High-Resolution Measurement of Data Center Microbursts*

Qiao Zhang
University of Washington
qiao@cs.washington.edu

Vincent Liu
University of Pennsylvania
liuv@cis.upenn.edu

Hongyi Zeng
Facebook, Inc.
zeng@fb.com

Arvind Krishnamurthy
University of Washington
arvind@cs.washington.edu

ABSTRACT

Data centers house some of the largest, fastest networks in the world. In contrast to and as a result of their speed, these networks operate on very small timescales—a 100 Gbps port processes a single packet in at most 500 ns with end-to-end network latencies of under a millisecond. In this study, we explore the fine-grained behaviors of a large production data center using extremely high-resolution measurements (10s to 100s of microsecond) of rack-level traffic. Our results show that characterizing network events like congestion and synchronized behavior in data centers does indeed require the use of such measurements. In fact, we observe that more than 70% of bursts on the racks we measured are sustained for at most tens of microseconds: a range that is orders of magnitude higher-resolution than most deployed measurement frameworks. Congestion events observed by less granular measurements are likely collections of smaller µbursts. Thus, we find that traffic at the edge is significantly less balanced than other metrics might suggest. Beyond the implications for measurement granularity, we hope these results will inform future data center load balancing and congestion control protocols.

CCS CONCEPTS

• **Networks → Network measurement**; **Data center networks**; *Network performance analysis*; *Network monitoring*; Social media networks;

KEYWORDS

Data center traffic, microbursts

ACM Reference Format:
Qiao Zhang, Vincent Liu, Hongyi Zeng, and Arvind Krishnamurthy. 2017. High-Resolution Measurement of Data Center Microbursts. In *Proceedings of IMC '17, London, United Kingdom, November 1–3, 2017,* 8 pages. https://doi.org/10.1145/3131365.3131375

*Raw data for the distributions presented in the paper are available at https://github.com/zhangqiaorjc/imc2017-data

IMC '17, November 1–3, 2017, London, United Kingdom

ACM ISBN 978-1-4503-5118-8/17/11…$15.00
https://doi.org/10.1145/3131365.3131375

1 INTRODUCTION

Data center networks are defined by their scale. The largest of today's data centers are massive facilities that house up to hundreds of thousands of servers connected by thousands of network switches. The switches in turn have high (and rapidly growing) capacity, with state-of-the-art models able to process terabits of traffic per second at 100 Gbps per port.

In contrast to the massive aggregate bandwidth of these deployments is the minuscule timescales on which they operate: a 100 Gbps port processes packets in at most 500 ns, and a packet can traverse the entire network in tens to hundreds of microseconds. Unfortunately, much of what we know about data center traffic (and, in fact, most production monitoring systems) are either on the scale of minutes [7] or are heavily sampled [16]. While such measurements can inform us of long-term network behavior and communication patterns, in modern data center networks, coarse-grained measurements fail to provide insight into many important behaviors.

One example: congestion. In most prior work, large cloud network operators have observed that packet discards occur, but are uncorrelated or weakly correlated with observed link utilization, implying that most congestion events are too short-lived to be characterized by existing data sets. Coarse-grained measurements also make it difficult to answer questions about concurrent behavior like how many ports are involved in each congestion event or how effective the network is at load balancing. The design of data center switches, networks, and protocols depend on this type of fine-grained behavior.

Our primary contribution is to provide a high-resolution characterization of a production data center network. To do so, we developed a custom high-resolution counter collection framework on top of the data center operator's in-house switch platform. This framework is able to poll switch statistics at a 10s to 100s of microseconds granularity with minimal impact on regular switch operations.

With the framework, we proceed to perform a data-driven analysis of various counters (including packet counters and buffer utilization statistics) from Top-of-Rack (ToR) switches in multiple clusters running multiple applications. While our measurements are limited to ToR switches, our measurements and prior work [6, 9, 18] indicate that the majority of congestion occurs at that layer. More generally, we do not claim that our results are representative of all modern data center networks—they are merely a slice of one large operator's network, albeit at a heretofore unprecedented granularity. Our main findings include:

- µbursts, periods of high utilization lasting less than 1 ms, exist in production data centers, and in fact, they encompass most congestion events. The p90 burst duration is ≤200 µs.

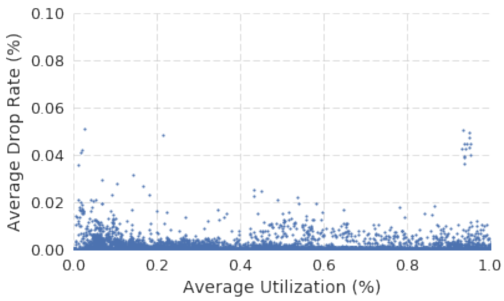


Figure 1: Scatter plot of ToR-server links' drop rates and utilization across the data center. Drops only include congestion discards and not packet corruptions. Measurements were taken at a granularity of 4 minutes, and samples were taken once per hour over the course of 24 hours.

- Link utilization is multimodal; when bursts occur, they are generally intense.
- At small timescales, many multi-statistic features become possible to measure: load can be very unbalanced, packets tend to be larger inside bursts than outside, and buffers are related to simultaneous bursts in a nonlinear fashion.

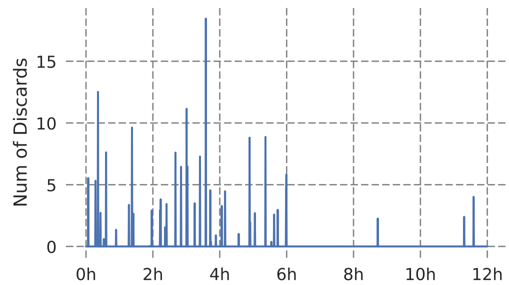
2 BACKGROUND AND RELATED WORK

Much effort has gone into measuring and understanding data center network behavior for the purpose of designing better networks. For large-scale measurements, the existing studies along these lines have taken one of two approaches:

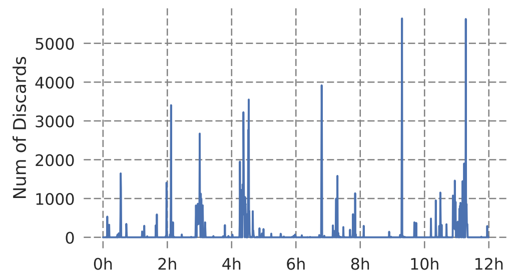
Packet sampling. One method of measuring networks is to examine packets directly. `tcpdump` provides this functionality, but raw dumps are not tractable for medium- to long-term measurement without substantial dedicated hardware/overhead [17]. Instead, some studies sample packets using sFlow in the network [16] or iptables collection on end hosts [18]. Sampling is typically done such that only one packet in thousands or tens of thousands are recorded. Facebook, for instance, typically samples packets with a probability of 1 in 30,000. Packet sampling can provide insight into traffic patterns, as was the case in [18], which found that patterns were mostly stable over time, but very different for different applications.

Coarse-grained counters. An alternative to sampling packets is to look at coarse-grained network counters like those provided by SNMP [7]. Prior studies have relied heavily on such counters. SNMP counters give up some information compared to packet samples (e.g., source and destination), but provide a view into the interaction of packets within the network. [6] and [9], for example, analyze utilization/drops in networks, and found indications of bursty behavior. Many data centers collect these statistics by default as they provide useful aggregate statistics and can be used to detect major problems in the network. Typical granularities for SNMP collection in data centers are on the order of minutes [6, 18].

In addition to the above, researchers have proposed switch hardware modifications to provide more scalable, accurate measurements [10, 12, 14, 20]; however, these are not deployed widely enough to perform large-scale production measurements.



(a) Low-utilization Port



(b) High-utilization Port

Figure 2: Time series of drops on two different ports. (a) has relatively low utilization, and (b) has relatively high utilization. Samples were taken at a granularity of 1 min over a 12 hr time span.

3 THE CASE FOR HIGH RESOLUTION

The granularity of coarse-grained counters and packet sampling makes it difficult to answer many important questions about the underlying behavior of the network. As an example of how granularity can hinder our view of network behavior, Fig. 1 shows a scatter plot of utilization and packet discard counters of ToR-server links across a data center. For every ToR-server in the data center we studied and every hour in a day, we sub-sample by randomly picking a 4-minute interval for the hour and take measurements during that period. The utilization and drop rates are computed over a 4-minute interval (the SNMP polling interval used in production). Despite a wide range of observed average utilization, utilization does not have a strong effect on drop rates (correlation coefficient of 0.098).

Part of the issue is that, at this granularity, only severe or sustained congestion would result in high drop rates. The time series of switch behavior shown in Fig. 2 provides some further insight into why utilization and drop rate do not match. We chose two switch ports that were experiencing congestion drops and plotted their drops at a granularity of 1 minute over time over the course of 12 hours. One switch port had relatively low utilization (~9%) as it was on the critical path for web requests. The other had high utilization (~43%) and ran offline data processing. In both cases, drops occur in bursts, often lasting less than the measurement granularity (1 minute in this case). Succeeding intervals often have no drops.

We therefore pose the following questions:

- What do bursts look like and how often do they occur?
- What role do μbursts (high utilization lasting <1 ms) play?
- Does network behavior differ significantly inside a burst?
- Is there synchronized behavior during bursts?

4 DATASETS AND METHODOLOGY

To answer the above questions, we built a measurement framework that is able to collect extremely fine-grained samples of various switch statistics. The goal of our framework is to be able to observe network changes that occur on the order of a few RTTs. In this section, we describe the framework we built for collecting fine-grained switch measurements, the associated collection methodology, and the resulting data sets.

4.1 High-resolution counter collection

Our sampling framework was built on top of the data center's in-house switch platform, and it allows operators to poll a subset of switch counters at microsecond-level granularity.

Our framework takes advantage of the fact that modern switches include relatively powerful general-purpose multi-core CPUs in addition to their switching ASICs (Application-Specific Integrated Circuits). The ASICs are responsible for packet processing, and as part of their functionality, they maintain many useful counters. The CPUs, on the other hand, are traditionally responsible for handling control plane logic. By modifying the switch platform, we can enlist the CPU to also poll its local counters at extremely low latency. The CPU batches the samples before sending them to a distributed collector service that is both fine-grained and scalable.

Polling intervals are best-effort as kernel interrupts and competing resource requests can cause the sampler to miss intervals. To obtain precise timing, the framework requires a dedicated core, but can trade away precision to decrease utilization to $\leq 20\%$ in most cases. The maximum polling rate depends on the target counter as well as the target switch ASIC. Differences arise due to hardware limitations: some counters are implemented in registers versus memory, others may involve multiple registers or memory blocks. For the counters we measure, we manually determine the minimum sampling interval possible while maintaining $\sim 1\%$ sampling loss. For instance, one of the counters we measure (a byte counter) exhibited the following sampling loss behavior, leading us to choose a 25 µs interval:

Sampling interval	Missed intervals
1 µs	100%
10 µs	~10%
25 µs	~1%

Table 1: The effect of sampling interval on miss rate for a byte counter. When the sampler misses an interval, we still capture the total number of bytes and correct timestamp.

Multiple counters can be polled together with a sublinear increase in sampling rate depending on the specific combination of counters. As with single counters, collection of groups of counters are tuned manually. With the exception of Sec. 5.3, measurements in Sec. 5 were all taken using single-counter measurement campaigns in order to achieve the highest resolution possible. Sec. 5.3 and Sec. 6 included multiple counters per measurement campaign, but one campaign per set of experimental results.

In this paper, we mainly focus on three sets of counters. We briefly describe them and list their single-instance sampling rate here.

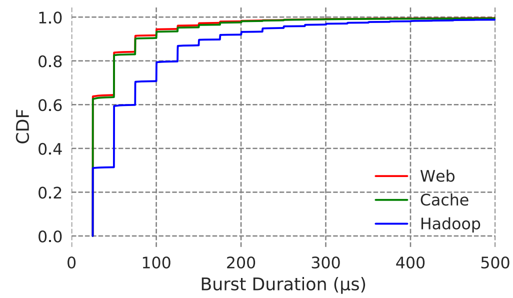


Figure 3: CDF of µburst durations at a 25 µs granularity.

Byte count. The primary set we use measures the cumulative number of bytes sent/received per switch port. We use these byte counts to calculate throughput. As mentioned above, our framework can poll a single instance of these counters every 25 µs with low sampling loss. When a sample miss does occur, we can still calculate throughput accurately using the sample's timestamp and byte count. At these timescales, we can measure the network at a granularity much smaller than even a single RTT.

Packet size. Similar to byte count, we also collect a histogram of the packet sizes sent/received at each switch port. The ASIC bins packets into several buckets that we list in Sec. 5.3. These can typically be polled at the same granularity as byte counters.

Peak buffer utilization. The third set measures the buffer utilization of the switch. For this counter, we take the peak utilization of the buffer since the last measurement so that we do not miss any congestion events, and we reset the counter after reading it. Thus, even when the sampling loop misses a sampling period, our results will still reflect bursts. This counter takes much longer to poll than byte or packet size counters (50 µs).

4.2 Data set

Network architecture. The data center we study uses a conventional 3-tier Clos network and is described in [4]. Machines are organized into racks and connected to a Top-of-Rack (ToR) switch via 10 Gbps Ethernet links. Each ToR is, in turn, connected by either 40 Gbps or 100 Gbps links to an aggregation layer of "fabric" switches. The fabric switches are then connected to a third layer of switches, called "spines". The entire structure forms a multi-rooted tree, with the spine switches as the roots of the tree, and the ToRs as the leaves.

Due to current deployment restrictions, we concentrate on ToR switches for this study and leave the study of other network tiers to future work. Prior work and our own measurements show that the majority of loss occurs at ToR switches and that they tend to be more bursty (lower utilization and higher loss) than higher-layer switches [19]. Most of these drops occur in the ToR-server direction (~90% in the data center we measured). In that sense, ToR measurements most likely represent a worst case over all switches in the data center.

Workload. Our data set spans a few applications, but a distinctive aspect of the data center we measured is that servers typically have a single role. In particular, we focus on three applications that show a diverse set of behaviors and are among the most prevalent types of machines in the data center.

- *Web:* These servers receive web requests and assemble a dynamic web page using data from many remote sources.
- *Cache:* These servers serve as an in-memory cache of data used by the web servers. Some of these servers are leaders, which handle cache coherency, and some are followers, which serve most read requests [15].
- *Hadoop:* Unlike the previous two categories, these servers are not part of the interactive path. Instead, Hadoop servers are used for offline analysis and data mining.

See [4] for a more detailed description of each application's traffic patterns.

As entire racks are typically dedicated to each of these roles, even when measuring at a ToR level, our results can isolate the behavior of different classes of applications. Our measurements span a total of 30 racks, consisting of 10 racks for each application type over the course of 24 hours. Due to data retention limitations, storing all samples of all counters over 24 hours was not feasible, so for each rack, we pick a random port, and pick a random 2-minute interval for every hour throughout the day. Diurnal patterns are therefore captured within our data set. In total, we sampled 720 two-minute intervals, each with around 5 million data points, totaling 250 GB. The full data would have taken hundreds of terabytes.

5 PORT-LEVEL BEHAVIOR

We begin our analysis by studying the fine-grained behavior of individual ports before proceeding in Sec. 6 to consider the interactions between ports in a switch. From fine-grained behavior, our goal is to observe and characterize the bursty nature of data center networks.

5.1 Existence of µbursts

In Sec. 3, we noted that coarse-grained measurements of data center networks suggest bursty behavior on very small timescales. To test this hypothesis, we measure the duration of bursts at 25 µs granularity. As in [8], we say that a switch's egress link is *hot* if, for the measurement period, its utilization exceeds 50%. An unbroken sequence of hot samples indicates a *burst.*[1] We can see a few interesting results from the measurements shown in Fig. 3.

High utilization is indeed short-lived. A significant fraction of these bursts are only one sampling period long. The 90th percentile duration is less than 200 µs for all three rack types, with Web racks having the lowest 90th percentile burst duration at 50 µs (two sampling periods). Hadoop racks have the longest tail of the three, but even then, almost all bursts concluded within 0.5 ms. The results indicate that bursts not only exist, almost all high utilization at the edge of the data center network is part of a µburst. Congestion events observed by less granular measurements are likely collections of smaller µbursts.

Bursts are correlated. While a significant portion of bursts last for less than a sampling period, these high-utilization intervals do tend to be correlated. We can demonstrate this using a simple likelihood ratio test. First, we create a two-state first-order Markov model. We classify each 25 µs interval as 'hot' ($x_t = 1$) or not ($x_t = 0$) based on its utilization level. Then, we count consecutive occurrences of same-state intervals or flipped intervals. This allows to compute the MLE (Maximum Likelihood Estimates) of its transition matrix, $p(x_t = a|x_{t-1} = b) = \frac{count(x_t=a, x_{t-1}=b)}{count(x_{t-1}=b)}$, shown in Tab. 2.

[1] We choose to define a burst by throughput rather than buffer utilization as buffers in our switches are shared and dynamically carved, making pure byte counts a more deterministic measure of burstiness.

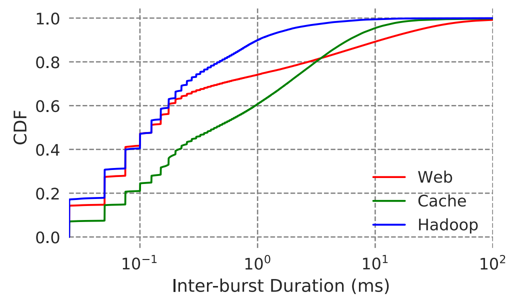


Figure 4: CDF of the time between bursts at a 25 µs granularity.

$p(x_t \mid x_{t-1})$	Web $x_t = 0$	Web $x_t = 1$	Cache $x_t = 0$	Cache $x_t = 1$	Hadoop $x_t = 0$	Hadoop $x_t = 1$
$x_{t-1} = 0$	0.997	0.003	0.984	0.016	0.958	0.042
$x_{t-1} = 1$	0.641	0.359	0.279	0.721	0.345	0.655

Table 2: Transition Matrix for Burst Markov Model

Given this Markov model, we can then compute the likelihood ratio of $r = \frac{p(x_t=1|x_{t-1}=1)}{p(x_t=1|x_{t-1}=0)}$. If burst intervals are independently arriving, we would expect $r \approx 1$ because it would imply the probability of seeing the next burst is the same whether the previous time period saw a burst or not. The actual ratios are much higher, indicating that high utilization samples are correlated:

$$r_{web} = 0.359/0.003 = 119.7 \quad (1)$$

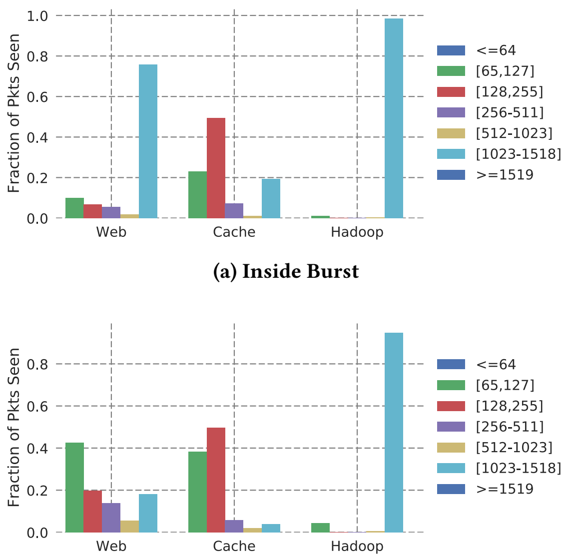
$$r_{cache} = 0.721/0.016 = 45.1 \quad (2)$$

$$r_{hadoop} = 0.655/0.042 = 15.6 \quad (3)$$

Fine-grained measurements are needed to capture certain behaviors. Our results also confirm our intuition that fine-grained measurements are needed to accurately measure bursty behavior. They also offer a potential explanation for the skewed behavior found in Sec. 3 and prior studies. In fact, it is possible that our 25 µs measurement granularity is itself too coarse as over 60% of Web and Cache bursts terminated within that period. Faster networks will likely increase the necessary granularity. Unfortunately, the sampling rate is fundamentally limited by latency between the CPU and the ASIC, suggesting that additional hardware support may be necessary for fine-grained measurements in the future.

5.2 Time between µbursts

The time *between* µbursts is just as interesting as the bursts themselves. Fig. 4 shows a CDF of the duration of these inter-burst periods. Unlike our measurements of µbursts and their duration, inter-burst periods have a much longer tail. It is still the case that most inter-burst periods are small, particularly for Cache and Web



(a) Inside Burst

(b) Outside Burst

Figure 5: Normalized histogram of packet size distribution over a 100 µs periods inside/outside a period of high utilization.

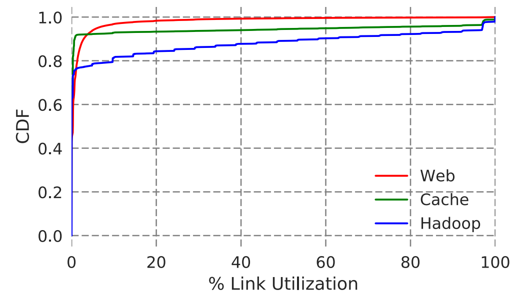


Figure 6: CDF of link utilization at a 25 µs granularity.

racks where 40% of inter-burst periods last less than 100 µs, but when idle periods are persistent, they tend to be measured on the order of hundreds of milliseconds—several orders of magnitude larger than burst durations. From this data, we can also see that the arrival rate of µbursts is not a homogeneous/constant-rate Poisson process. We tested that using a Kolmogorov-Smirnov goodness of fit test on the inter-arrival time with exponential distribution, and got a p-value close to 0, allowing us to reject the null hypothesis that the burst arrivals are Poisson.

5.3 Packet size distribution

The overall packet size distribution of our traffic conforms to prior work: Hadoop sees mostly full-MTU packets, while Web and Cache sees a wider range of packet sizes [6, 18].

Interestingly, however, burst and non-burst periods can sometimes differ substantially in their makeup. This effect varies from application to application, but generally speaking, bursty periods tend to include more large packets than non-bursty periods. Fig. 5 compares the two cases using packet size histograms for the three rack types we measured. Packets were binned by their size into several ranges and polled alongside the total byte count of the interface in order to classify the samples.

The increase in large packets during bursts exists, but is not very pronounced in Hadoop, where the vast majority of packets are always large. Cache servers see a relative large-packet increase of about 20%, but smaller packets still dominate measurements. Web servers see a large relative increase of about 60% coming from all other packet sizes. The material packet-level difference between packets inside and outside bursts suggests that bursts at the ToR layer, even in the Hadoop case, are often a result of application-behavior changes, rather than random collisions.

5.4 High-resolution network utilization

Taking the links as a whole (both burst and non-burst periods), we find that different applications have different utilization patterns, but that all of them are extremely long tailed (Fig. 6). As such, our choice of 50% as a high-utilization threshold generates similar results compared to other possible thresholds—when bursts occur, they are generally intense, particularly for Hadoop, which spends 10% of sampling periods at close to 100% utilization. Further demonstrating the burstiness of network traffic, we find that Cache and Hadoop have multimodal utilization at this granularity. Of the three types, Hadoop ports spend the most time in bursts at ∼15%.

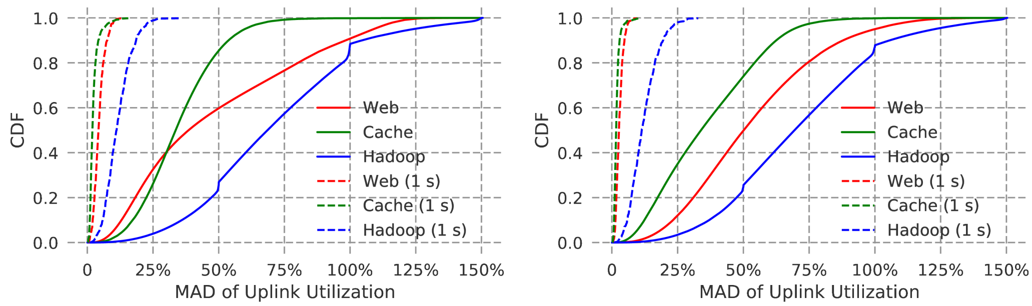
6 CROSS-PORT BEHAVIOR

Given observations of the bursty behavior of individual links in data center networks, we now delve into the synchronized behavior of those ports. Conceptually, each switch's ports can be split into two classes: uplinks and downlinks. The uplinks connect the rack to the rest of the data center, and modulo network failures, they are symmetric in both capacity and reachability. Downlinks connect to individual servers, which in our data set all serve similar functions. The granularity of our measurements allows us to explore the relationships between these ports at the scale of individual congestion events.

6.1 Efficacy of network load balancing

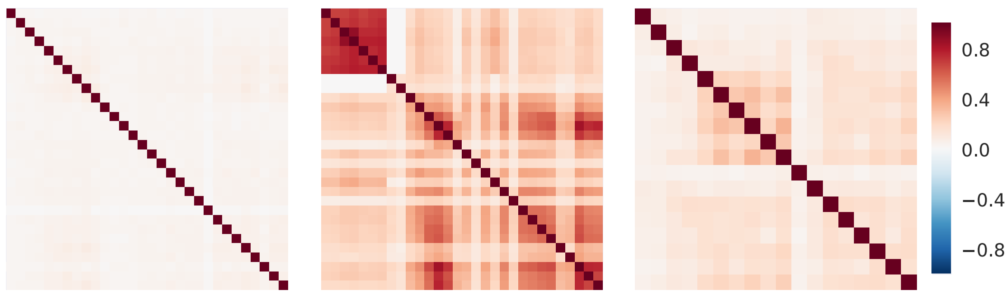
ToR switches use Equal-Cost MultiPath (ECMP) to spread load over each of their four uplinks. In principle, a per-packet, round-robin protocol would perfectly balance outgoing traffic. In practice, however, typical ECMP configurations introduce at least two sources of potential imbalance in order to avoid TCP reordering: (1) ECMP operates on the level of flows, rather than packets, and (2) it uses consistent hashing, which cannot guarantee optimal balance.

Uplinks are unbalanced at small timescales. High-resolution measurements allow us to more accurately quantify *how much* these differ from optimal. The instantaneous efficacy of load balancing has implications for drop- and latency-sensitive protocols like RDMA and TIMELY [13]. Fig. 7a shows the mean absolute deviation (MAD) of the four uplinks within a sampling period (40 µs or 1 s). Unsurprisingly, Hadoop, with its longer flows, is less balanced than the other two racks. This imbalance can sometimes be large at small timescales, with the Hadoop racks' 90th percentile showing an average deviation of 100%. Even in the median case, all three types of racks had a MAD of over 25%, indicating that



(a) Egress **(b) Ingress**

Figure 7: CDF of the mean absolute deviation (MAD) of uplink utilization within a given sampling period. A deviation of 0 means that the uplinks are perfectly balanced. We show both egress and ingress directions, as well as 1 s and 40 µs granularities.



(a) Web Rack **(b) Cache Rack** **(c) Hadoop Rack**

Figure 8: Heatmap of Pearson correlation coefficients for servers in the same rack. We measured the ToR-to-server utilization of three representative racks at a granularity of 250 µs.

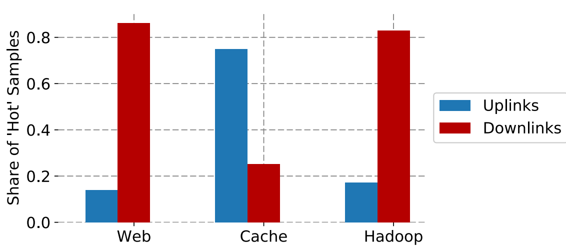


Figure 9: Uplink/downlink share of hot ports given 300 µs sampling.

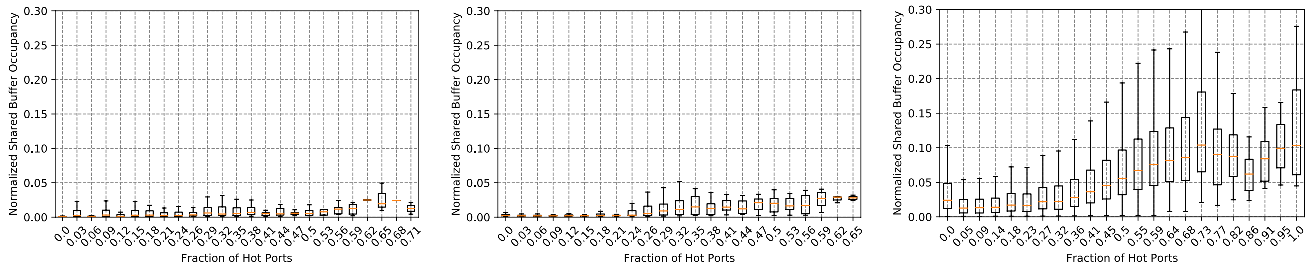
flow-level load balancing can often be inefficient in the short term. Any slow-reacting load balancing approach (e.g., WCMP) would not resolve the issue as, at the moderately longer timescale of 1 s, the links appear to be balanced.

The interconnect does not add significant variance. After traversing the interconnect, traffic entering the ToR switch exhibits a similar pattern. We can observe the MAD of ingress traffic in Fig. 7b. While there are changes in dispersion, they are relatively small indicating that the network, with its current structure and load levels, is not adding additional variance between flows. Prior work has indicated that imbalance becomes significantly worse in the presence of asymmetry caused by failures [1, 11], but we were not able to intercept such cases for the racks we measured.

6.2 Correlation between servers

One might also expect, with ideal application and Layer-4 load balancing, that downlink utilization is balanced. As above, the reality is a bit more nuanced and is heavily dependent on the type of the rack in question. To factor out differences in the absolute amount of traffic going to each server, we show in Fig. 8 Pearson correlation coefficients, which track the linear correlation between pairs of servers. Ingress and egress trends were almost identical, so we only show the ToR-to-server direction.

For Web racks, there is almost no correlation. Diurnal patterns and flash crowds have been shown to cause correlation at longer timescales, but at small timescales, the effect of those factors are not easily discernible. Instead, because Web servers run stateless services that are entirely driven by user requests, correlation is close to zero. For Hadoop, there is some amount of correlation, but it is modest at these timescales. The Cache rack exhibits very different behavior from the other two types of racks. Subsets of the Cache servers show very strong correlation with one another. This is due to the fact that their requests are initiated in groups from web servers. As such, those subsets are potentially involved in the same scatter-gather requests.



(a) Web (b) Cache (c) Hadoop

Figure 10: Normalized peak shared buffer occupancy versus number of hot ports for web/cache/hadoop racks at a 300 µs granularity.

6.3 Directionality of bursts

Having examined both uplinks and downlinks separately, we now look at their relative behavior. Fig. 9 shows for each rack type the relative frequency of hot uplinks/downlinks.

We find that, for Web and Hadoop racks, there is a significant bias toward servers as opposed to toward uplinks. Only 18% of hot Hadoop samples were for uplinks, with Web uplinks responsible for an even lower share. For these racks, bursts tend to be a result of high fan-in where many servers send to a single destination.

Cache servers show the opposite trend, with most bursts occurring on the uplinks. This can be attributed to two properties of Cache servers: (1) that they exhibit a simple response-to-request communication pattern, and (2) that cache responses are typically much larger than the requests. Thus, Cache servers will almost always send more traffic than they receive. Combined with modest oversubscription at the ToR layer (at a ratio of 1:4), the communication bottleneck for these racks lies in their ToRs' uplinks.

6.4 Effect of µbursts on shared buffers

Finally, we examine the effect of synchronized bursts on ToRs' shared buffer utilization. Fig. 10 depicts a boxplot of the peak buffer occupancy during a 50 ms interval versus the number of hot ports during that same span. Granularity of these measurements was lower than others because of a relatively inefficient interface to poll the shared buffer utilization. As mentioned, buffer carving is dynamic so, for simplicity, we normalize the occupancy to the maximum value we observed in any of our data sets. We note that drops can occur at much lower buffer utilization because of these effects.

As expected, Hadoop racks put significantly more stress on ToR buffers than either Web or Cache racks. This manifests in a few different ways. First, we observed that Hadoop sometimes drove 100% of its ports to > 50% utilization. Web and Cache only drove a maximum of 71% and 64% of their ports to simultaneous high utilization within the observation period. Further, Hadoop experiences high standing buffer occupancy compared to Web/Cache, and the buffer occupancy scales with the number of hot ports more drastically than in Web/Cache.

In all cases, average occupancy levels off for high numbers of hot ports, possibly due to self selection of communication patterns or the effect where buffer requirements scale sublinearly with the number of TCP connections [5].

7 DATA CENTER DESIGN IMPLICATIONS

Our measurements of a production data center point to the need for fine-grained measurement in order to truly understand network behavior. This has implications not only for network measurement, but also the evaluation of new protocols and architectures.

It also has implications for the design of those protocols and architectures. While domain knowledge suggests that application-level demand and traffic patterns are a significant contributor to bursts, the data does not explicitly point to a cause[2]. Regardless, the fact remains that µbursts both exist and are responsible for the majority of congestion in the measured production data center. We discuss some of the implications of that observation below.

Implications for load balancing. Many recent proposals suggest load balancing on microflows rather than 5-tuples—essentially splitting a flow as soon as the inter-packet gap is long enough to guarantee no reordering. While our framework does not measure inter-packet gaps directly, we note that most observed inter-burst periods exceed typical end-to-end latencies (Sec. 5.2) and that non-burst utilization is low (Sec. 5.4). The caveat is that different applications can have significantly different behavior, and faster networks may decrease the gaps relative to the reordering constraints.

Implications for congestion control. Traditional congestion control algorithms either react to packet drops, RTT variation [13] or ECN [2] as a congestion signal. All of these signals require at least RTT/2 to arrive at the sender, and the protocols can potentially take many RTTs to adapt. Unfortunately, our measurements show that a large number of µbursts are shorter than a single RTT. Buffering can handle momentary congestion, but if buffers become comparatively smaller or initial sending rates become more aggressive, lower-latency congestion signals may be required.

Implications for pacing. TCP pacing was one of the original mechanisms that prevented bursty traffic. Over time, however, it has been rendered ineffective through features like segmentation offload and interrupt coalescing. The results presented in this paper give some insight into the degree of the problem in practice. They may point to the importance of recent pacing proposals at either the hardware [3] and software [13] levels. Even so, existing protocols mostly deal with single-flow or single-machine pacing.

[2] Doing so would require correlating and synchronizing switch and end hosts measurements at a microsecond level, which was not feasible in our current deployment.

8 CONCLUSION

As network bandwidth continues to rise in data centers, the timescale of network events will decrease accordingly. Thus, it is essential to understand the behavior of these networks at high-resolution. This is particularly true as we find that most bursts of traffic are *µbursts*, i.e., that they occur at a microsecond-level granularity. Our results show that at these small timescales, traffic is extremely bursty, load is relatively unbalanced, and that different applications have significantly different behavior. We hope that these findings will inform future data center network measurement and design.

REFERENCES

[1] Mohammad Alizadeh, Tom Edsall, Sarang Dharmapurikar, Ramanan Vaidyanathan, Kevin Chu, Andy Fingerhut, Vinh The Lam, Francis Matus, Rong Pan, Navindra Yadav, and George Varghese. 2014. CONGA: Distributed Congestion-aware Load Balancing for Datacenters. In *Proceedings of the 2014 ACM Conference on SIGCOMM (SIGCOMM '14)*. ACM, New York, NY, USA, 503–514. https://doi.org/10.1145/2619239.2626316

[2] Mohammad Alizadeh, Albert Greenberg, David A. Maltz, Jitendra Padhye, Parveen Patel, Balaji Prabhakar, Sudipta Sengupta, and Murari Sridharan. 2010. Data Center TCP (DCTCP). In *Proceedings of the ACM SIGCOMM 2010 Conference on Data Communication (SIGCOMM '10)*. ACM, New York, NY, USA, 63–74. https://doi.org/10.1145/1851182.1851192

[3] Mohammad Alizadeh, Abdul Kabbani, Tom Edsall, Balaji Prabhakar, Amin Vahdat, and Masato Yasuda. 2012. Less Is More: Trading a Little Bandwidth for Ultra-Low Latency in the Data Center. In *Proceedings of the 9th USENIX Symposium on Networked Systems Design and Implementation (NSDI 12)*. USENIX, San Jose, CA, 253–266.

[4] Alexey Andreyev. 2014. Introducing data center fabric, the next-generation Facebook data center network. https://code.facebook.com. (Nov. 2014).

[5] Guido Appenzeller, Isaac Keslassy, and Nick McKeown. 2004. Sizing Router Buffers. In *Proceedings of the 2004 ACM SIGCOMM Conference on Data Communication (SIGCOMM '04)*. ACM, New York, NY, USA, 281–292. https://doi.org/10.1145/1015467.1015499

[6] Theophilus Benson, Aditya Akella, and David A. Maltz. 2010. Network Traffic Characteristics of Data Centers in the Wild. In *Proceedings of the 10th ACM SIGCOMM Conference on Internet Measurement (IMC '10)*. ACM, New York, NY, USA, 267–280. https://doi.org/10.1145/1879141.1879175

[7] J. Case, Mundy R., Partain D., and Stewart B. 2002. Introduction and Applicability Statements for Internet Standard Management Framework. (2002). https://tools.ietf.org/html/rfc3410.

[8] Daniel Halperin, Srikanth Kandula, Jitendra Padhye, Paramvir Bahl, and David Wetherall. 2011. Augmenting Data Center Networks with Multi-gigabit Wireless Links. In *Proceedings of the ACM SIGCOMM 2011 Conference on Data Communication (SIGCOMM '11)*. ACM, New York, NY, USA, 38–49. https://doi.org/10.1145/2018436.2018442

[9] Srikanth Kandula, Sudipta Sengupta, Albert Greenberg, Parveen Patel, and Ronnie Chaiken. 2009. The Nature of Data Center Traffic: Measurements & Analysis. In *Proceedings of the 9th ACM SIGCOMM Conference on Internet Measurement (IMC '09)*. ACM, New York, NY, USA, 202–208. https://doi.org/10.1145/1644893.1644918

[10] Changhoon Kim, Anirudh Sivaraman, Naga Katta, Antonin Bas, Advait Dixit, and Lawrence J Wobker. 2015. In-band network telemetry via programmable dataplanes. *SIGCOMM Demo* (2015).

[11] Vincent Liu, Daniel Halperin, Arvind Krishnamurthy, and Thomas Anderson. 2013. F10: A Fault-tolerant Engineered Network. In *Proceedings of the 10th USENIX Conference on Networked Systems Design and Implementation (NSDI'13)*. USENIX Association, Berkeley, CA, USA, 399–412. http://dl.acm.org/citation.cfm?id=2482626.2482665

[12] Zaoxing Liu, Antonis Manousis, Gregory Vorsanger, Vyas Sekar, and Vladimir Braverman. 2016. One Sketch to Rule Them All: Rethinking Network Flow Monitoring with UnivMon. In *Proceedings of the 2016 ACM SIGCOMM Conference on Data Communication (SIGCOMM '16)*. ACM, New York, NY, USA, 101–114. https://doi.org/10.1145/2934872.2934906

[13] Radhika Mittal, Vinh The Lam, Nandita Dukkipati, Emily Blem, Hassan Wassel, Monia Ghobadi, Amin Vahdat, Yaogong Wang, David Wetherall, and David Zats. 2015. TIMELY: RTT-based Congestion Control for the Datacenter. In *Proceedings of the 2015 ACM Conference on Data Communication (SIGCOMM '15)*. ACM, New York, NY, USA, 537–550. https://doi.org/10.1145/2785956.2787510

[14] Masoud Moshref, Minlan Yu, Ramesh Govindan, and Amin Vahdat. 2015. SCREAM: Sketch Resource Allocation for Software-defined Measurement. In *Proceedings of the 11th ACM Conference on Emerging Networking Experiments and Technologies (CoNEXT '15)*. ACM, New York, NY, USA, 14:1–14:13. https://doi.org/10.1145/2716281.2836099

[15] Rajesh Nishtala, Hans Fugal, Steven Grimm, Marc Kwiatkowski, Herman Lee, Harry C. Li, Ryan McElroy, Mike Paleczny, Daniel Peek, Paul Saab, David Stafford, Tony Tung, and Venkateshwaran Venkataramani. 2013. Scaling Memcache at Facebook. In *Proceedings of the 10th USENIX Conference on Networked Systems Design and Implementation (NSDI'13)*. USENIX Association, Berkeley, CA, USA, 385–398. http://dl.acm.org/citation.cfm?id=2482626.2482663

[16] P. Phaal, S. Panchen, and N. McKee. 2001. InMon Corporation's sFlow: A Method for Monitoring Traffic in Switched and Routed Networks. RFC 3176 (Informational). (2001).

[17] Jeff Rasley, Brent Stephens, Colin Dixon, Eric Rozner, Wes Felter, Kanak Agarwal, John Carter, and Rodrigo Fonseca. 2014. Planck: Millisecond-scale Monitoring and Control for Commodity Networks. In *Proceedings of the 2014 ACM Conference on SIGCOMM (SIGCOMM '14)*. ACM, New York, NY, USA, 407–418. https://doi.org/10.1145/2619239.2626310

[18] Arjun Roy, Hongyi Zeng, Jasmeet Bagga, George Porter, and Alex C. Snoeren. 2015. Inside the Social Network's (Datacenter) Network. In *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication (SIGCOMM '15)*. ACM, New York, NY, USA, 123–137. https://doi.org/10.1145/2785956.2787472

[19] Arjun Singh, Joon Ong, Amit Agarwal, Glen Anderson, Ashby Armistead, Roy Bannon, Seb Boving, Gaurav Desai, Bob Felderman, Paulie Germano, Anand Kanagala, Jeff Provost, Jason Simmons, Eiichi Tanda, Jim Wanderer, Urs Hölzle, Stephen Stuart, and Amin Vahdat. 2015. Jupiter Rising: A Decade of Clos Topologies and Centralized Control in Google's Datacenter Network. In *Proceedings of the 2015 ACM Conference on Data Communication (SIGCOMM '15)*. ACM, New York, NY, USA, 183–197. https://doi.org/10.1145/2785956.2787508

[20] Minlan Yu, Lavanya Jose, and Rui Miao. 2013. Software Defined Traffic Measurement with OpenSketch. In *Proceedings of the 10th USENIX Conference on Networked Systems Design and Implementation (NSDI'13)*. USENIX Association, Berkeley, CA, USA, 29–42. http://dl.acm.org/citation.cfm?id=2482626.2482631