolve over time. We discuss the implications of network reliability on the design, implementation, and operation of large-scale data center systems and how it affects highly-available web services. We hope our study forms the foundation of understanding the reliability of large-scale network infrastructure, and inspires new reliability solutions to network incidents.

KEYWORDS

data centers, network, reliability

IMC '18, October 31–November 2, 2018, Boston, MA, USA

ACM ISBN TBA…$TBA
https://doi.org/TBA

ACM Reference Format:
Justin Meza, Tianyin Xu, Kaushik Veeraraghavan, and Onur Mutlu. 2018. A Large Scale Study of Data Center Network Reliability. In *Proceedings of IMC '18.* ACM, New York, NY, USA, 17 pages. https://doi.org/TBA

1 INTRODUCTION

Data center network infrastructure, consisting of both intra and inter data center networks, is among the cornerstones of large-scale, geo-distributed web services. In the last two decades, data center network infrastructure has been evolving rapidly, driven by the scalability challenges derived from ever-increasing traffic demand and the consequent desire for even bigger data centers and more data centers across the planet. New technologies for data center networks [1–3, 24, 34, 36, 37, 42, 48, 54, 57, 70] and WAN backbones [38, 40, 44, 59] have been designed, deployed, and operated in the field. At a high level, both data center and backbone networks are moving from traditional Clos-based network designs [19] running on large vendor hardware toward designs featuring more centralized automation, monitoring, and control.

Despite the significance of network infrastructure in data center operation, we find that three aspects of data center network infrastructure are *not* well understood in the literature. First, little has been reported on the *overall* reliability characteristics of data center network infrastructure – from the networks *within* data centers, to the networks that interconnect different data centers. Second, no previous study that we are aware of has examined the reliability trends that appear with the transition away from cluster-based Clos network designs toward modern fabric-based topologies with automated failover [70]. Third, there is relatively little discussion regarding the impact of network design decisions from the perspective of the software services that run on top of the networks (i.e., the network's effect on *service-level availability*).

In the past, lack of understanding has hindered researchers and practitioners while combating network incidents [7]. Today, network incidents have become a major root cause of data center system outages, as shown in recent studies [8, 27, 35, 60] and news reports [20, 23, 31, 32, 72]. For example, Gunawi et al. [35] studied 597 unplanned web and cloud service outages from headline news and public postmortem reports in a 7-year span from 2009 to 2015. Their study shows that network incidents contributed to 15% of these outages, constituting the second largest root cause category. Therefore, understanding network reliability and its implications on service-level availablity is of critical importance to the design, implementation, and operation of large-scale data center systems.

A number of prior studies examined the reliability of data center networks and the network devices [29, 30, 33, 62, 63]. However, most of these studies focus on failure characteristics of network devices and links, *without* connecting their impact to *service-level events*. Service-level events often manifest as failures and anomalies in the production systems running on data center network infrastructure and transmitting data on the network. In fact, all device- and link-level failures are *not* created equal – many failures are masked by built-in hardware redundancy, path diversity, and other fault-tolerance logic. Compared with network device failures,[1] *network incidents*, i.e., misbehavior of the network as a whole that impairs the availability or quality of large-scale web services, are among the least understood types of failures in the literature.

This paper fills the gap by presenting a large-scale, longitudinal study on the reliability of data center network infrastructure. The study is based on seven years of *intra* data center operation data and eighteen months of *inter* data center operation data collected from the production systems of Facebook, one of the largest web service providers in the world. Facebook serves 2.4 billion monthly users and operates twelve geographically distributed data centers with multiple generations of data center network design; these data centers are interconnected via our *Wide Area Network (WAN)* backbone designed for serving different use cases (we will provide more detail on these use cases in §3).

We study the reliability characteristics of both data center and backbone networks from the perspective of *service-level incidents in large-scale systems*, and how these reliability characteristics evolve over time. Specifically, we discuss in depth how data center network reliability influences the design, implementation, and operation of large-scale software systems that run highly-available web services. This paper makes the following major observations:

- Among the failures that cannot be remediated by software, we observe that most failures involve maintenance, faulty hardware, and misconfiguration as devices and routing configurations become more complex and challenging to maintain (§5.1).
- Network devices with higher bisection bandwidth have a higher likelihood of affecting service-level software and that network devices built from collodity chips have much lower incident rates compared to devices from third-party vendors due to the ease of integration with automated failover and remediation software (§5.2).
- Rack switches contribute to around 28% of the service-level incidents that occurred in the past year (2017). Despite rack switches having the largest mean time to incident (over ten million hours) compared to other types of network devices, the sheer size of the rack switch population leads to a noticeable reliability effect on the software systems running in data centers (§5.4).
- Core network devices that connect data centers and the backbone contribute to around 34% of service-level incidents compared to other network devices. This is because network devices with higher bisection bandwidth tend to affect a larger number of connected downstream devices and are thus correlated with widespread impact when these types of devices fail (§5.4).
- As opposed to cluster-based Clos networks, fabric networks contributed to around 50% of the number of *intra* data center network incidents in the past year (2017). We find that data center fabric networks are more reliable due to their simpler, commodity-chip based switches and automated remediation software that dynamically adapts to and tolerate device failures (§5.5).
- While high reliability is essential for widely deployed devices, such as RSWs, incident rate can be greatly decreased through the use of software managed failover and automated remediation (§5.6).
- We model the reliability of a diverse set of edge networks and links that provide backbone connectivity and find that time to failure and time to repair closely follow *exponential* functions. We provide models for these phenomena so that future studies can compare against them and use them to understand the nature of backbone failures (§6.1–§6.3).
- Backbone links that connect data centers typically fail on the order of weeks to months and typically recover on the order of hours. However, some backbone links exhibit very high mean time to repair, on the order of days. In these cases, path diversity in the backbone topology ensures that large-scale networks can tolerate failures with long repair times (§6.1, §6.1).
- Backbone link vendors exhibit a wide degree of variance in failure rates of their backbone links. This problem makes the task of planning and maintaining network connectivity and capacity a key challenge for large-scale web service companies (§6.2).

2 MOTIVATION

The reliability of data center network infrastructure is critically important for building and operating highly available and scalable web applications [7, 16]. Despite an abundance of device- and link-level monitoring, the effects of network infrastructure reliability on the production systems that run on them is not well understood.

The fundamental problem lies in *the difficulty of correlating device- and link-level faults with system-level impact.* First, with the redundancy built into most network infrastructure (including device, path, and protocol redundancy), many faults do *not* manifest as issues in the production systems that run on them. Second, large-scale network infrastructure is typically equipped with automated mitigation and failover mechanisms that take actions to resolve faults when they occur. As we will discuss in §4.1, at Facebook the vast majority of device- and link-level faults are automatically mitigated and resolved, with neither human intervention nor noticeable service-level impact.

As a result, we argue that it is *impossible to fully characterize the impact of network incidents* in the wild without examining how they affect services (their service-level effects). The systems community coined a term for this type behavior that can only be studied at a full-system scale: *emergent behavior*. Emergent behavior describes the unexpected ways that complex systems (mis)behave that are

[1] We use "failures" to refer to any misbehavior of a network device. The root cause of a failure includes not only hardware faults, but also misconfigurations, maintenance mistakes, and firmware bugs.

not easily predictable from the behavior of their components [55]. We adopt this term to describe the types of system-level effects that we examine in large scale data center networks. As software systems grow in complexity, interconnectedness, and geographic distribution, unwanted emergent behavior from network infrastructure has the potential to become a key limiting factor in the ability to reliably operate software systems at a large scale.

To understand the behavior of network failures we must be able to answer questions such as: *How long do network failures affect software when they occur?*, *What are the root causes of the network failures that affect software?*, and *How do network failures manifest themselves in production systems?* Unfortunately, past efforts to understand network incidents in large-scale network infrastructure are limited to informal surveys and a small number of public postmortem reports [7, 35], which could be biased toward certain types of failures and not comprehensive. As noted in [7], due to scant evidence and even less data, it is hard to discuss the reliability of distributed systems in the face of network incidents: *"much of what we believe about the failure modes is founded on guesswork and rumor."*

Even for large web and cloud service providers, understanding the reliability of network infrastructure is challenging, given the complex, dynamic, and heterogeneous nature of large-scale networks. With complex and constantly evolving network designs built from a wide variety of vendor devices, it is hard to reason about the end-to-end reliability of network infrastructure under different failure modes, let alone how it affects the software that runs on it. As far as we know, from what limited information has been publicly discussed, this is a common challenge across the industry.

Facebook has attempted to address this challenge by seeking to understand the reliability of its data center network infrastructure. At Facebook, service-level events that affect reliability (known as *SEVs*) are rigorously documented and reviewed to uncover their root causes, duration, service-level impact, as well as recommended mitigation and recovery procedures (cf. §4.2) [52]. These postmortem reports form an invaluable source of information for analyzing and understanding network reliability from the perspective of large-scale web services.

Our goal is to shed light on the network reliability incidents that affect the design of reliable large scale systems both *within* and *between* data centers and its system-level effects. Our study focuses on the perspective of the software systems that power large-scale web services. We hope that this work helps researchers and practitioners anticipate and prepare for network incidents, and inspire new reliability solutions.

3 FACEBOOK'S NETWORK ARCHITECTURE

Figure 1 shows Facebook's network architecture [3, 24, 41, 67]. Facebook's network consists of interconnected data center *regions*. Each region contains buildings called *data centers*. Facebook operates both data center and backbone networks. We call the network *within* data centers the *intra* data center network and the backbone network *between* data centers the *inter* data center network.

The diversity of Facebook's network provides an opportunity to compare the reliability across network designs. Though diverse, Facebook's network is by no means unique. Published network architectures from Google and Microsoft use similar design principles and building blocks [30, 33, 40, 62, 63, 70]. We suspect our findings and implications apply to other large scale data center networks.

3.1 Intra Data Center Networks

Facebook uses two intra data center network designs: an older *cluster-based Clos* design [19, 24] and an newer *data center fabric* design [3]. We call these the *cluster network* and the *fabric network*. Unlike the cluster network, the fabric network uses a five-stage Folded Clos [1], built from simple commodity hardware, with automated repairs.

Cluster network design. In Facebook's older network (Figure 1, Region A), a *cluster* is the basic unit of network deployment. Each cluster comprises four *cluster switches (CSWs, ①)*, each of which aggregates physically contiguous *rack switches (RSW, ②)* via 10 Gb/s Ethernet links. In turn, a *cluster switch aggregator (CSA, ③)* aggregates CSWs and keeps inter cluster traffic within the data center. Inter data center traffic flows through *core network devices (Cores, ④)*, which aggregate CSAs.

A cluster network has two main limitations:

(1) ***Hard wired switches*** require manual repair. When a port becomes unresponsive, a human must inspect and repair the port. When a device becomes unresponsive, a human must power cycle the device. Compared to software, humans perform slow repairs. Slow repairs mean fewer switches to route requests, higher bisection bandwidth on the remaining switches, and more congestion in the network.
(2) ***Proprietary switches*** are challenging to maintain and customize. The proprietary firmware on switches requires technicians trained in the vendor's software stack to configure and update settings. Proprietary software on switches makes customization difficult or impossible. Once deployed, proprietary switches must be repaired in-place.

Despite its limitations, the cluster networks remain in use in a dwindling fraction of Facebook's data centers. Ultimately, these data centers will join new data centers in using the fabric network design.

Fabric network design. Facebook's newer network (Figure 1, Region B) addresses the cluster network's limitations. A *pod* is the basic unit of network deployment in a fabric network. Unlike the physically contiguous RSWs in a cluster, RSWs in a pod have no physical constraints within a data center. Each RSW (⑥) connects to four *fabric switches (FSWs, ⑦)*. The 1:4 ratio of RSWs to FSWs maintains the connectivity benefits of the cluster network. *Spine switches (SSWs, ⑧)* aggregate a dynamic number of FSWs, defined by software. Each SSW connects to a set of *edge switches (ESWs, ⑨)*. Core network devices (⑩) connect ESWs between data centers.

Fabric networks are managed by software and differ from cluster networks in three ways:

(1) ***Simple, custom switches.*** Fabric devices contain simple, commodity chips and eschew proprietary firmware and software.
(2) ***Fungible resources.*** The devices in a fabric network are *not* connected in a strict hierarchy. Control software manages FSWs, SSWs, and ESWs like a fungible pool of resources.

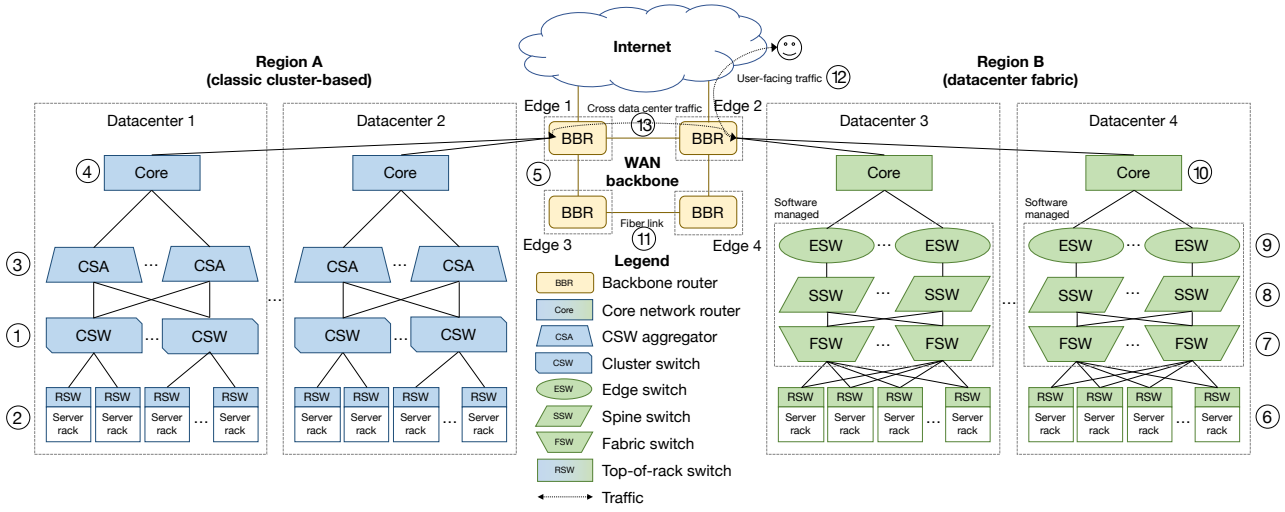

Figure 1: Facebook's network architecture, explained in §3. Data centers use either an older cluster network or a newer fabric network. Cluster networks and fabric networks communicate across the WAN backbone and Internet.

Resources dynamically expand or contract based on network bandwidth and reliability needs.

(3) ***Automated repairs.*** Failures on data center fabric devices can be repaired automatically by software [65]. Centralized management software continuously checks for device misbehavior. A skipped heartbeat or an inconsistent network setting raise alarms for management software to handle. Management software triages the problem and attempts to perform automated repairs. Repairs include restarting device interfaces, restarting the device itself, and deleting and restoring a device's persistent storage. If the repair fails, management software opens a support ticket for investigation by a human.

(4) ***Stacked devices.*** The same type of fabric device can be stacked in the same rack to create a higher bandwidth virtual device. Stacking enables port density in fabric networks to scale faster than proprietary network devices [4–6, 69].

Both cluster networks and fabric networks use *backbone routers (BBRs) located in edges (⑤)* to communicate across the WAN backbone and Internet.

3.2 Inter Data Center Networks

Facebook's physical backbone infrastructure can be abstracted as *edge nodes* connected through *fiber links* (⑪ in Figure 1). Edge nodes are the geographical location where Facebook deploys hardware to route backbone traffic, while fiber links are abstraction of physical optical fibers that connects the edges. Each end-to-end *fiber link* is embodied by optical *circuits* that consist of multiple optical segments. An optical segment corresponds to a fiber and carries multiple channels, where each channel corresponds to a different wavelength mapped to a specific router port.

The reliability of fiber links is critically important to systems that run across multiple data centers, especially those requiring consistency and high-availability [7, 16]. Without careful planning, fiber cuts (e.g., due to natural disasters) would cause network partitions that cut off an entire data center or region from the rest of the network. At Facebook, we have not seen catastrophic network partitions that disconnect data centers from each other. This is in part due to network planning decisions made from simulations based on the data presented in this section. The more common results of fiber cuts are the loss of capacity from edges to regions or between two regions. In this case, we have to reroute the traffic using other available links, which could increase end-to-end latency.

On top of the physical fiber-based infrastructure, multiple WAN backbone architectures are employed to satisfy the distinct characteristics and requirements of two types of traffic labeled in Figure 1: (1) user-facing traffic and (2) cross data center traffic.

- **User-facing traffic** (⑫ in Figure 1) connects a person using Facebook applications like those hosted at `www.facebook.com`, to software systems running in Facebook data centers. In order to reach a Facebook data center, user-facing traffic goes through the broader Internet via a process known as *peering* [76]. There, *Internet Service Providers (ISPs)* exchange traffic with other Internet domains. User-generated traffic uses the *Domain Name System (DNS)* to connect users to geographically local servers operated by Facebook called *edge presences* (also known as points of presence) [68, 76]. From there, user traffic is delivered to Facebook's data center regions through the backbone network.
- **Cross data center traffic** (⑬ in Figure 1) connects a service running in one Facebook data center to a service running in another Facebook data center. The backbone network interconnects those data center regions, including both the classic network and the fabric network regions. By volume, cross data center traffic consists primarily by *bulk data transfer* streams of data such as for replication and consistency. These are generated by back end services that perform batch

processing [21, 50], distributed storage [10, 56], and real-time processing [18, 39], for example.

To serve user-facing traffic, the backbone networks support a broad range of protocols and standards to connect to a wide variety of external networks from different ISPs. Facebook uses a traditional WAN backbone design that consists of backbone routers placed in every edge (the BBRs in Edge 1 through 4 in the WAN backbone, ⑤ in Figure 1).

On the other hand, as cross data center traffic is mostly for bulk data transfer, the traffic is only destined to a few dozen data centers and is partitioned in the optical layer in four planes where each plane has one backbone router per data center [41]. Moreover, inter data center traffic is managed by software systems where centralized traffic engineering is employed to manage inter data center backbone routers built from commodity chips. The design in principle is similar as Google's B2 and B4 described in [33, 40].

4 METHODOLOGY

We next describe how we measure and analyze the reliability of intra and inter data center networks, including the scope of our study (§4.1), the data sources (§4.2), and the analytical methodology (§4.3).

4.1 Network Incident Definition

We define *network incidents* as network-level events that have observable impact (e.g., data corruption, timed out connections, and excessive latency) on Facebook's production systems, such as our frontend web servers [22], caching systems [17, 58], storage systems [10, 56], data processing systems [18, 39], and real-time monitoring systems [43, 61]. Most of these network incidents are tolerated or mitigated by the system software and do not cause service-level impact.

4.1.1 Remediation Prevents Network Incidents. Not every network event or failure leads to an incident. Starting in 2013, Facebook began to automate the process of *remediating* common modes of failure for certain network devices, such as RSWs and later FSWs and certain models of Core devices, using an automated repair system [65]. Facebook relies on this automated repair system to shield our infrastructure from the vast majority of issues that arise in our intra data center networks. Remediation coordinates between using software to repair simple issues and alerting human technicians to repair complex issues.

4.1.2 The Effects of Automated Remediation. While automated repair is employed only for RSWs, FSWs, and a small percentage of Core devices, it has provided significant benefits. In a recent month (from April 1, 2018 to May 1, 2018), only 1 out of every 397 issues for RSWs was unable to be fixed by automated repair and required human intervention. During the same period, only 1 out of every 214 and 1 out of every 4 issues were unable to be fixed by automated repair for FSWs and Core devices, respectively. Note that Core devices, where Facebook has not yet pervasively deployed its own software stack, are unable to take as much advantage of automated repair as RSWs and FSWs.

4.1.3 Characterizing Automated Remediation. Table 1 summarizes the automated repair data we analyzed and also lists additional details per device type. Each repair is assigned a *priority* from 0 (highest priority) to 3 (lowest priority). We see that Core devices, some of the most critical devices in the data center network, are assigned repairs with the highest priority. In contrast, FSW and RSW repairs are assigned lower priorities on average: 2.25 and 2.22, respectively. The automated repair system uses a repair's priority to schedule when the repair should take place. Repairs assigned a lower priority wait longer than repairs assigned a higher priority. Core device repairs have the highest priority and only have to wait four minutes on average. FSW and FSW repairs have a much lower priority and wait up to *three days* for repair. The repairs themselves happen relatively quickly, taking less than a minute on average. Core device repairs can sometimes be more complex and take around 30.1 seconds on average. FSW and RSW repairs take only 4.45 and 2.91 seconds on average, respectively.

Device	Repair Ratio	Avg Priority / Wait / Repair Time
Core	75%	0 (highest) / 4 m / 30.1 s
FSW	99.5%	2.25 / 3 d / 4.45 s
RSW	99.7%	2.22 / 1 d / 2.91 s

Table 1: The repair ratio (fraction of issues repaired with automated remediation versus those which ultimately led to a network incident) and average priority (0 = highest, 3 = lowest), average wait time, and average repair time for the network device types that support automated remediation.

Some network device issues cannot be remediated with software alone and require help from a technician to repair. The most frequent 90% of automated repairs are: device port ping failures that are repaired by turning the port off and on again (50% of remediations), configuration file backup failures are repaired by restarting the configuration service and reestablishing a secure shell connection (32.4% of remediations), fan failures which are remediated by extracting failure details and alerting a technician to examine the faulty fan (4.5% of remediations), unable to ping the device from a dedicated service to monitor device liveness which collects details about the device and assigns a task to a technician (4.0% of remediations).

We focus our analysis on the class of incidents that can *not* be solved by automated repair. Our intent is to portray, as accurately as possible, the types of non-trivial failures that cause production impact.

4.2 Service-Level Events (SEVs)

We analyze a production incident dataset collected over seven years, from 2011 to 2018. The dataset comprises thousands of incidents. We describe the dataset structure next.

Engineers at Facebook routinely document infrastructure incidents. Facebook calls such incidents *Service-level EVents (SEVs[2])*. SEVs fall into three categories of severity ranging from SEV3 (lowest severity, no external outage) to SEV1 (highest severity, widespread external outage). Engineers who responded to a SEV, or whose service the SEV affected, write its report. The report contains the

[2]Pronounced [sɛv] in International Phonetic Alphabet, rhyming with dev.

incident's root cause, the root cause's affect on services, and steps to prevent the incident from happening again [52]. Each SEV goes through a review process to verify the accuracy and completeness of the report. SEV reports help engineers at Facebook prevent similar incidents from happening again.

The SEV report dataset resides in a MySQL database. The database contains reports dating to January 2011. Network SEVs contain details on the incident: the network device implicated in the incident, the duration of the incident (measured from when the root cause manifested until when engineers fixed the root cause), the incident's affects on services (for example, increased load from lost capacity, message retries from corrupted packets, downtime from partitioned connectivity, and increased latency from congested links). We use SQL queries to analyze the SEV report dataset for our study.

SEVs come in a variety of shapes and sizes. We summarize three representative SEVs with differing levels of service affect:

Switch crash from software bug (SEV3): A bug in the switching software triggered an RSW to crash whenever the software disabled a port. The incident occurred on August 17, 2017 at 11:52 am PDT after an engineer updated the software on a RSW and noticed the behavior. The engineer identified the root cause by reproducing the crash and debugging the software: an attempt to allocate a new hardware counter failed, triggering a hardware fault. On August 22, 2017 at 11:51 am PDT the engineer fixed the bug and confirmed the fix.

Traffic drop from faulty hardware module (SEV2): A faulty hardware module in a CSA caused traffic to drop on October 25, 2013 between 7:39 am PDT and 7:44 am PDT. After the drop, traffic shifted rapidly to alternate network devices. Web servers and cache servers, unable to handle the influx of load, exhausted their CPU and failed 2.4% of requests. Service resumed normally after five minutes when web servers and cache servers recovered. An on-site technician diagnosed the problem, replaced the faulty hardware module, verified the fix, and closed the SEV on October 26, 2013 at 8:22 am PDT.

Data center outage from incorrect load balancing (SEV1): A DR with an incorrectly configured load balancing policy caused a data center network outage on January 25, 2012 at 3:46 am PST. Following a software upgrade, a DR began routing traffic on a single path, overloading the ports associated with the path. The overload at the DR level caused a data center outage. Site reliability engineers detected the incident with alarms. Engineers working on the DR immediately attempted to downgrade the software. Despite the downgrade, DR load remained imbalanced. An engineer resolved the incident by manually resetting the load balancer and configuring a particular load balancer setting. The engineer closed the SEV on January 25, 2012 at 7:47 am PST.

4.3 Analytical Methodology

We examine two sets of data for our study. For intra data center reliability, we use seven years of service-level event data collected from our SEV database (cf. §4.2). For inter data center reliability we use eighteen months of data collected from vendors on fiber repairs collected between October 2016 and April 2018. We describe the analysis for each data source below.

4.3.1 Intra Data Center Networks. For intra data center reliability, we study the network incidents in three aspects: (1) root cause, (2) network device types, and (3) network architecture (cluster-based design versus data center fabric).

We use the root causes chosen by the engineers who authored the corresponding SEV reports. The root cause category (listed in Table 2) is a mandatory field in our SEV authoring workflow.

To classify network incidents by the type of offending devices, we leverage the naming convention enforced by Facebook where each network device is named with a unique, machine-understandable string prefixed with the device type. For example, every rack switch has a name prefixed with "`rsw.`". Therefore, by parsing the prefix of the name of the offending device, we are able to classify the SEVs based on the device types.

We also classify network incidents based on network architecture. Recall from Figure 1, CSA and CSW belong to classic cluster-based networks, and ESW, SSW, and FSW devices are a part of the data center fabric.

4.3.2 Inter Data Center Networks (Backbone). For inter data center reliability, we study the reliability of end-to-end fiber links (§3.2) based on repair tickets from fiber vendors whose links form Facebook's backbone networks that connect the data centers. Facebook has extensive monitoring systems that check the health of every fiber link, as unavailability of the links could significantly affect the ingress traffic or can disconnect a data center from the rest of our infrastructure.

When the vendor starts repairing a link (when the link is down) or performing maintenance for a fiber link, Facebook is notified via email. The email is in a structured form, including the logical IDs of the fiber link, the physical location of the affected fiber circuits, the starting time of the repair/maintenance, the estimated duration, etc. Similarly, when the vendor completes the repair/maintenance of a link, they send an email for confirmation. The emails are automatically parsed and stored in a database for later analysis. We examine the eighteen months of repair data in this database ranging from October 2016 to to April 2018. From this data, we measure the mean time to incident (MTTI) and mean time to repair (MTTR) of fiber links.

4.3.3 Limitations and Conflating Factors. We found it challenging to control for all variables in a longitudinal study of failures at a company of Facebook's scale. So, our study has limitations and conflating factors:

- ***Absolute versus relative number of failures.*** We cannot report the absolute number of failures for legal reasons. Instead, we report failure rates using a fixed baseline when the trend of the absolute failures aids our discussion.
- ***Logged versus unlogged failures.*** Our intra data center network study relies on SEVs reported by employees. While Facebook fosters a culture of opening SEVs for all incidents affecting production, we cannot guarantee our incident dataset is exhaustive.
- ***Technology changes over time.*** Switch hardware consists of a variety of devices sourced and assembled from multiple vendors. We do not account for these factors in our study. Instead,

Table 2: Common root causes of intra data center network incidents in a seven-year period from 2011 to 2018.

Category	Distribution	Description
Maintenance	17%	Routine maintenance (for example, upgrading the software and firmware of network devices).
Hardware	13%	Failing devices (for example, faulty memory modules, processors, and ports).
Configuration	13%	Incorrect or unintended configurations (for example, routing rules blocking production traffic).
Bug	12%	Logical errors in network device software or firmware.
Accidents	10%	Unintended actions (for example, disconnecting or power cycling the wrong network device).
Capacity	5%	High load due to insufficient capacity planning.
Undetermined	29%	Inconclusive root cause.

we analyze trends by switch type when a switch's architecture significantly deviates from others.
- ***Switch maturity.*** Switch architectures vary in their lifecycle, from newly-introduced switches to switches ready for retirement. We do differentiate between a switch's maturity in Facebook's fleet.

Thoughout our analysis, we state when a factor not under our control may affect our conclusions.

5 INTRA DATA CENTER RELIABILITY

In this section, we study the reliability of data center networks. We analyze network incidents within Facebook data centers over the course of seven years, from 2011 to 2018, comprising thousands of real world events. A network incident occurs when the root cause of a SEV relates to a network device. We analyze root causes (§5.1), incident rate (§5.2), incident severity (§5.3), incident distribution (§5.4), incidents by network design (§5.5), and switch reliability (§5.6).

5.1 Root Causes

- *Maintenance failure contribute the most documented incidents*
- 2× *more human errors than hardware errors*

Table 2 lists network incident root causes[3]. If a SEV has *multiple* root causes, we count the SEV toward multiple categories. If a SEV has *no* root causes, we count the SEV as *undetermined*. Human classification of root causes implies SEVs can be misclassified [53, 64]. While the rest of our analysis does *not* depend on the accuracy of root cause classification, we find it instructive to examine the types of root causes that occur in the networks of a large web service.

We find the root cause of 29% of network incidents is *undetermined*. We observe these SEVs correspond typically to transient and isolated incidents where engineers only reported on the incident's symptoms. Wu et al. noted a similar fraction of unknown issues (23%, [75], Table 1), while Turner et al. had a smaller, but higher fidelity, set of data (5%, [74], Table 5).

Maintenance failures contribute the most documented root causes (17%). This suggests that in the network infrastructure of a large web service provider like Facebook, despite the best efforts to automate and standardize the maintenance procedures, maintenance failures will still occur and lead to disruptions. Therefore, it is important to build mechanisms for quickly and reliably routing around faulty devices or devices under maintenance.

Hardware failures represent 13% of the root causes, while human-induced software issues – bugs and misconfiguration – occur at nearly double the rate of those caused by hardware failures. Prior studies the report hardware incident rate, such as Turner et al. [74] and Wu et al. [75], observe incident rates within 7% of us, suggesting hardware incidents remain a fundamental. Misconfiguration causes as many incidents as faulty hardware, something outside of operator control. This corroborates the findings of prior work that report misconfiguration as a large source of network failures in data centers [14, 15, 33, 47, 49, 62, 75], and shows the potential for emulation, verification, and more extensive automated repair based approaches to reduce the number of incidents [11–13, 25, 26, 28, 45, 47].

Compared to prior work, we corroborate the lower rate of configuration-related incidents in Turner et al. [74], Table 5 (9%), but contrast with Wu et al. [75], Table 1, who found configuration incidents dominate (38%). We suspect network operators play a large role in determining how configuration causes network incidents. At Facebook for example, all configuration changes require code review and typically get tested on a small number of switches before being deployed to the fleet. These practices may contribute to the lower misconfiguration incident rate we observe compared to Wu et al..

A potpourri of accidents and capacity planning makes up the last 15% of incidents (cf. Table 2). This is a testament to the many sources of entropy in large-scale production data center networks. Designing network devices to tolerate all of these issues in practice is prohibitively difficult (if not impossible). Therefore, one reliability engineering principle is to prepare for the unexpected in large-scale data center networks.

Figure 2 further illustrates the root cause breakdown across different types of network devices. Note that the major root cause categories, including undetermined, maintenance, configuration, hardware, and bugs, have relatively even representation across all types of network devices. Some root cause categories are represented unequally among devices. This is because there are not many incidents with these root causes, leading to too small population to be statistically meaningful. For example, ESWs, which represent a smaller population size compared to SSWs, FSWs, and RSWs, do not have SEVs with a "bug" root cause. In fact, ESWs run the same Facebook Open Switching System (FBOSS)-based software stack [5, 69]

[3]We use Govindan et al. [33]'s definition of *root cause*: "A failure event's root-cause is one that, if it had not occurred, the failure event would not have manifested."

as SSWs and FSWs, which do have network incidents caused by bugs.

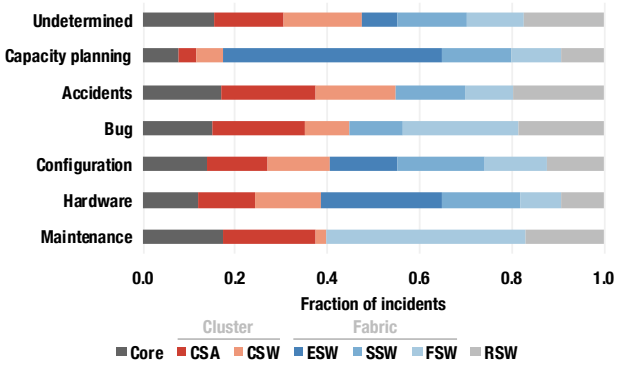

Figure 2: Root cause distribution of production incidents by network device type.

We conclude that among the failures that cannot be remediated by software, we observe that most failures involve maintenance, faulty hardware, and misconfiguration as devices and routing configurations become more complex and challenging to maintain.

5.2 Incident Rate

- *Higher incident rates on higher bandwidth devices*
- *Lower incident rates on fabric devices*
- *Better repair practices lower incident rates*

The reliability of data center networks is determined by the reliability of each interconnected network device. To measure the frequency of incidents related to each device type, we define *incident rate* of a device type as $r = \frac{i}{n}$, where i denotes the number of incidents caused by this type of network device and n is the number of active devices of that type (the *population*). Note that the incident rate could be larger than 1.0, meaning that each device of the target type caused more than one network incident on average. Figure 3 shows the incident rate of each type of network device in Facebook's data centers over the seven-year span. From Figure 3 we draw four key observations.

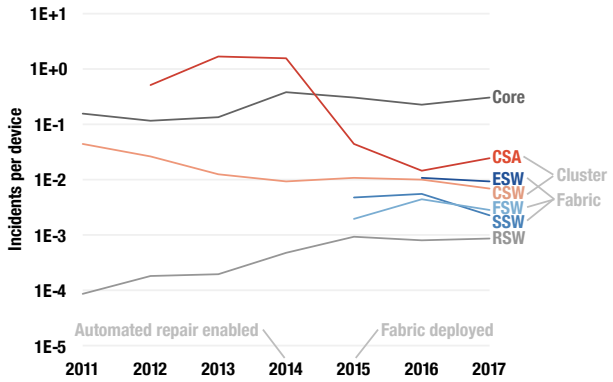

Figure 3: Rate of incidents per year for each device type (normalized to the number of active devices of each type) Note that the y axis scale is logarithmic and some devices have an incident rate of 0, e.g., if they did not exist in the fleet in a year.

First, *network devices with higher bisection bandwidth (e.g., Cores and CSAs in Figure 1) generally have higher incident rates, in comparison to the devices with lower bisection bandwidth (e.g., RSWs).* Intuitively, devices with higher bisection bandwidth tend to affect a larger number of connected devices and are thus correlated with more widespread impact when these types of devices fail. The annual incidence rate for ESWs, SSWs, FSWs, RSWs, and CSWs in 2017 is less than 1%.

These reliability characteristics largely influence our fault-tolerant data center network design. For example, we currently provision eight Cores in each data center, which allows us to tolerate one unavailable Core (e.g., if it must be removed from operation for maintenance) without any impact on the data center network. Note that nearly all of the Cores and CSAs are third-party vendor switches. In principle, if we do *not* have direct control of the proprietary software on these devices, the network design and implementation must take this into consideration.[4] For example, it may be more challenging to diagnose, debug, and repair switches that rely on firmware whose souce code is unavailable. In these cases it may make sense to increase the redundancy of switches in case some must be removed for repair by a vendor.

Second, *devices in the fabric network design (including ESWs, SSWs, and FSWs) have much lower incident rates compared to those devices in the cluster design (CSAs and CSWs).* There are two differences between the fabric network devices and the cluster devices: (a) the devices deployed in data center fabric are built from commodity chips [4, 5], while devices in the cluster network are purchased from third-party vendors and (b) the fabric network employs automated remediation software to handle common sources of failures [65].

Third, the fact that fabric-based devices are less frequently associated with failures verifies that fabric-based data center network, equipped with automated failover and remediation, is more resilient to device failures. Specifically, we can see excessive CSA-related incidents during 2013 and 2014, where the number of incidents

[4] Facebook has been manufacturing customized RSWs and modular switches since 2013. Please refer to the details in [4–6, 69].

exceeds the number of CSAs (with the incident rate as high as 1.7× and 1.5×, respectively). These high incidence rates were motivation to transition from the cluster network to fabric network.

Fourth, the CSA-related incident rate decreased in 2015, while the Core-related incident rate has generally increased from pre-2015 levels. This trend can be attributed to three causes: (1) the decreasing size of the CSA population, (2) the decreased development of their software as they were being replaced by newer fabric networks, and (3) new repair practices that were adopted around the time. For example, prior to 2014, network device repairs were often performed *without* draining the traffic on their links. This meant that in the worst case, when things went wrong, maintenance could affect a large volume of traffic. Draining devices prior to maintenance provided a simple but effective means to limit the likelihood of repair affecting production traffic. In addition, the data center fabric, with its more resilient design with higher path diversity, has been proved to be more amenable to automated remediation.

Network devices with higher bisection bandwidth have a higher likelihood of affecting service-level software and that network devices built from commodity chips have much lower incident rates compared to devices from third-party vendors due to the ease of integration with automated failover and remediation software.

5.3 Incident Severity

- *Core devices have the most incidents, but low severity*
- *Fewest incidents on fabric devices*
- *Lowest severity on fabric devices*

Not all incidents are created equal. Facebook classifies incidents into three severity levels from SEV3 (lowest severity) to SEV1 (highest severity). A SEV level reflects the high water mark for an incident. A SEV's level is never downgraded to reflect progress in resolving the SEV. Table 3 provides examples of incidents for each SEV level.

Level	Incident Examples
SEV3	Redundant or contained system failures, system impairments that do not affect or only minimally affect customer experience, internal tool failures.
SEV2	Service outages that affect a particular Facebook feature, regional network impairment, critical internal tool outages that put the site at risk.
SEV1	Entire Facebook product or service outage, data center outage, major portions of the site are unavailable, outages that affect multiple products or services.

Table 3: SEV levels and incident examples.

Figure 4 shows how all the network-related SEVs in 2017 were distributed among network devices. We draw four conclusions from Figure 4 that complement our raw incident rate findings from §5.2:

(1) While Core devices having the highest number of SEVs, their impact is typically low, with around 81% of SEVs being level 3, 15% being level 2, and 4% being level 1. RSWs have nearly as many incidents as Core network devices, with impact distributed in roughly the same proportion (85%, 10%, and 5% for SEV levels 3, 2, and 1, respectively).
(2) Data center fabric network devices have the lowest number of incidents among all network devices. ESWs account for 3% of SEVs, SSWs for 2%, and FSWs for 8%.
(3) Compared to cluster based data center network devices (CSAs and CSWs), fabric network devices typically have lower impact, with 66% fewer SEV1s, 33% *more* SEV2s (though the overall rate is still relatively low), and 52% fewer SEV3s.

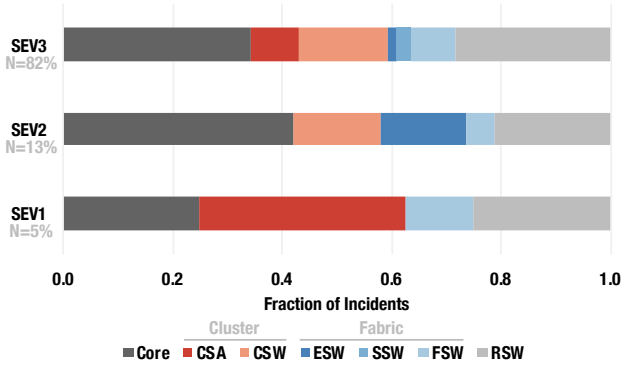

Figure 4: The distribution of SEVs and their level among different network devices in 2017.

Figure 5 shows how the rate of each SEV level has changed over the years, normalized to the total number of devices in the population during that year. While we do not disclose the absolute size of the population, we note that it is orders of magnitude larger than similar studies, such as Turner et al. [74]. The main conclusion we draw from Figure 5 is that the overall rate of SEVs per device had an inflection point in 2015, corresponding to the deployment of fabric data center networks. This was a significant turnaround, as prior to 2015, the rate of SEV3s grew at a nearly exponential rate.

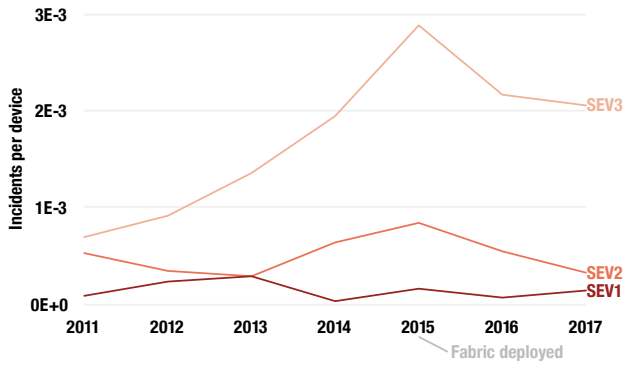

Figure 5: The number of network-related SEVs over time normalized to the number of deployed network devices.

We wondered whether more engineers working on network devices led to more SEVs. To test this theory, we used publicly

available data [71] to plot the yearly rate of SEVs per employee, our best proxy for engineers, working at Facebook, and found that the trends resembled those in Figure 5. To understand why, we plot the normalized number of switches at Facebook versus the number of Facebook employees in Figure 6. Switches grew in proportion to employees. We conclude that for our study, *the number of engineers working on network devices does not correlate with an increase in network device failures.*

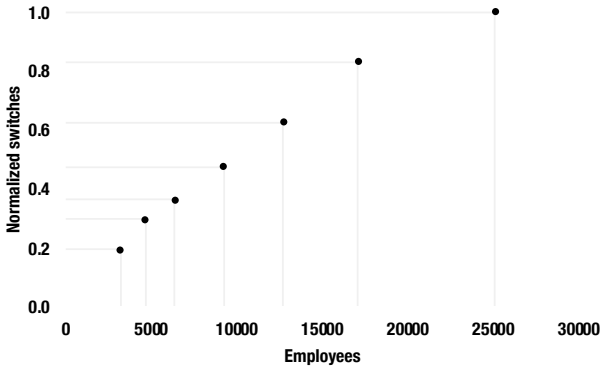

Figure 6: The normalized number of switches at Facebook versus the number of Facebook employees.

5.4 Incident Distribution

- *Stable incident rates for fabric devices over time*
- *RSW incident rates increasing over time*
- *Most incidents on high bandwidth or low redundancy devices*

The incident rate measures the frequency of the incidents induced by each device type. However, it does *not* reflect the overall amount of the network incidents that need to be handled in the field. Figure 7 shows the distribution of incidents caused by each type of network device on a yearly basis. We can see that network devices specific to the cluster network (CSAs and CSWs) have smaller proportion of incidents over time, as more and more fabric-based data centers are built and turned up (we analyze this trend in Section 5.5).

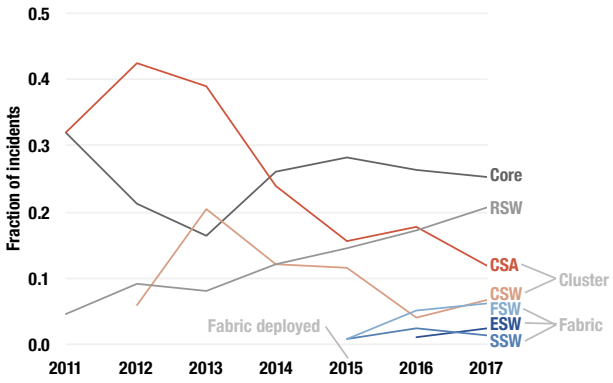

Figure 7: Fraction of network incidents per year broken down by device type.

Interestingly, we can observe that devices in the fabric network have not demonstrated any large increases in incidents over time. This again suggests that fabric-based data center topologies with automated failover provide good fault tolerance. On the other hand, RSW-related incidents have been steadily increasing over time (a finding that corroborates Potharaju et al. [62, 63]). This is partially driven by an increase in the size of the rack population over time. In addition, this is also a result of Facebook's data center network design, where we use only one single RSW as the Top-Of-Rack (TOR) switch. Some other companies, especially cloud service providers and enterprises, typically design with two TORs and have each server connected to both TORs for redundancy. At Facebook's scale, we find that it is more cost-effective to handle RSW failures in software using replication and distribution of server resources than to use redundant RSWs in every rack.

We also plot the relative number of incidents induced by each device type in Figure 8, using a fixed baseline, the total number of SEVs in 2017. There is a general increase in number of incidents across *all* device types until 2015 (with incidents due to some devices like FSWs and ESWs continuing to grow). The increasing number of incidents can be attributed to the rapid growth of Facebook's data center infrastructure: from 2011 to 2017 the total number of network device SEVs increased by 9.4×.

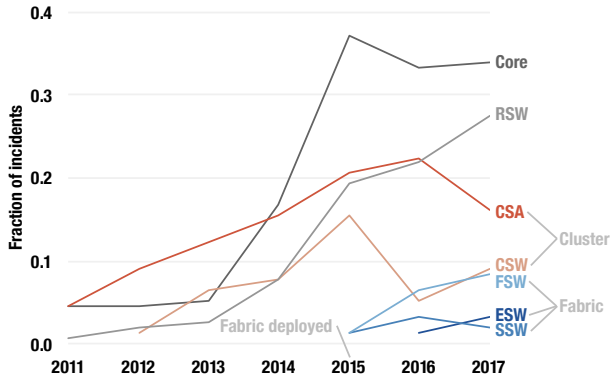

Figure 8: The number of incidents per year for different network device types, normalized to a fixed baseline, the total number of SEVs in 2017.

We find that rack switches contribute to around 28% of the service-level incidents that occurred in the past year (2017). This is due to the sheer size of the rack switch population (an affect we examine in §5.5), which leads to a noticeable reliability effect on the software systems running in data centers. In addition, we find that Core network devices that connect data centers and the backbone contribute to around 34% of service-level incidents compared to other network devices. This is because network devices with higher bisection bandwidth tend to affect a larger number of connected downstream devices and are thus correlated with widespread impact when these types of devices fail.

5.5 Incidents by Network Design

- *Cluster device incidents scale super-linearly*

- *Lower incidents per device for fabric devices*
- *Half the fabric device incidents versus cluster*

As revealed in §5.2 and §5.4, data center topologies play an important role on the network reliability. Figure 9 shows how the proportion of network incidents from different network topologies has changed over time. The proportion is calculated by aggregating network incidents across *all* of the devices that make up the network design, so every data point in Figure 9 is normalized to the same common baseline. For the cluster network, the devices are the CSAs and the CSWs. For the fabric network, the devices are the ESWs, the SSWs, and the FSWs. We make three major findings.

First, note that fabric topologies have been adopted steadily since 2015. Hence, an *inflection point in network incidents of the cluster network occurs with the introduction of the fabric network* in 2015. This corresponds to when Facebook began to replace classic networks with data center fabric networks. Similarly, fabric networks have seen an increase in incident rates over time.

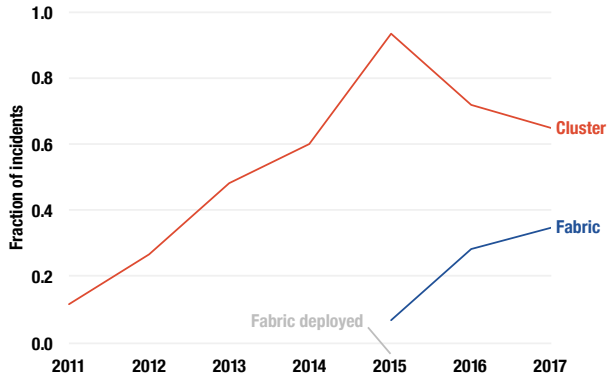

Figure 9: Number of incidents for each network design type normalized to a fixed baseline, the total number of SEVs in 2017.

Figure 10 shows another way to understand this trend: by comparing the incidents for each network type to the size of its respective population of devices (i.e., incidents per device). What we see is that for classic cluster network devices, incidents have scaled super-linearly with population size until around 2014, when it became challenging to made additional reliability improvements to the classic cluster network design. Since its introduction in 2015, the fabric based network design has consistently had lower incidents per device compared to the classic cluster network design.

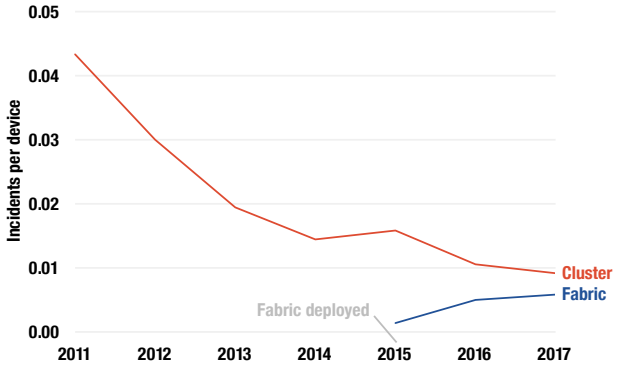

Figure 10: Incidents for each network design type normalized to the size of their respective populations over time.

Second, in the last year (2017), the number of incidents for data center fabric devices was around 50% of that of classic cluster devices. While the proportion of incidents from fabric topologies has increased over time, the number of incidents per device has remained relatively low compared to classic network topologies (as Figure 3 showed). Intuitively, this is because the software managed fault tolerance and automated remediation provided by fabric network topologies can *mask* some failures which could otherwise cause incidents in classic network topologies.

Third, we can see that since 2015, as the fabric network has been adopted more widely and the cluster network has stopped being deployed, the incident rates for the network devices associated with each type of network design show a corresponding trend. We also plot the population breakdown of device types deployed in Facebook's data centers over the seven-year span in Figure 11. Aside from illustrating the proliferation of RSWs in the fleet, it shows that an inflection point occurs in 2015, when the population of CSWs and CSAs begin to decrease and the population of FSWs, SSWs, and ESWs begins to increase. This is due to the adoption of data center fabric topologies across more Facebook data centers.

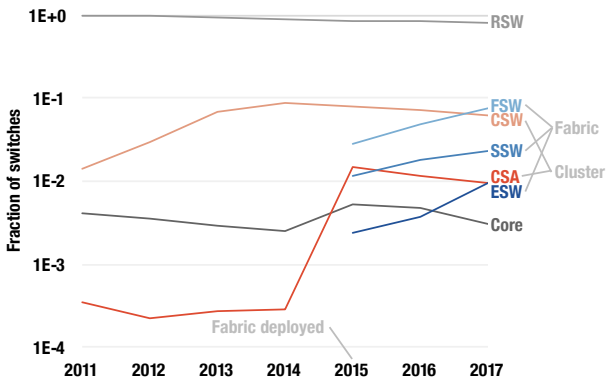

Figure 11: Population breakdown by network device type over the 7-year span.

We conclude that, compared to the cluster network, data center fabric networks contributed to around 50% of the number of intra data center network incidents in the past year (2017). We find that data center fabric networks are more reliable due to their simpler, commodity-chip based switches and automated remediation software that dynamically adapts to and tolerate device failures.

5.6 Switch Reliability

- *Failure rate varies by three orders of magnitude across switch types*
- *Fabric switches fail 3.2× less frequently than cluster switches*
- *Larger networks increase incident resolution time*

We analyze last the reliability of Facebook data center switches. We use SEV timing data to measure *mean time between incidents (MTBI)* and *75th percentile (p75) incident resolution time (p75IRT)*. p75IRT deserves additional explanation. Engineers at Facebook document *resolution time*, not *repair time*, in a SEV. Resolution time exceeds repair time and includes time engineers spend on prevention. To prevent occasional months-long incident recovery times from dominating the mean, we examine the 75th percentile incident resolution time.

MTBI. We measure the average time between the start of two consecutive incidents for MTBI. Figure 12 plots MTBI for each switch type by year. We draw three conclusions from the data.

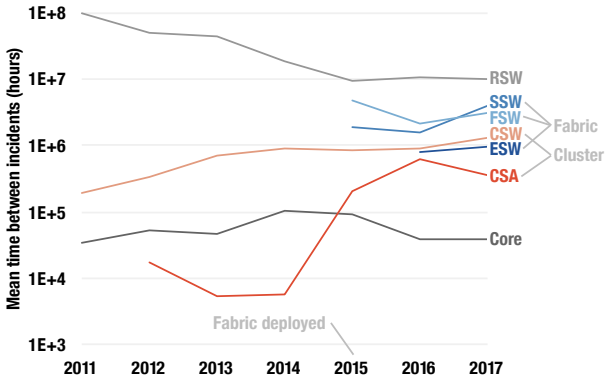

Figure 12: Mean time between incidents in hours for different network device types. Note that the y axis is in logarithmic scale.

First, we find over the seven years from 2011 to 2017, MTBI did not change more than 10× across each switch type, except CSAs. In 2015, in response to frequent CSA maintenance incidents, engineers strengthened CSA operational procedure guidelines, adding checks to ensure operators drained CSAs before performing maintenance, for example. These operational improvements increased CSA MTBI by two orders of magnitude between 2014 and 2016.

Second, we find in 2017, *MTBI varied by three orders of magnitude across switch types* from 39 495 device-hours for Cores to 9 958 828 device-hours for RSWs. If we compare MTBI to switch type population size in 2017 in Figure 11, we find devices with larger population sizes tend to have larger MTBIs. This is not mere correlation. Facebook *designs* its switches to ensure their rate of failure does not overwhelm engineers or automated repair systems.

Third, we find in 2017, *fabric network switches fail 3.2× less frequently than cluster network switches*, with an average MTBI of 2636818 device-hours compared to 822518 device-hours. We attribute the higher reliability of fabric switches to their simple switches, software management, and automated repairs.

p75IRT. We measure the average time between the start and the resolution of incidents for p75IRT. Figure 13 plots p75IRT for each switch type by year.

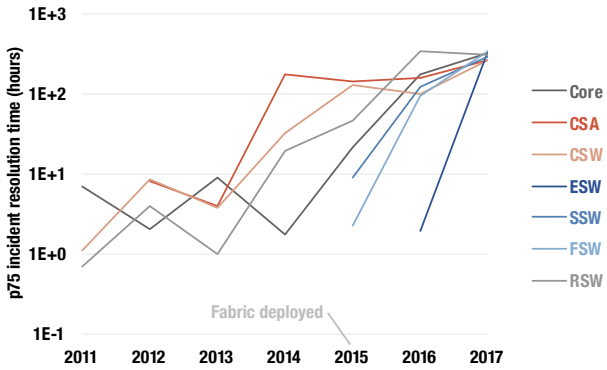

Figure 13

We find over the seven years from 2011 to 2017, p75IRT increased similarly across switch types. The increase happened without changes to individual switch design, operation, and management. To explain the overall increase in p75IRT, we plot p75IRT versus the normalized number of switches at Facebook in Figure 14.

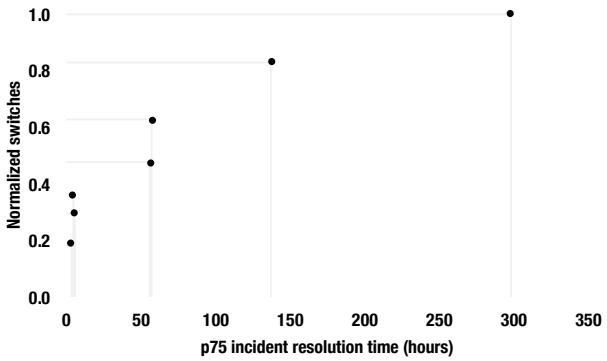

Figure 14

We observe a positive correlation between p75IRT and number of switches. At Facebook, we find *larger networks increase the time humans take to resolve network incidents*. We attribute part of the increased resolution time to more standardized processes for releasing fixes to production. Today, switch configuration and software changes go through more thorough review processes, testing, and deployment than in the past. We observe the additional time spent in resolution contributes toward more reliable fixes and reduces the likelihood of repeat incidents.

5.7 Intra Data Center Reliability Implications

Software-based failure tolerance techniques for data center networks. At Facebook's scale, we have made an effort to spend the time to develop software systems that are more tolerant to network hardware failures (featuring data replication and failover techniques) than to over-provision networking devices, such as rack switches, which make up the majority of Facebook's network device fleet. We are interested in understanding how we can move from designing defensive software techniques that treat the network as a black box with potentially unknown failure modes toward *co-designing* software alongside network infrastructure. This will involve making software aware of the reliability characteristics of the network, such as those we have reported in this study.

Software* assisted *networks as an alternative to software* defined *networks. At Facebook we have attempted to strike the right balance between network reliability and software complexity in our infrastructure. Instead of shouldering the burden of maintaining fully programmable network topologies, we have opted for simpler fabric-based topologies assisted by centralized automated remediation software. Yet, it is unclear how much automation should be employed, how this automation can be standardized across an industry, and how to provide the flexibility for large scale web service companies to easily deploy and configure it without relying on their own custom purpose hardware.

Preparing software systems for network incidents. We observe that it is challenging for software systems to prepare for network incidents. At Facebook, we run periodical tests, including both fault injection testing [9, 73] and disaster recovery testing [46, 66], to exercise the reliability of our production systems by simulating different types of network failures, such as device outages and disconnection of an entire data center. Yet despite these efforts, we still experience system-level issues caused by network incidents. We are exploring *proactive* solutions to help our production systems *anticipate* and defend against network incidents.

6 INTER DATA CENTER RELIABILITY

In this section, we study the reliability of backbone networks. We analyze network failures between Facebook's data centers over the course of eighteen months, from October 2016 to April 2018, comprising tens of thousands of real world events, comparable in size to Turner et al. [74] and three times longer than Wu et al. [75]. We analyze two types of backbone network failures:

- ***Link failures,*** where an individual bundle of optical fiber linking two edges (Figure 1, ⑤) fails.
- ***Edge failures,*** where multiple link failures cause an edge to fail. An edge connects to the backbone and Internet using at least three links. When all of an edge's links fail, the edge fails.

Our backbone network dataset does not contain root causes. We measure *mean time between failures (MTBF)* and *mean time to recovery (MTTR)* for edges and links. We analyze edge reliability (§6.1), link reliability by fiber vendor (§6.2), and edge reliability by geographic location (§6.3).

6.1 Edge Reliability

- *Typical edge failure rate is on the order of months*
- *Typical edge recovery rate is on the order of hours*
- *High variance in both edge MTBF and MTTR*

We first analyze the MTBF and MTTR of the edges in Facebook's backbone network. An edge fails when a combination of planned fiber maintenances or unplanned fiber cuts sever its backbone and Internet connectivity. An edge recovers when repairs restore its backbone and Internet connectivity.

MTBF. The solid line in Figure 15 plots edge MTBF in hours as a function of the percentage of edges with that MTBF or lower. Most edges fail infrequently because fiber vendors strive to maintain reliable links. 50% of edges fail less than once every 1710 h, or 2.3 months. And 90% of edges fail less than once every 3521 h, or 4.8 months.

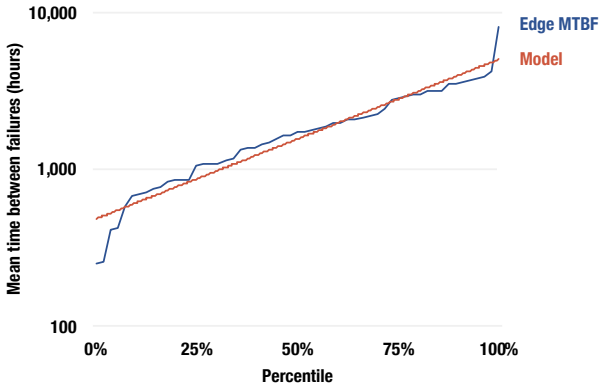

Figure 15: MTBF as a function of percentage of edges connecting Facebook data centers with that MTBF or lower.

Edges exhibit high variance in MTBF due to their diverse fiber vendor makeup and geographic locations (observations we explore in §6.2 and §6.3). The standard deviation of edge MTBF is 1320 h, with the least reliable edge failing on average once every 253 h and the most reliable edge failing on average once every 8025 h.

We model $MTBF_{edge}(p)$ as an exponential function of the percentage of edges, $0 \leq p \leq 1$, with that MTBF or lower. We built the models in this section by fitting an exponential function using the least squares method. At Facebook, we use these models in capacity planning to calculate *conditional risk*, the likelihood of edge or link being unavailable given a set of failures. We plan edge and link capacity to tolerate the 99.99th percentile of conditional risk. We find that $MTBF_{edge}(p) = 462.88e^{2.3408p}$ (the dotted line in Figure 15) with $R^2 = 0.94$.

MTTR. The solid line in Figure 16 plots edge MTTR in hours as a function of the percentage of edges with that MTTR or lower. Edge recovery occurs much faster than the time between outages because edges contain multiple links (at least three) and fiber vendors work to repair link failures rapidly. 50% of edges recover within 10 h of a failure; 90% within 71 h.

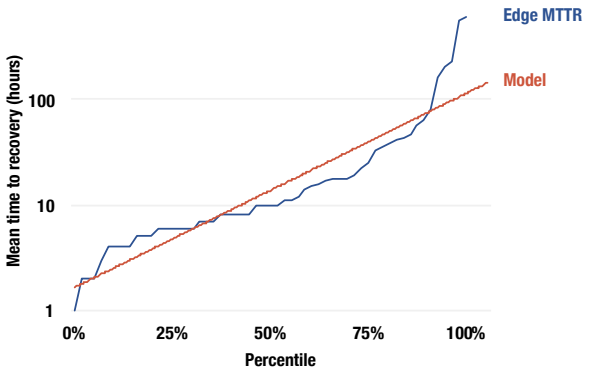

Figure 16: MTTR as a function of percentage of edges connecting Facebook data centers with that MTTR or lower.

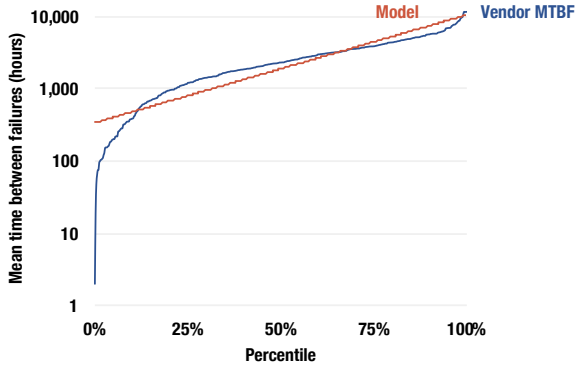

Figure 17: MTBF as a function of percentage of fiber vendors with that MTBF or lower.

Edges exhibit high variance in MTTR because some edges are easier to repair than others. Imagine the differences between an edge on a remote island compared to an edge in a big city. Weather conditions, physical terrain, and travel time affect the time it takes a fiber vendor to repair an edge's links. The standard deviation of edge MTTR is 112 h, with the slowest edge to recover taking on average 608 h and the fastest edge to recover taking 1 h.

We model $MTTR_{edge}(p)$ as an exponential function of the percentage of edges, $0 \leq p \leq 1$ with that MTTR or lower. We find that $MTTR_{edge}(p) = 1.513e^{4.256p}$ (the dotted line in Figure 15) with $R^2 = 0.87$.

The high variances in edge MTBF and MTTR motivate us to study the reliability characteristics of the links connecting edges in §6.2 and the geographic location of edges in §6.3.

6.2 Link Reliability by Fiber Vendor

- *Typical vendor link failure rate is on the order of months*
- *Higher MTBF in high competition markets*
- *Vendor MTBF and MTTR each span multiple orders of magnitude*

We next analyze the MTBF and MTTR for fiber vendors based on when the links they operate fail or recover. For brevity, we shorten "the MTBF/MTTR of the links operated by a fiber vendor" to "fiber vendor MTBF/MTTR."

MTBF. The solid line in Figure 17 plots the fiber vendor MTBF in hours as a function of the percentage of fiber vendors with that MTBF or lower. For most vendors, link failure happens only occasionally due to regular maintenance and monitoring. 50% of vendors have at least one link failure every 2326 h, or once every 3.2 months. And 90% of vendors have at least one link failure every 5709 h.

Fiber vendor MTBF varies by orders of magnitude. The standard deviation of fiber vendor MTBF is 2207 h, with the least reliable vendor's links failing on average once every 2 h and the most reliable vendor's links failing on average once every 11 721 h. Anecdotally, we observe that fiber markets with high competition lead to more incentive for fiber vendors to increase reliability. For example, the most reliable vendor operates in a big city in the USA.

MTTR. The solid line in Figure 18 plots fiber vendor MTTR as a function of the percentage of fiber vendors with that MTTR or lower. Most vendors repair links promptly. 50% of vendors repair links within 13 h of a failure; 90% within 60 h.

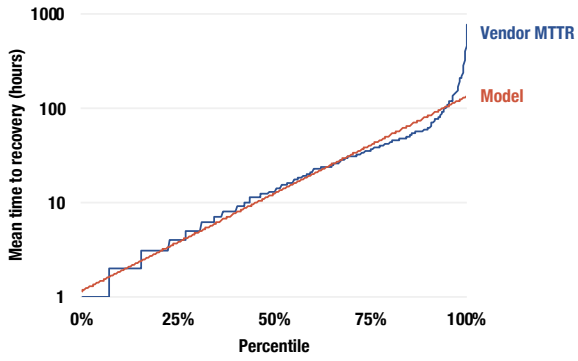

Figure 18: MTTR as a function of percentage of fiber vendors with that MTTR or lower.

Fiber vendors exhibit high variance in MTTR because some fiber vendors operate in areas where they can more easily repair links (an observation we analyze in §6.3). The standard deviation of fiber vendor MTTR is 56 h, with the slowest vendor taking on average 744 h to repair their links and the most reliable vendor taking on average 1 h to repair their links.

We model $MTTR_{vendor}(p)$ as an exponential function of the percentage of vendors, $0 \leq p \leq 1$ with that MTTR or lower. We find that $MTTR_{vendor}(p) = 1.1345e^{4.7709p}$ (the dotted line in Figure 18) with $R^2 = 0.98$.

We conclude that *not all fiber vendors operate equally*. We next explore how the geographic location of edges affects their reliability.

6.3 Edge Reliability by Geographic Location

- *Edge failure rate is similar across most continents*
- *Edges recover within* 1 d *on average on all continents*

We analyze last the reliability of edges by their geographic location using the continent they reside on. Table 4 shows the distribution of edges in Facebook's network among continents. Most edges reside in North America, followed closely by Europe. The continents with the fewest edges are Africa and Australia.

Table 4: The distribution and reliability of edges in Facebook's network among continents.

Continent	Distribution	MTBF (hours)	MTTR (hours)
North America	37%	1848	17
Europe	33%	2029	19
Asia	14%	2352	11
South America	10%	1579	9
Africa	4%	5400	22
Australia	2%	1642	2

MTBF. We show the average MTBF for the edges in each continent in Table 4. Edges in Africa are outliers, with an average MTBF of 5400 h, or 7.4 months. Edge reliability in Africa is important because edges in Africa are few and connect Europe and Asia. Edges in North America, South America, Europe, Asia, and Australia have average MTBFs ranging from 1579 h (2.2 months, for South America) to 2352 h (3.2 months, for Asia).

MTTR. We show the average MTTR for the edges in each continent in Table 4. Across continents, edges recover within 1 d on average. Edges in Africa, despite their long uptime, take the longest time on average to recover at 22 h due to their submarine links. Edges in Australia take the shortest time on average to recover at 2 h due to their locations in big cities. We observe a 7 h standard deviation in edge MTTR among continents.

6.4 Inter Data Center Reliability Implications

Extending automated remediation to the edge. While Facebook can use software automation to improve the reliability of networks *within* our data centers, we see providing similar automation to the backbone networks that connect our data centers as a key challenge. Given that we often do *not* have full control of the devices along the links that connect the edges, coming up with a standard protocol and an automated procedure to enabling this type of interaction or designing generic remediation systems are an important focus.

Coordinated understanding of shared backbone resources. We reported on the reliability characteristics of the backbone links that Facebook operates on, yet in many cases, Facebook is only one of the many entities that share backbone link connectivity. Despite the shared fate of these links, relatively little openly accessible data is available on their reliability characteristics. While we have characterized the links that Facebook operates on, we hope that our work inspires other practitioners throughout the community to contribute to the body of knowledge available on this less understood aspect of web service operation.

7 RELATED WORK

To our knowledge, this paper provides the first comprehensive study of network incidents from the perspective of large-scale web services. Prior large-scale data center failure studies [8, 16, 35, 60] report that network incidents are among the major causes of web service outages; however, none of these studies systematically analyze network incidents at a large scale, focusing on the availability of an entire web service, across both inter and intra data center networks, in a long term, longitudinal study.

There are a number of prior studies examined the failure characteristics of network links and devices in different types of networks studied in this paper, including both data center networks [30, 62, 63, 77] and optical backbones [29, 51, 63]. Specially, Potharaju and Jain [63] and Turner et al. [74] also studies data center network infrastructure by characterizing device/link failures in both intra and inter data center networks. Their studies also characterize the failure impact, including connectivity losses, high latency, package drops, and so on. These studies significantly boost the understanding of network failure characteristics, and provide insights for network engineers and operators to improve the fault tolerance of existing networks and to design more robustness networks.

While our work is closely related to these prior studies, it is also *fundamentally* different and *complementary* in the following three aspects. First, our work has a different goal. Unlike these prior studies that focus on understanding fine-grained per-device, per-link failures and their impact to the system-level services above the network stack, our work focuses on how network incidents affect *the availability of the Internet service*. Our goal is to reveal and quantify the incidents that *cannot* be tolerated despite industry best practices, and shed light on how large scale systems operate reliably in the face of these incidents. Second, prior studies only cover the data center and backbone networks with traditional Clos-based architectures, whereas our work presents a comparative study of the reliability characteristics of data center network infrastructure with *both* a traditional Clos-based design and a contemporary fabric-based design with smaller, merchant-silicon-based switches. As introduced in §3, we achieve this due to the heterogeneity of the data center network infrastructure of Facebook where networks with different designs co-exist and co-operate. Third, we present a long-term (seven years for intra data center networks and eighteen months for inter data center networks) longitudinal analysis to reveal the evolution of network reliability characteristics, while prior studies typically provide only aggregated results, often over a much shorter period or with orders of magnitude fewer switches [74].

Govindan et al. [33] study over 100 failure events at Google WAN and data center networks, offering insights on why maintaining high levels of availability for content providers is challenging. Their study, similar to [8, 16, 35, 60], focuses on network management and the design principles for building robust networks. Many of the high-level design principles mentioned in [33], such as using multiple layers of fallback (defense in depth), continuous prevention,

and recovering fast, are applicable for large-scale software systems to protect against network incidents.

8 CONCLUSIONS

At Facebook, we strive to maintain a reliable network infrastructure both within (intra) and between (inter) data centers. In this study, we have characterized the network incidents we observe and their behavior on the software systems that run on them. We have also analyzed the backbone links that connect data centers and modeled their reliability characteristics. Our analysis revealed several key observations about the networks within and between data centers.

As software systems grow in complexity, interconnectedness, and geographic distribution, unwanted behavior from network infrastructure has the potential to become a key limiting factor in the ability to reliably operate distributed software systems at a large scale. To ameliorate the negative effects of network errors on service reliability, we have highlighted several promising implications for future network research.

We look forward to future research that sheds light on some of these important directions. It is our hope that the research community can build upon this study to better characterize, understand, and improve the reliability of data center networks and systems.

ACKNOWLEDGEMENTS

We would like to thank Boliu Xu, Alexander Nikolaidas, James Zeng, Jimmy Williams, Omar Baldonado, and Hans Ragas for their feedback and suggestions while writing this paper.

REFERENCES

[1] Al-Fares, M., Loukissas, A., and Vahdat, A. A Scalable, Commodity Data Center Network Architecture. In *Proceedings of the 2008 ACM SIGCOMM Conference (SIGCOMM'08)* (Seattle, WA, USA, 2008).

[2] Alizadeh, M., Greenberg, A., Maltz, D. A., Padhye, J., Patel, P., Prabhakar, B., Sengupta, S., and Sridharan, M. Data Center TCP (DCTCP). In *Proceedings of the 2010 ACM SIGCOMM Conference (SIGCOMM'10)* (New Delhi, India, 2010).

[3] Andreyev, A. Introducing data center fabric, the next-generation Facebook data center network. https://code.facebook.com/posts/360346274145943/introducing-data-center-fabric-the-next-generation-facebook-data-center-network/, Nov. 2014.

[4] Bachar, Y. Introducing "6-pack": the first open hardware modular switch. https://code.facebook.com/posts/843620439027582/facebook-open-switching-system-fboss-and-wedge-in-the-open/, Feb. 2015.

[5] Bachar, Y., and Simpkins, A. Introducing "Wedge" and "FBOSS," the next steps toward a disaggregated network. https://code.facebook.com/posts/681382905244727/introducing-wedge-and-fboss-the-next-steps-toward-a-disaggregated-network/, June 2014.

[6] Bagga, J., and Yao, Z. Open networking advances with Wedge and FBOSS. https://code.facebook.com/posts/145488969140934/open-networking-advances-with-wedge-and-fboss/, Nov. 2015.

[7] Bailis, P., and Kingsbury, K. The Network is Reliable: An informal survey of real-world communications failures. *Communications of the ACM (CACM) 57*, 9 (Sept. 2014), 48–55.

[8] Barroso, L. A., Clidaras, J., and Hölzle, U. *The Datacenter as a Computer: An Introduction to the Design of Warehouse-scale Machines*, 2 ed. Morgan and Claypool Publishers, 2013.

[9] Basiri, A., Behnam, N., de Rooij, R., Hochstein, L., Kosewski, L., Reynolds, J., and Rosenthal, C. Chaos Engineering. *IEEE Software 33*, 3 (May 2016), 35–41.

[10] Beaver, D., Kumar, S., Li, H. C., Sobel, J., and Vajgel, P. Finding a Needle in Haystack: Facebook's Photo Storage. In *Proceedings of the 9th USENIX Conference on Operating Systems Design and Implementation (OSDI'10)* (Vancouver, BC, Canada, 2010).

[11] Beckett, R., Gupta, A., Mahajan, R., and Walker, D. A General Approach to Network Configuration Verification. In *Proceedings of the 2017 ACM SIGCOMM Conference (SIGCOMM'17)* (Los Angeles, CA, USA, 2017).

[12] Beckett, R., Mahajan, R., Millstein, T., Padhye, J., and Walker, D. Don't Mind the Gap: Bridging Network-wide Objectives and Device-level Configurations. In *Proceedings of the 2016 ACM SIGCOMM Conference (SIGCOMM'16)* (Florianópolis, Brazil, 2016).

[13] Beckett, R., Mahajan, R., Millstein, T., Padhye, J., and Walker, D. Network Configuration Synthesis with Abstract Topologies. In *Proceedings of the 38th ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI'17)* (Barcelona, Spain, 2017).

[14] Benson, T., Akella, A., and Maltz, D. Unraveling the Complexity of Network Management. In *Proceedings of the 6th USENIX Symposium on Networked Systems Design and Implementation (NSDI'09)* (Boston, MA, USA, 2009).

[15] Benson, T., Akella, A., and Shaikh, A. Demystifying Configuration Challenges and Trade-offs in Network-based ISP Services. In *Proceedings of the 2011 ACM SIGCOMM Conference (SIGCOMM'11)* (Toronto, ON, Canada, 2011).

[16] Brewer, E. Spanner, TrueTime and the CAP Theorem. Tech. rep., Google Inc., Feb. 2017. https://static.googleusercontent.com/media/research.google.com/en//pubs/archive/45855.pdf.

[17] Bronson, N., Amsden, Z., Cabrera, G., Chakka, P., Dimov, P., Ding, H., Ferris, J., Giardullo, A., Kulkarni, S., Li, H. C., Marchukov, M., Petrov, D., Puzar, L., Song, Y. J., and Venkataramani, V. TAO: Facebook's Distributed Data Store for the Social Graph. In *Proceedings of the 2013 USENIX Annual Technical Conference (USENIX ATC'13)* (San Jose, CA, June 2013).

[18] Chen, G. J., Wiener, J. L., Iyer, S., Jaiswal, A., Lei, R., Simha, N., Wang, W., Wilfong, K., Williamson, T., and Yilmaz, S. Realtime Data Processing at Facebook. In *Proceedings of the 2016 ACM SIGMOD/PODS Conference (SIGMOD'16)* (San Francisco, CA, USA, 2016).

[19] Clos, C. A study of non-blocking switching networks. *The Bell System Technical Journal 32*, 2 (Mar. 1953), 406–424.

[20] Darrow, B. Superstorm Sandy wreaks havoc on internet infrastructure. https://gigaom.com/2012/10/30/superstorm-sandy-wreaks-havoc-on-internet-infrastructure/, 2012.

[21] Dean, J., and Ghemawat, S. MapReduce: Simplified Data Processing on Large Clusters. In *Proceedings of the 6th Symposium on Operating Systems Design and Implementation (OSDI'04)* (San Francisco, CA, USA, 2004).

[22] Evans, J. The HipHop Virtual Machine. https://code.facebook.com/posts/495167483903373/the-hiphop-virtual-machine/, Dec. 2011.

[23] Farmer, S. More On Gmail's Delivery Delays. https://cloud.googleblog.com/2013/09/more-on-gmails-delivery-delays.html, 2013. Google Cloud Official Blog.

[24] Farrington, N., and Andreyev, A. Facebook's Data Center Network Architecture. In *Proceedings of the 2013 IEEE Optical Interconnects Conference* (Santa Fe, New Mexico, May 2013).

[25] Fayaz, S. K., Sharma, T., Fogel, A., Mahajan, R., Millstein, T., Sekar, V., and Varghese, G. Efficient Network Reachability Analysis using a Succinct Control Plane Representation. In *Proceedings of the 12th USENIX Symposium on Operating Systems Design and Implementation (OSDI'16)* (Savannah, GA, USA, 2016).

[26] Fogel, A., Fung, S., Pedrosa, L., Walraed-Sullivan, M., Govindan, R., Mahajan, R., and Millstein, T. A General Approach to Network Configuration Analysis. In *Proceedings of the 12th USENIX Conference on Networked Systems Design and Implementation (NSDI'15)* (Oakland, CA, USA, 2015).

[27] Ford, D., Labelle, F., Popovici, F. I., Stokely, M., Truong, V.-A., Barroso, L., Grimes, C., and Quinlan, S. Availability in Globally Distributed Storage Systems. In *Proceedings of the 9th USENIX Conference on Operating Systems Design and Implementation (OSDI'10)* (Vancouver, BC, Canada, Oct. 2010).

[28] Gember-Jacobson, A., Akella, A., Mahajan, R., and Liu, H. H. Automatically Repairing Network Control Planes Using an Abstract Representation. In *Proceedings of the 26th Symposium on Operating Systems Principles (SOSP'17)* (Shanghai, China, 2017).

[29] Ghobadi, M., and Mahajan, R. Optical Layer Failures in a Large Backbone. In *Proceedings of the 2016 Internet Measurement Conference (IMC'16)* (Santa Monica, CA, USA, 2016), IMC '16.

[30] Gill, P., Jain, N., and Nagappan, N. Understanding Network Failures in Data Centers: Measurement, Analysis, and Implications. In *Proceedings of the 2011 ACM SIGCOMM Conference (SIGCOMM'11)* (Toronto, ON, Canada, 2011).

[31] Google Cloud Platform. Google Compute Engine Incident #16007. https://status.cloud.google.com/incident/compute/16007, 2016.

[32] Google Cloud Platform. Google Cloud Networking Incident #17002. https://status.cloud.google.com/incident/cloud-networking/17002, 2017.

[33] Govindan, R., Minei, I., Kallahalla, M., Koley, B., and Vahdat, A. Evolve or Die: High-Availability Design Principles Drawn from Googles Network Infrastructure. In *Proceedings of the 2016 ACM SIGCOMM Conference (SIGCOMM'16)* (Florianópolis, Brazil, Aug. 2016).

[34] Greenberg, A., Hamilton, J. R., Jain, N., Kandula, S., Kim, C., Lahiri, P., Maltz, D. A., Patel, P., and Sengupta, S. VL2: A Scalable and Flexible Data Center Network. In *Proceedings of the 2009 ACM SIGCOMM Conference (SIGCOMM'09)* (Barcelona, Spain, 2009).

[35] Gunawi, H. S., Hao, M., Suminto, R. O., Laksono, A., Satria, A. D., Adityatama, J., and Eliazar, K. J. Why Does the Cloud Stop Computing? Lessons from Hundreds of Service Outages. In *Proceedings of the 7th ACM Symposium on Cloud*

Computing (SoCC'16) (Santa Clara, CA, Oct. 2016).

[36] Guo, C., Lu, G., Li, D., Wu, H., Zhang, X., Shi, Y., Tian, C., Zhang, Y., and Lu, S. BCube: A High Performance, Server-centric Network Architecture for Modular Data Centers. In *Proceedings of the 2009 ACM SIGCOMM Conference (SIGCOMM'09)* (Barcelona, Spain, 2009).

[37] Guo, C., Wu, H., Tan, K., Shi, L., Zhang, Y., and Lu, S. DCell: A Scalable and Fault-tolerant Network Structure for Data Centers. In *Proceedings of the 2008 ACM SIGCOMM Conference (SIGCOMM'08)* (Seattle, WA, USA, 2008).

[38] Hong, C.-Y., Kandula, S., Mahajan, R., Zhang, M., Gill, V., Nanduri, M., and Wattenhofer, R. Achieving High Utilization with Software-driven WAN. In *Proceedings of the 2013 ACM SIGCOMM Conference (SIGCOMM'13)* (Hong Kong, China, 2013).

[39] Huang, Q., Ang, P., Knowles, P., Nykiel, T., Tverdokhlib, I., Yajurvedi, A., Dapolito, IV, P., Yan, X., Bykov, M., Liang, C., Talwar, M., Mathur, A., Kulkarni, S., Burke, M., and Lloyd, W. SVE: Distributed Video Processing at Facebook Scale. In *Proceedings of the 26th Symposium on Operating Systems Principles (SOSP'17)* (Shanghai, China, 2017).

[40] Jain, S., Kumar, A., Mandal, S., Ong, J., Poutievski, L., Singh, A., Venkata, S., Wanderer, J., Zhou, J., Zhu, M., Zolla, J., Hölzle, U., Stuart, S., and Vahdat, A. B4: Experience with a Globally-deployed Software Defined WAN. In *Proceedings of the 2013 ACM SIGCOMM Conference (SIGCOMM '13)* (Hong Kong, China, 2013).

[41] Jimenez, M., and Kwok, H. Building Express Backbone: Facebook's new long-haul network. https://code.facebook.com/posts/1782709872057497/building-express-backbone-facebook-s-new-long-haul-network/, May 2017.

[42] Joseph, D. A., Tavakoli, A., and Stoica, I. A Policy-aware Switching Layer for Data Centers. In *Proceedings of the 2008 ACM SIGCOMM Conference (SIGCOMM'08)* (Seattle, WA, USA, 2008).

[43] Kaldor, J., Mace, J., Bejda, M., Gao, E., Kuropatwa, W., O'Neill, J., Ong, K. W., Schaller, B., Shan, P., Viscomi, B., Venkataraman, V., Veeraraghavan, K., and Song, Y. J. Canopy: An End-to-End Performance Tracing And Analysis System. In *Proceedings of the 26th Symposium on Operating Systems Principles (SOSP'17)* (Shanghai, China, Oct. 2017).

[44] Kandula, S., Menache, I., Schwartz, R., and Babbula, S. R. Calendaring for Wide Area Networks. In *Proceedings of the 2014 ACM SIGCOMM Conference (SIGCOMM'14)* (Chicago, IL, USA, 2014).

[45] Khurshid, A., Zou, X., Zhou, W., Caesar, M., and Godfrey, P. B. VeriFlow: Verifying Network-Wide Invariants in Real Time. In *Proceedings of the 10th USENIX Conference on Networked Systems Design and Implementation (NSDI'13)* (Lombard, IL, USA, 2013).

[46] Krishnan, K. Weathering the Unexpected. *Communications of the ACM (CACM) 55*, 11 (Nov. 2012), 48–52.

[47] Liu, H. H., Zhu, Y., Padhye, J., Cao, J., Tallapragada, S., Lopes, N. P., Rybalchenko, A., Lu, G., and Yuan, L. CrystalNet: Faithfully Emulating Large Production Networks. In *Proceedings of the 26th Symposium on Operating Systems Principles (SOSP'17)* (Shanghai, China, 2017).

[48] Liu, V., Halperin, D., Krishnamurthy, A., and Anderson, T. F10: A Fault-Tolerant Engineered Network. In *Proceedings of the 10th USENIX Conference on Networked Systems Design and Implementation (NSDI'13)* (Lombard, IL, USA, 2013).

[49] Mahajan, R., Wetherall, D., and Anderson, T. Understanding BGP Misconfiguration. In *Proceedings of the ACM 2002 SIGCOMM Conference (SIGCOMM'02)* (Pittsburgh, PA, USA, 2002).

[50] Malewicz, G., Austern, M. H., Bik, A. J., Dehnert, J. C., Horn, I., Leiser, N., and Czajkowski, G. Pregel: A System for Large-scale Graph Processing. In *Proceedings of the 2010 ACM SIGMOD/PODS Conference (SIGMOD'10)* (Indianapolis, IN, USA, 2010).

[51] Markopoulou, A., Iannaccone, G., Bhattacharyya, S., Chuah, C.-N., Ganjali, Y., and Diot, C. Characterization of Failures in an Operational IP Backbone Network. *IEEE/ACM Transactions on Networking 16*, 4 (Aug. 2008), 749–762.

[52] Maurer, B. Fail at Scale: Reliability in the Face of Rapid Change. *Communications of the ACM (CACM) 58*, 11 (Nov. 2015), 44–49.

[53] Medem, A., Akodjenou, M.-I., and Teixeira, R. TroubleMiner: Mining Network Trouble Tickets. In *Proceedings of the 11th IFIP/IEEE International Symposium on Integrated Network Management (IM'09)* (New York, NY, USA, 2009).

[54] Mellette, W. M., McGuinness, R., Roy, A., Forencich, A., Papen, G., Snoeren, A. C., and Porter, G. RotorNet: A Scalable, Low-complexity, Optical Datacenter Network. In *Proceedings of the 2017 ACM SIGCOMM Conference (SIGCOMM'17)* (Los Angeles, CA, USA, 2017).

[55] Mogul, J. C. Emergent (Mis)behavior vs. Complex Software Systems. Tech. Rep. HPL-2006-2, HP Laboratories Palo Alto, DEC 2005. http://www.hpl.hp.com/techreports/2006/HPL-2006-2.pdf.

[56] Muralidhar, S., Lloyd, W., Roy, S., Hill, C., Lin, E., Liu, W., Pan, S., Shankar, S., Sivakumar, V., Tang, L., and Kumar, S. f4: Facebook's Warm BLOB Storage System. In *Proceedings of the 11th USENIX Symposium on Operating Systems Design and Implementation (OSDI'14)* (Broomfield, CO, USA, 2014).

[57] Niranjan Mysore, R., Pamboris, A., Farrington, N., Huang, N., Miri, P., Radhakrishnan, S., Subramanya, V., and Vahdat, A. PortLand: A Scalable Fault-tolerant Layer 2 Data Center Network Fabric. In *Proceedings of the 2009 ACM SIGCOMM Conference (SIGCOMM'09)* (Barcelona, Spain, 2009).

[58] Nishtala, R., Fugal, H., Grimm, S., Kwiatkowski, M., Lee, H., Li, H., McElroy, R., Paleczny, M., Peek, D., Saab, P., Stafford, D., Tung, T., and Venkataramani, V. Scaling Memcache at Facebook. In *Proceedings of the 10th USENIX Symposium on Networked Systems Designa nd Implementation (NSDI'13)* (Lombard, IL, USA, 2013).

[59] Noormohammadpour, M., Raghavendra, C. S., Rao, S., and Kandula, S. DCCast: Efficient Point to Multipoint Transfers Across Datacenters. In *Proceedings of the 9th USENIX Workshop on Hot Topics in Cloud Computing (HotCloud'17)* (Santa Clara, CA, USA, 2017).

[60] Oppenheimer, D., Ganapathi, A., and Patterson, D. A. Why Do Internet Services Fail, and What Can Be Done About It? In *Proceedings of the 4th USENIX Symposium on Internet Technologies and Systems (USITS'03)* (Seattle, WA, USA, Mar. 2003).

[61] Pelkonen, T., Franklin, S., Teller, J., Cavallaro, P., Huang, Q., Meza, J., and Veeraraghavan, K. Gorilla: A Fast, Scalable, In-Memory Time Series Database. In *Proceedings of the 41st International Conference on Very Large Data Bases (VLDB'15)* (Kohala Coast, HI, USA, Aug. 2015).

[62] Potharaju, R., and Jain, N. Demystifying the Dark Side of the Middle: A Field Study of Middlebox Failures in Datacenters. In *Proceedings of the 2013 Conference on Internet Measurement Conference (IMC'13)* (Barcelona, Spain, 2013).

[63] Potharaju, R., and Jain, N. When the Network Crumbles: An Empirical Study of Cloud Network Failures and Their Impact on Services. In *Proceedings of the 4th Annual Symposium on Cloud Computing (SOCC'13)* (Santa Clara, CA, USA, 2013).

[64] Potharaju, R., Jain, N., and Nita-Rotaru, C. Juggling the Jigsaw: Towards Automated Problem Inference from Network Trouble Tickets. In *Proceedings of the 10th USENIX Symposium on Networked Systems Designa nd Implementation (NSDI'13)* (Lombard, IL, USA, 2013).

[65] Power, A. Making facebook self-healing. https://www.facebook.com/notes/facebook-engineering/making-facebook-self-healing/10150275248698920/, 2011.

[66] Robbins, J., Krishnan, K., Allspaw, J., and Limoncelli, T. Resilience Engineering: Learning to Embrace Failure. *ACM Queue 10*, 9 (Sept. 2012), 1–9.

[67] Roy, A., Zeng, H., Bagga, J., Porter, G., and Snoeren, A. C. Inside the Social Network's (Datacenter) Network. In *Proceedings of the 2015 ACM SIGCOMM Conference (SIGCOMM'15)* (London, United Kingdom, 2015).

[68] Schlinker, B., Kim, H., Cui, T., Katz-Bassett, E., Madhyastha, H. V., Cunha, I., Quinn, J., Hasan, S., Lapukhov, P., and Zeng, H. Engineering Egress with Edge Fabric: Steering Oceans of Content to the World. In *Proceedings of the 2017 ACM SIGCOMM Conference (SIGCOMM'17)* (Los Angeles, CA, USA, 2017).

[69] Simpkins, A. Facebook Open Switching System ("FBOSS") and Wedge in the open. https://code.facebook.com/posts/843620439027582/facebook-open-switching-system-fboss-and-wedge-in-the-open/, Mar. 2015.

[70] Singh, A., Ong, J., Agarwal, A., Anderson, G., Armistead, A., Bannon, R., Boving, S., Desai, G., Felderman, B., Germano, P., Kanagala, A., Provost, J., Simmons, J., Tanda, E., Wanderer, J., Hölzle, U., Stuart, S., and Vahdat, A. Jupiter Rising: A Decade of Clos Topologies and Centralized Control in Google's Datacenter Network. In *Proceedings of the 2015 ACM SIGCOMM Conference (SIGCOMM'15)* (London, United Kingdom, 2015).

[71] Statista. Number of facebook employees from 2004 to 2017 (full-time). https://www.statista.com/statistics/273563/number-of-facebook-employees/, Feb. 2018.

[72] The AWS Team. Summary of the October 22, 2012 AWS Service Event in the US-East Region. https://aws.amazon.com/message/680342/, 2012.

[73] Tseitlin, A. The Antifragile Organization. *Communications of the ACM (CACM) 56*, 8 (August 2013), 40–44.

[74] Turner, D., Levchenko, K., Snoeren, A. C., and Savage, S. California Fault Lines: Understanding the Causes and Impact of Network Failures. In *Proceedings of the 2010 ACM SIGCOMM Conference (SIGCOMM'10)* (New Delhi, India, 2010).

[75] Wu, X., Turner, D., Chen, C.-C., Maltz, D. A., Yang, X., Yuan, L., and Zhang, M. NetPilot: Automating Datacenter Network Failure Mitigation. In *Proceedings of the 2012 ACM SIGCOMM Conference (SIGCOMM'12)* (Helsinki, Finland, 2012).

[76] Yap, K.-K., Motiwala, M., Rahe, J., Padgett, S., Holliman, M., Baldus, G., Hines, M., Kim, T., Narayanan, A., Jain, A., Lin, V., Rice, C., Rogan, B., Singh, A., Tanaka, B., Verma, M., Sood, P., Tariq, M., Tierney, M., Trumic, D., Valancius, V., Ying, C., Kallahalla, M., Koley, B., and Vahdat, A. Taking the edge off with espresso: Scale, reliability and programmability for global internet peering. In *Proceedings of the 2017 ACM SIGCOMM Conference (SIGCOMM'17)* (Los Angeles, CA, USA, 2017).

[77] Zhuo, D., Ghobadi, M., Mahajan, R., Förster, K.-T., Krishnamurthy, A., and Anderson, T. Understanding and Mitigating Packet Corruption in Data Center Networks. In *Proceedings of the 2017 ACM SIGCOMM Conference (SIGCOMM'17)* (Los Angeles, CA, USA, 2017).