Cloud Datacenter SDN Monitoring: Experiences and Challenges

Arjun Roy, Deepak Bansal[†], David Brumley[†], Harish Kumar Chandrappa[†]
Parag Sharma[†], Rishabh Tewari[†], Behnaz Arzani[†] and Alex C. Snoeren

Department of Computer Science and Engineering
University of California, San Diego

[†]Microsoft Corporation

ABSTRACT

Cloud customers require highly reliable and performant leased datacenter infrastructure to deliver quality service for their users. It is thus critical for cloud providers to quickly detect and mitigate infrastructure faults. While much is known about managing faults that arise in the datacenter physical infrastructure (i.e., network and server equipment), comparatively little has been published regarding management of the logical overlay networks frequently employed to provide strong isolation in multi-tenant datacenters.

We present a first look into the nuances of monitoring these "virtualized" networks through the lens of a large cloud provider. We describe challenges to building cloud-based fault monitoring systems, and use the output of a production system to illuminate how virtualization impacts multi-tenant datacenter fault management. We show that interactions between the virtualization, tenant software, and lower layers of the network fabric both simplify and complicate different aspects of fault detection and diagnosis efforts.

CCS CONCEPTS

• **Networks** → **Network measurement**; **Cloud computing**; **Network monitoring**; *Network performance analysis*; *Data center networks*; *Overlay and other logical network structures*; Network management;

ACM Reference Format:
Arjun Roy, Deepak Bansal, David Brumley, Harish Kumar Chandrappa,Parag Sharma, Rishabh Tewari, Behnaz Arzani and Alex C. Snoeren. 2018. Cloud Datacenter SDN Monitoring: Experiences and Challenges. In *2018 Internet Measurement Conference (IMC '18), October 31-November 2, 2018, Boston, MA, USA.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3278532.3278572

1 INTRODUCTION

Web service operators and IT-dependent enterprises increasingly rely on cloud providers to address their computational needs. Cloud tenants simultaneously expect high reliability and performance while realizing cost and management advantages from leveraging leased, shared infrastructure. To support potentially conflicting customer use-cases in shared infrastructure, cloud providers employ host virtualization to provide resource multiplexing and isolation. Similarly, network virtualization allows tenants to operate within cloud datacenters without undue complexity or contention. Rather than sharing IP addresses and logical network topology with other customers, tenants typically operate inside of virtual networks (VNETs) within a software-defined network (SDN) overlay provided by the datacenter operator.

To provide high performance, datacenter operators monitor networks to rapidly detect, localize, and mitigate faults as they inevitably occur [4, 7, 9, 12, 17, 18] and potentially harm application performance [9, 14]. Datacenter monitoring has received much recent attention in the context of physical network fabrics. Little is yet known—in academic literature, at least—of the operational realities of virtualized, multi-tenant networks. While the end goal is the same, virtualized networks impart additional challenges in the form of black-box tenants and increased infrastructure complexity.

Here, we provide a first look into cloud network fault management, focusing on tenant VNET monitoring within Microsoft Azure using VNET Pingmesh. We present some of our early operational experiences over a period of several months (backed by collected production data where relevant), allowing us to answer the following high-level questions:

(1) How can cloud operators ***monitor*** VNET performance, given black-box tenants? Can physical tools [9, 12, 18] be usefully adapted to virtualized networks?
(2) How accurate are adapted monitors within virtualized environments? Can they ***detect*** customer-impacting faults? Do they exhibit high *precision* and *recall*?
(3) Beyond monitoring, how do virtual environments impact fault ***management***? How might we *triage* which layer of the network is resposible for a fault? How are fault *diagnosis*, *root-causing* and *mitigation* impacted?

While VNET Pingmesh demonstrates that physical-layer monitoring tools can be successfully adapted to VNET overlays and deliver monitoring wins, we find that the additional complexities of tenant networks and virtualization create subtle challenges that can confound VNET monitoring. Our preliminary experience suggests two takeaway lessons:

First, *cross-layer aliasing* can complicate monitoring measurement interpretation compared to well-understood physical-layer monitors. Both compute-substrate and customer-driven effects have complicated analysis by either raising alerts unrelated to customer impact, or potentially overshadowing actual faults. Thus, cross-layer aliasing can impact detection *precision* and *recall*. Furthermore, while confounding effects can be accounted for when encountered, the ever-increasing search space of cross-layer interactions coupled with ever-expanding network feature sets suggests that handling all possible effects *a priori* is infeasible.

Second, even after fault detection, *diagnosing*, *root-causing* and *mitigating* network overlay faults are significantly harder than at the physical layer. Since (baremetal) hosts serve as packet-forwarding devices, we are subject both to performance impacts caused by workloads we do not control—both customer applications and first-party host management—as well as occasional, hard to diagnose server anomalies. While prior approaches to physical layer faults can route around damage or reboot affected devices, VNET architecture and tenant workloads can make overlay faults comparatively *sticky*, since it is harder to route around the only virtual switch providing tenant VM network access, or to reboot servers hosting smaller tenant deployments.

2 PHYSICAL NETWORK MONITORING

Large-scale, multi-path datacenter network fabrics have been subject to considerable scrutiny; various studies have provided both taxonomies of common network failures [9, 13, 14, 16, 18], networked application performance [11, 14], and methods for pinpointing the cause and location of performance-sapping faults [3, 4, 12, 14].

Contemporary datacenter (physical-layer) monitoring often actively probes networks by injecting synthetic traffic to ascertain liveness and adequate performance [2, 8, 9, 15]. The Azure cloud uses Pingmesh [9] (among other systems [5, 8, 18]) to do so. Pingmesh maintains a (non-full due to scaling concerns) mesh of ping measurements, allowing at-a-glance visualizations depicting latency and connectivity between servers in a rack, between racks in a datacenter and amongst datacenters. Two-dimensional heatmap visualizations provide distinctive patterns depending on the various kinds of switch failures encountered (e.g. rack failures are visually distinct from core switch failures), while the measured latency provides insight into the quality of network performance. For example, an increase in p50 latency might signify switch queue buildups in the network core, while an increase only in p99 latency may signify errors causing packet drops.

Pingmesh has provided several monitoring wins at Microsoft, including detecting hard-to-find silent packet drops and black holes [9]. Thus, a natural question to ask is whether Pingmesh can be adapted to monitor VNETs, and how effective it would be in this environment. Next, we discuss an uplifted version of this system called VNET Pingmesh and examine its efficacy.

3 MONITORING AZURE VNETS

Microsoft Azure consists of datacenters across the world. A region can contain several datacenters, each with several clusters. Clusters contain racks aggregating servers that multiplex VMs that are organized into non-interfering VNETs.

3.1 VNET addressing and packet handling

VNETs are topologically flat, L3-addressed IP network overlays built atop the physical topology. VNETs can aggregate thousands of individual VMs, each with one or more virtual NICs. Each NIC has a customer-chosen virtualized "customer IP address" or "CA". Customer applications on a VM address other VMs in the VNET using CAs; CAs can be reused without conflict in disjoint VNETs.

VM network access is provided via a bespoke physical-server-based virtual switch ("VSwitch") called VFP [5]. Each VFP instance has several virtual ports; one is connected to a physical NIC and the others to VM virtual NICs. An outbound VM NIC packet is transformed by per-port processing layers, each with distinct tasks (like metering traffic or implementing customer ACLs [5]). One layer translates CAs to an IP "physical address" ("PA") that is routable on the underlying network, providing VNET isolation (VFP translates PAs to CAs for received packets as well). CA $\Rightarrow$ PA mappings are dynamic; actions like creating or deleting a VNET or VM can change mappings. Mappings reside in a reliable distributed directory. A per-server userspace agent receives updated mappings from this directory as network allocation state evolves. When a VM starts a network flow to a given CA, the CA $\Rightarrow$ PA mapping for the flow is queried from the userspace agent and cached in the kernel datapath; subsequent packets leverage this cache. Cached mappings are evicted after inactivity timeouts.

3.2 Monitoring via VNET Pingmesh

Baremetal monitoring alone does not account for VNET-specific performance anomalies. To make a VNET-level Pingmesh-like system, however, several cloud-specific challenges must be accounted for. We divide these challenges into two categories—readily apparent implementation hurdles that are foreseen and handled in the design phase, and more subtle behaviours that only became apparent once the system was deployed. Here, we discuss the former, and defer examining subtle behaviours to Section 4.

(1) **Black-box VMs.** Privacy concerns mean we may not run any software, nor collect statistics, within VMs.
(2) **Avoiding customer impact.** Probes must not be billed to, visible to, impacted by or spoofable by tenants.
(3) **Interactions with customer rulesets.** Customer ACLs supporting firewalls and gateways, or "User-Defined Routes" ("UDRs") supporting middlebox behavior, can both interfere with VNET Pingmesh.
(4) **Interactions with customer actions.** Tenants can unpredictably shutdown VMs, causing ping failures. If unaccounted for, we suffer persistent false-positive loss indications within VNETs where tenants shut down VMs during non-business hours to save costs.

To account for black-box VMs, VNET Pingmesh is implemented via VSwitch interposition. VFP injects outbound TCP-based ping packets after the tenant metering and ACL management layers. Outbound packets are invisible (and not metered/billed) to the tenant and avoid tenant ACL rules, while still being subject to the rest of the VNET processing stack. On the remote end, VFP intercepts ping packets before the tenant processes it, and sends a response. VFP also installs rules to prevent tenants from spoofing pings, ensuring that they cannot interfere with monitoring infrastructure. To (physical) switches, probes are indistinguishable from VM-generated traffic. VNET/VM churn is accounted for by post-processing collected ping data with separate systems that track VM and VM-NIC liveness. We are also investigating VSwitch mechanisms that can disambiguate between failed and administratively disabled VMs.

Certain advantages apply over physical Pingmesh. Since VNETs contain less nodes than the physical network, we maintain full-mesh CA⇒CA VNET ping statistics. While physical Pingmesh measures latency from userspace, VNET Pingmesh measures latency from the kernel. Thus, VNET Pingmesh probes that hit the mapping lookup cache are not subject to context switching and scheduler variation, and can more accurately measure latency than physical Pingmesh.

4 FAULT DETECTION

VNET Pingmesh indicates good overall performance, under normal circumstances; ≥ 94% of Azure VNETs meet or exceed latency requirements ≥99.999% of the time when considering per-VM 5-minute averages, and Azure achieves just under five-9's connectivity across every VNET in a typical hour. Despite favorable high-level metrics, large network scale makes faults inevitable, and so we must be certain of monitoring accuracy. Thus, we examine VNET Pingmesh's fault detection effectiveness, focusing specifically on precision and recall.

4.1 Fundamental blind spots impact recall

Like physical-layer Pingmesh, VNET Pingmesh has had a good track record detecting customer-impacting network anomalies within its purview. For example, in March 2017, a datacenter incident prompted various clients to raise support requests complaining about high VM-to-storage latency. While storage was investigated initially, a correlated VNET Pingmesh latency spike suggested a network cause, despite a lack of physical layer and VM-level monitoring alarms. Later correlation with physical Pingmesh along with address resolution failures during the affected time-period confirmed a network fault, where the impact magnitude flew under physical-layer monitoring detection thresholds. Investigations revealed a congestion-causing linecard misconfiguration root cause. Another incident involved customer VM connectivity issues, confirmed through a transient but significant VNET Pingmesh connectivity drop. Correlation with other monitoring systems identified a ToR reboot root cause.

Despite these successes, we cannot fully quantify recall since VNET Pingmesh fundamentally includes coverage blind spots. First, our system cannot fully monitor hybrid client networks containing VMs within Azure as well as client servers outside of Azure, which necessarily cannot run VNET Pingmesh infrastructure. A side effect of the inability to monitor the entire network is that VNET Pingmesh cannot monitor routing tunnels (e.g. middleboxes). Thus, rather than ascertaining the specific fate of customer traffic by mimicking their routing behaviour exactly, we instead are limited to measuring the health of the virtualization infrastructure in a point-to-point manner only. Second, while active probes can provide indications of queueing or widely-impacting network losses, they can miss 'gray faults' that only impact unpredictable subsets of traffic [3, 9, 12, 18]. For such cases, supplementing VNET Pingmesh with passive monitoring techniques [6, 12] may prove useful.

4.2 Cross-layer aliasing impacts precision

In its early days, VNET Pingmesh frustrated operators due to wasted effort diagnosing alerts with no customer impact, potentially masking actual performance issues. This reduction in monitoring precision stemmed from difficulty *interpreting* collected data. We discuss these cases and how we changed our usage of VNET pingmesh to successfully account for them.

Effectively interpreting Pingmesh statistics can convey significant insight into network health. Physical latency can be explained by locality (inter-pod ≥ intra-pod), queuing delay (O(μ-seconds)) and packet loss (O(milliseconds)). Deviations from baseline performance may reveal faults [2, 9, 12]; p99 latency ≥ TCP RTO could indicate intermittent silent packet loss, while p50 latency of O(milliseconds) could indicate persistent switch queues [9].

VNET Pingmesh also tracks latency and loss. However, we discovered that different layers of the Azure stack impacted our measurements in ways that were hard to tease apart. We call these interactions *cross-layer aliasing*, where different actors (either in Azure, or the tenants themselves) perform (possibly non-network) actions that impact networking metrics and complicate their interpretation. Until aliasing is accounted for, we can receive confusing outcomes that both complicate determining if a problematic measurement indicates an actual customer issue (and not a false positive) and risk masking actual issues. As a case study, we consider latency. Figure 1 depicts VNET latency in a cluster over 1 hour, projected to physical ToR. Cells depict source (x-axis) to destination (y-axis) ToR latency, measured as the average for all VMs in all VNETs in all servers in the source. We see that:

(1) Latency ranges from O(100s-1000) microseconds, higher than expected for datacenters. Intra-rack latency is ≥ 2.5× inter-rack latency on average.
(2) Some racks (darkly-colored intersecting horizontal/vertical stripes) show ≥ 1.5× median rack latency. Conversely, others exhibit ping latency averages an order-of-magnitude smaller than other servers.
(3) Some servers (omitted for space) exhibit outbound latency (pings generated by the server) ≥ 2× inbound.

While higher VNET latency can correlate with physical-layer faults, queueing or locality, diagnosis revealed that VNET architecture and large (≥ 30 VMs) tenant traffic patterns were responsible—specifically, kernel CA ⇒ PA mapping cache misses in large VNETs. Since VNET Pingmesh iterates through CAs consecutively, large VNETs with sparse traffic matrices may lead to cache misses for a given ping, unless the VM was actively communicating with the pinged VM (upon which measured latency would drop to the expected O(10s of microseconds) latency). High average latencies can thus be explained by large VNETs—as a large fraction of measured latencies are due to large VNETs, their performance dominates. High intra-rack latency can also be explained by large VNETs; since VM placement within a cluster is random, pinging a VM in the same rack is more likely for large VNETs. Figure 2 reveals that large VNET latency is higher, cluster-dependent, and noisier. Latencies range from 100s of microseconds to a millisecond depending on cluster, with a deviation on par with the total latency for small VNETs. Locality and latency trends vanish—instead, jitter from mapping lookup context switches dominates.

Cold (dark green) stripes in the latency heatmap correspond with either servers handling VMs within small VNETs exclusively (if the VNET size is small enough, mappings will always be within the cache due to VNET Pingmesh) or VMs in large VNETs that constantly communicate with other VMs (keeping the mapping cache hot). In other words, the tenant traffic pattern influenced the measured latency as well, by driving it down for certain VM pairs while mapping latency drove it up for other pairs.

In the initial VNET Pingmesh implementation, these two competing aliasing effects—neither signifying physical or VNET dataplane latency—caused VNET Pingmesh to report high server-average latencies that were heavily influenced by mapping latencies for *non-communicating* VM pairs. While mapping latency *is* important, especially for large-scale sparse/intermittent traffic [11], it must be tracked separately from fast-path dataplane latency. By overloading our latency signal with both modes of operation, we risk both masking actual poor network performance (O(100s of microseconds) congestion-based latency masked by millisecond-plus mapping latencies) and yield false alerts on high latency that do not correspond with tenant impact (since they are triggered by non-communicating VM pairs).

Once the issue was identified, fixing monitoring was trivial—we send multiple consecutive pings. The initial ping is saved as a (new) mapping latency metric, while the average of the remainder is saved as the (intended) dataplane latency metric. After accounting for aliasing, VNET Pingmesh behaviour comports with physical, as seen in the generally $\leq$ 100 microsecond and locality-correlated small-VNET dataplane latency measurement in Figure 2.

Some takeaway lessons apply. First, cross-layer aliasing can reduce both monitoring precision (e.g. false-positive latency alarms) and recall (e.g. congestion masked by mapping latency). Interpretation suffers when aliasing causes measurement cross-contamination from non-network causes (we later see how server-based SDN virtualization can cause such cross-contamination). Second, while accounting for aliasing is easy post-diagnosis, diagnosis itself can be complicated by cross-layer aliasing, as discussed next.

5 FAULT MANAGEMENT

Detected faults require management: triage (is the network or the virtual overlay faulty?), diagnosis (e.g. is the network experiencing packet loss?), root-causing (e.g. why is there packet loss?) and mitigation (e.g. can I take another path?). Here, we show how VNET Pingmesh can aid triage when used with cross-layer monitoring; however, its effectiveness in aiding both diagnosis and root-causing is impacted by cross-layer aliasing.

5.1 Cross-layer monitoring aids triage

At a high level, we seek to determine whether we can use VNET Pingmesh and other (different layer) monitors in conjunction to appropriately triage faults: Is a given fault due to the underlying physical network, or is it a VNET specific ailment? We focus on the minority of servers with higher than usual VNET latency, and the relationship between concurrent VNET and physical-layer Pingmesh measurements for these servers.

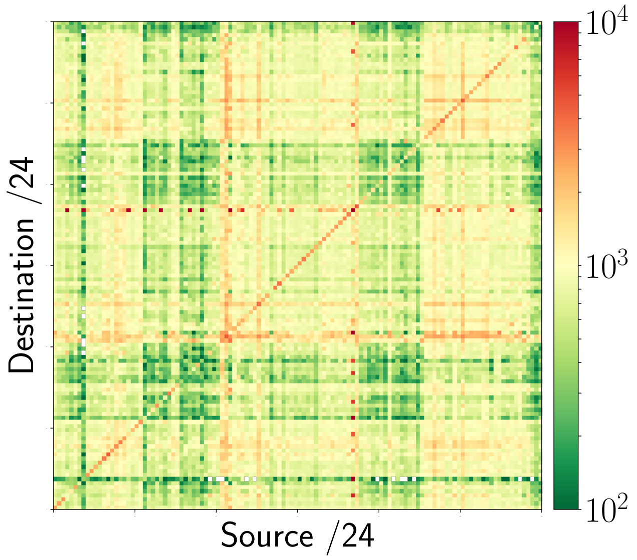

Figure 1: Rack-level latency heatmap (microseconds)

Table 1: Probability that physical layer Pingmesh $\geq$ 1.5× cluster average if VNET latency $\geq$ 2× average for a VM.

Category	% incidents	% physical anomalous
Total	100	40.90
Large VNETs only	59.00	23.50
Small VNETs only	22.60	55.00
Large and small VNETs	18.40	79.00
Server->ToR packet loss	2.20	92.60

Over a randomly picked hour with no active network alerts, we examined servers with $\geq$ 1 VM with average outbound ping latencies of $\geq$ 2× their VNET average. Table 1 categorizes servers depending on if their poor-latency VMs belong to large VNETs only, small VNETs only, or a mix of large and small VNETs; specifically, it depicts the probability that physical Pingmesh also reports latency anomalies in each case. We call physical latency anomalous if the average server ping latency is $\geq$ 1.5x the cluster average.

Large and small VNETs are concurrently impacted ≈20% of the time; physical Pingmesh is also impacted in ≈80% of these cases. Here, it seems reasonable to suspect a physical networking issue impacting all monitoring layers at or above it. Conversely, ≈60% of high latency indications impact large VNETs only; physical Pingmesh is only impacted 23.5% of the time here. These statistics are from the original VNET Pingmesh mechanism which tracks mapping latency for large VNETs; thus, one may surmise that the ≈77% of servers with VNET-only latency increases may be subject to CPU or IO contention slowing down map operations.

However, confounding factors remain, due to the generally disjoint network coverage provided by VNET and physical-layer

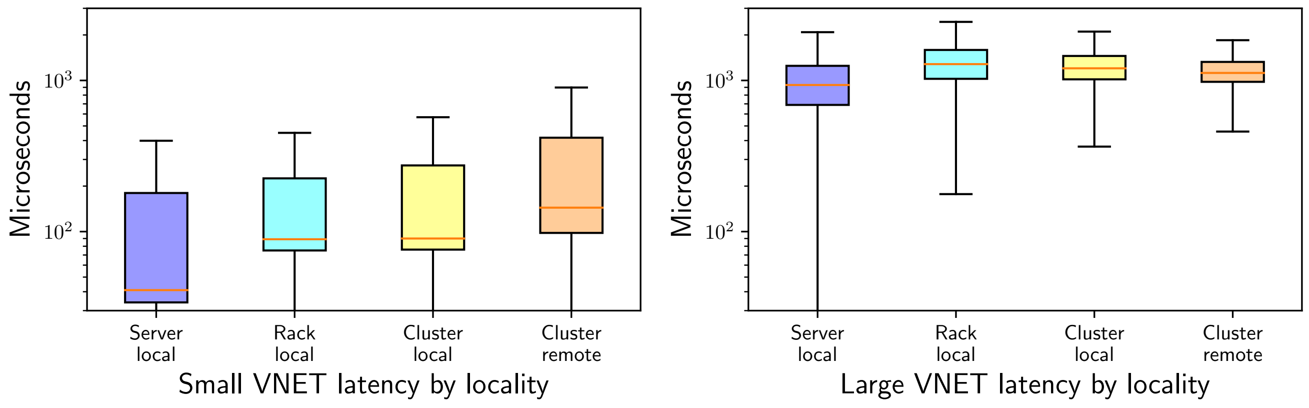

Figure 2: VNET latency vs. locality

Pingmesh at a given server. While VMs in a VNET can be spread across datacenters in a region, a server only maintains a full physical Ping mesh with others in the same rack [9]. Additionally, ping packets likely take different paths through the network core. Thus, if a network fault that would impact physical and VNET pings alike is several hops away from a server, it is possible that physical Pingmesh may miss a problem that VNET Pingmesh notices, or vice versa. Conversely, if a fault is located at the server to ToR uplink (2.2% of examined incidents) we see that physical Pingmesh is impacted 90+% of the time.

These results suggest that we can use a simple heuristic for triaging latency anomalies—if only large VNET mapping latency is impacted, we may investigate VNET infrastructure and server metrics first; if multiple VNETs of differing sizes suffer, we may suspect the physical network.

5.2 Cross-layer aliasing complicates diagnosis

While monitoring oddities may be visible at a glance from VNET Pingmesh, cross-layer aliasing and tenant behaviours can complicate anomaly diagnosis. In one instance, several VNETs (large and small) revealed unpredictable but persistent and total ping losses for certain VMs, split amongst subnet lines. While this may indicate ToR or pod connectivity loss in the physical network, this is unexpected at the VNET layer where VMs with adjacent addresses are unlikely to be physically adjacent. Confusingly, while connectivity failures were relatively long-term (hours to days), they were unpredictable; only certain VMs in a VNET were afflicted, and connectivity loss had a chance of spontaneously resolving. Furthermore, despite alarming connectivity statistics for these VNETs, no customer issues were raised.

Investigations revealed that these losses were due to a monitoring bug affecting just probe packets. Specifically, 'UDRs' (Section 3.2) were, for some VNETs, stealing Pingmesh responses—resulting in subnet-level loss for VNETs with tunnels. While pings arrived at the destination, ping responses were intercepted by tunnel rules and redirected to an unrelated remote server that discarded them.

Infrequent problem incidence confounded diagnosis; a minority of VNETs possess UDRs, a minority of which triggered ping loss. Furthermore, UDRs are subject to tenant modification. Before we diagnosed the bug, changing rule sets would resolve individual faults, leaving us at the mercy of tenant behaviour to examine the problem. Ultimately, a related bug (leaking ping packets between paired VNETs) revealed a VSwitch rule priority bug that was also the root cause for connectivity loss within non-paired VNETs.

Thus, cross-layer aliasing complicates VNET-Pingmesh-driven diagnosis. Two takeaways emerge; (1) that cross-layer interactions can complicate diagnosis in terms of timing and reproducibility of bugs, and (2) that again, the mechanical effort required to fix the bug was relatively simple, once diagnosis had occured. Here, we simply modified VNET rulebase generation to prevent probes from being processed by UDRs, thus ensuring that VNET Pingmesh measured point-to-point non-tunneled latency.

5.3 More difficult root-causing and mitigation

We distinguish between diagnosing and root-causing faults; e.g. a diagnosis may reveal that connection timeouts are caused by lost packets for a subnet at a switch, while root-cause analysis may reveal the causative ACL misconfiguration. For physical-layer faults, this distinction may be unnecessary where simply rebooting a device may clear the underlying error [14]. Alternatively, traffic may be re-routed around a fault under examination [10].

However, VNETs and smaller tenant deployments can complicate mitigation. Unlike a network core fault, we cannot route around the only VSwitch providing VM network connectivity. While this may be akin to ToR failures in traditional datacenters, large-scale services in such networks may be more amenable to losing capacity or migrating application workloads [1, 11] compared to smaller cloud tenants. Similarly, rebooting physical servers to possibly clear faults may inacceptably impact availability. While VM live migration may provide respite to a tenant, it may not solve the underlying problem and prevent another tenant from being impacted down the line. Thus, effective root-cause analysis takes on greater importance in cloud networks; unfortunately, it can also be complicated by cross-layer aliasing.

We re-examine VNET mapping latency measurements in the context of root-cause analysis. A small minority of servers exhibit slow mapping lookups of $\geq$ 10 msec in length. As a userspace operation, address mapping is subject to scheduler-variation effects and thus spikes in CPU utilization and (as we will discuss) IO bus contention. To quantify this effect, we examine the prevalence of high latency (specifically, the likelihood that the server-average VNET latency $\geq$ 2 msec) as a function of server disk I/O and CPU utilization in Figure 3. Both graphs are normalized to the baseline probability at nominal utilization levels. We see a clear correlation between utilization and latency; as average disk utility passes 10%, we see a sharp increase in likelihood that latency is past acceptable boundaries. CPU utilization is measured as a 5-minute average, however, and so we cannot distinguish between a server CPU constantly at 5% utilization (unlikely to impact VNET latency) or at 100% utilization $\approx$5% of the time—which would significantly impact latency during that period. Even so, a (weaker) correlation between CPU utilization and high latency emerges.

Since both CPU and IO utilization impact mapping latency, a large variety of root causes can apply. In one case, a latency alert correlated with an internal OS-update roll-out. Investigations showed that sustained disk I/O was interfering with userspace address mapping lookups due to a logging statement blocked on disk I/O. While small VNETs and existing connections (both in-cache) were not impacted, connections requiring a userspace mapping lookup were. (Client disk traffic was not impacted due to storage disaggregation.) Subsequent analysis across Azure showed a correlation between server I/O utilization and high (false-positive) latency indications. In another server, a pair of processes briefly spiked CPU utilization every minute, periodically impacting mapping latency. Yet another server had high disk utilization due to a mysteriously large number of sync operations. In other incidents, Azure-infrastructure driven disk I/O caused synchronized latency spikes of $\leq$ 1 minute spread across geographically-disparate clusters. Thus, a large variety of root-causes impact the same monitoring signals, yielding an aliasing problem where monitoring metrics may not provide enough insight on the underlying problems. Since these incidents have different and possibly invasive fixes (e.g. killing a service with runaway CPU utilization), blindly and optimistically trying fixes is inacceptable—we must accurately root cause the issue first. Thus, tenant virtualized networks simultaneously increase the importance of, while simultaneously complicating, root-causing faults.

6 CONCLUSIONS

Several open problems remain. First, while we have dealt with cross-layer aliasing in an ad-hoc manner—identifying measurement oddities as they crop up, diagnosing and root-causing their source and accounting for them—the ever increasing feature sets for tenant virtualized networks both increases the future likelihood of such interactions and complicates the task of searching the state space of all possible interactions *a priori*. A systematic methodology for avoiding such interactions or accounting for them during the design phase would be ideal. Second, while characterizing precision and recall are important for validating monitoring effectiveness, we find it hard to characterize them in the absence of customer impact ground truth. This is due both to the fundamentally incomplete coverage that a system like VNET Pingmesh is capable of providing in the presence of hybridized networks (networks split between Azure and private customer resources) and due to the fact that customers themselves may miss issues like gray faults, and that probe traffic may not be susceptible to gray faults that do impact tenant application traffic. Thus, evaluating passive methodologies that infer customer VM network performance from the VSwitch layer [6] in conjunction with active-probing monitoring systems may be a productive line of inquiry. Third, we have thus far only scratched the surface of using multiple layers of monitoring systems to perform fault triage and attribution; a more longitudinal study may reveal deeper insights.

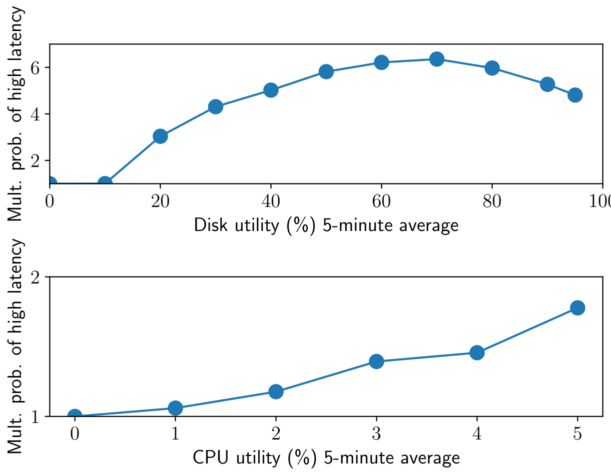

Figure 3: Multiplicative likelihood of high VNET Pingmesh latency vs. server utilization.

ACKNOWLEDGEMENTS

N. Aggarwal, M. Dasgupta, A. E. Sreenath, A. Fu, D. Firestone, S. Garg, D. Jagasia, A. Kumar, N. Motwani, J. Park, C. Raje, P. Shrivastava, A. Shukla, Q. Zuhair, M. Zygmunt, and the entire VNET and VFP teams at Microsoft provided valuable insight into Azure Networking.

REFERENCES

[1] Hadoop. http://hadoop.apache.org/.
[2] A. Adams, P. Lapukhov, and H. Zeng. https://code.facebook.com/posts/1534350660228025/netnorad-troubleshooting-networks-via-end-to-end-probing/.
[3] B. Arzani, S. Ciraci, L. Chamon, Y. Zhu, H. H. Liu, J. Padhye, B. T. Loo, and G. Outhred. 007: Democratically finding the cause of packet drops. In *15th USENIX Symposium on Networked Systems Design and Implementation*, Renton, WA, 2018.
[4] B. Arzani, S. Ciraci, B. T. Loo, A. Schuster, and G. Outhred. Taking the blame game out of data centers operations with NetPoirot. In *Proceedings of the ACM SIGCOMM Conference*, Florianopolis, Brazil, 2016.
[5] D. Firestone. VFP: a virtual switch platform for host SDN in the public cloud. In *Proceedings of the 14th USENIX Conference on Networked Systems Design and Implementation*, Boston, MA, 2017.
[6] M. Ghasemi, T. Benson, and J. Rexford. Dapper: Data plane performance diagnosis of TCP. In *Proceedings of the Symposium on SDN Research*, Santa Clara, CA, 2017.
[7] M. Ghobadi and R. Mahajan. Optical layer failures in a large backbone. IMC '16, Santa Monica, California, USA, 2016. ACM.
[8] A. Greenberg. Pingmesh + NetBouncer: Fine-grained path and link monitoring for data centers. https://atscaleconference.com/videos/pingmesh-netbouncer-fine-grained-path-and-link-monitoring-for-data-centers/, 2016.
[9] C. Guo, L. Yuan, D. Xiang, Y. Dang, R. Huang, D. Maltz, Z. Liu, V. Wang, B. Pang, H. Chen, Z.-W. Lin, and V. Kurien. Pingmesh: A large-scale system for data center network latency measurement and analysis. In *Proceedings of the ACM SIGCOMM Conference*, Aug. 2015.
[10] V. Liu, D. Halperin, A. Krishnamurthy, and T. Anderson. F10: A fault-tolerant engineered network. In *Proceedings of the USENIX Symposium on Networked Systems Design and Implementation*, Apr. 2013.

[11] A. Roy, H. Zeng, J. Bagga, G. Porter, and A. C. Snoeren. Inside the social network’s (datacenter) network. In *Proceedings of the ACM SIGCOMM Conference*, London, England, Aug. 2015.
[12] A. Roy, H. Zeng, J. Bagga, and A. C. Snoeren. Passive realtime datacenter fault detection and localization. In *Proceedings of the 14th USENIX Conference on Networked Systems Design and Implementation*, Boston, MA, 2017.
[13] A. Singh, J. Ong, A. Agarwal, G. Anderson, A. Armistead, R. Bannon, S. Boving, G. Desai, B. Felderman, P. Germano, A. Kanagala, J. Provost, J. Simmons, E. Tanda, J. Wanderer, U. Hölzle, S. Stuart, and A. Vahdat. Jupiter rising: A decade of Clos topologies and centralized control in Google’s datacenter network. In *Proceedings of the ACM SIGCOMM Conference*, 2015.
[14] X. Wi, D. Turner, G. Chen, D. Maltz, X. Yang, L. Yuan, and M. Zhang. NetPilot: Automating Datacenter Network Failure Mitigation. In *Proceedings of the ACM SIGCOMM Conference*, Helsinki, Finland, Aug. 2012.
[15] H. Zeng, P. Kazemian, G. Varghese, and N. McKeown. Automatic test packet generation. CoNEXT ’12, Nice, France, 2012. ACM.
[16] Q. Zhang, V. Liu, H. Zeng, and A. Krishnamurthy. High-resolution measurement of data center microbursts. In *Proceedings of the Internet Measurement Conference*, London, United Kingdom, 2017. ACM.
[17] Q. Zhang, G. Yu, C. Guo, Y. Dang, N. Swanson, X. Yang, R. Yao, M. Chintalapati, A. Krishnamurthy, and T. Anderson. Deepview: Virtual disk failure diagnosis and pattern detection for azure. NSDI ’18, Renton, WA, 2018. USENIX Association.
[18] Y. Zhu, N. Kang, J. Cao, A. Greenberg, G. Lu, R. Mahajan, D. Maltz, L. Yuan, M. Zhang, B. Y. Zhao, and H. Zheng. Packet-level telemetry in large datacenter networks. In *Proceedings of the ACM SIGCOMM Conference*, London, United Kingdom, 2015.