Securing Public Clouds using Dynamic Communication Graphs

Sathiya Kumaran Mani[§] Kevin Hsieh[§] Santiago Segarra[§⋆] Trevor Eberl[§] Ranveer Chandra[§]
Eliran Azulai[§] Narayan Annamalai[§] Deepak Bansal[§] Srikanth Kandula[§]

[§]*Microsoft* [⋆]*Rice University*

Abstract– We leverage a novel telemetry source available in public clouds today: periodic summaries of every flow that enters or leaves any VM. A key aspect is that such telemetry can be collected transparently to customers and with minimal impact on their workloads. By consuming this telemetry, we show how one may realize complete and dynamic graphs of the communication inside cloud subscriptions. We describe novel analyses over these communication graphs with implications on network security and management.

1 Introduction

As more enterprises transition their IT workloads to public clouds, securing and optimizing the network communication within cloud subscriptions is increasingly important.

Public clouds have a communication visibility problem however. For traffic in/out of a subscription, one may deploy firewalls or IDSes [7, 10, 20] at a network chokepoint. However, for traffic within the subscription, customers do not have a view on who talks to who, when or why in their cloud deployments. There is no one chokepoint that has a complete view of the internal communication.

Furthermore, a typical cloud deployment uses many resources of different kinds such as VMs, databases and lambdas. Enterprises such as Walmart [19, 30, 47], which use millions of cloud resources, have large teams that own different portions of their cloud software. Thus, synthesizing a view of the communication by examining all of the corresponding software appears less promising.

In this work, we focus on generating network-wide communication graphs such as the one shown in Figure 1. Nodes correspond to IP addresses here but can also be services, kubernetes pods or {(IP, port)} tuples. Edges represent communication between nodes, e.g., the number of packets, bytes and connections. We can generate a time-series of graphs or embed timeseries in the node and edge attributes of one graph. The novel aspects are:

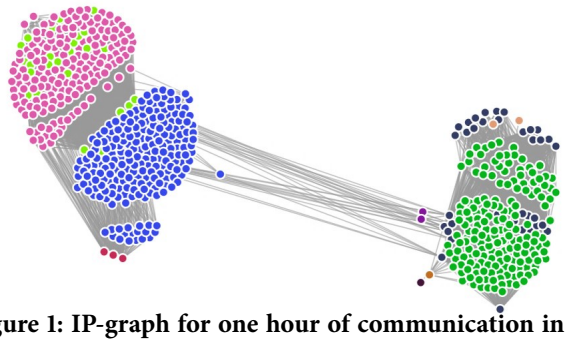


Figure 1: IP-graph for one hour of communication in the K8s PaaS **cluster. Node colors represent roles automatically inferred from communication patterns (using Jaccard score on neighbor set overlap and hierarchical louvain clustering on the scored clique).**

Complete	the graphs aim to capture all of the communication (as opposed to using one or a few packet traces from network chokepoints).
Dynamic	the telemetry is continuous so that an administrator gets up-to-date views while also being able to do historical analysis such as 'what changed?' or 'what happened during that (past) event?'
Multi-faceted	the graphs should be able to capture information at different timescales and different granularities so as to enable meaningful discovery of patterns and rich analyses.

Using such graphs, we posit that customers can identify wholistic optimization possibilities and improve protection in the presence of advanced persistent threats [1, 23] or breaches because, otherwise, even a single breached resource may open up access to many other resources in a subscription [68, 69].

Previous works have discussed network-wide dynamic monitoring and analytics systems, so what's new? In ISP networks, at AT&T, Gigascope[40–42] discusses a network-wide collection and analysis of netflow data from switches. In datacenter networks, Facebook [66] and Microsoft [56], discuss deploying agents at every end-host in their datacenters and analyzing the logs using data analytics systems. We are not aware however whether any of these prior systems were deployed widely,

HotNets '23, November 28–29, 2023, Cambridge, MA, USA

ACM ISBN 979-8-4007-0415-4/23/11…$15.00
https://doi.org/10.1145/3626111.3628198

	#IPs mon.	Graph Size: #nodes (#edges) IP Graph	IP-port Graph	#Records /minute
Portal	4	4K (5K)	13K (13K)	332
μserviceBench	16	33 (268)	0.2M (1M)	48K
K8s PaaS	390	541 (12K)	1.3M (3M)	68K
KQuery	1400	6K (1.3M)	12M (79M)	2.3M

Table 1: Cloud clusters and some aspects of their communication graphs that we built and analyzed in this paper. The source telemetry is flow-level summaries every minute with schema as in Table 2.

or for a sustained period or have had net positive revenue. In this paper, we offer some reasons why the converse can happen in public clouds. That is, sustained and wide deployments with a positive revenue. Our case rests on three aspects:

- Minimal Interference to customer workloads: That is, the benefits of visibility such as enhanced security, diagnosability and manageability do not come at the cost of worse performance. A key to this is if the loggers have no effect on the resources that a customer pays for in the cloud (e.g., a VM's cores, memory and network bandwidth).
- Low COGS[1]: The market will not support a high surcharge for enhanced security, diagnosability and manageability. If an average VM costs 0.5$/hr[2], a likely price-point that the market will bear is 0.02$/hr/VM for a surcharge of about 4%.[3]
- Appealing use-cases that are likely to be revenue positive. We will discuss security, especially micro- segmentation, succinctly summarizing communication patterns and counterfactual reasoning in this paper.

We are in the early stages of building and deploying a system that generates and analyzes cloud communication graphs. In §3 we will describe a telemetry stream that is available in three large public clouds today and our analytics system that consumes the telemetry. Our goal here is to keep COGS low and have zero impact on customer workloads.

Table 1 depicts a few example deployments and aspects of the corresponding communication graphs. Portal is the web portal for a large cloud (e.g., http://aws.amazon.com); we analyze all of the communication at one of the many geographically distributed clusters that serve web requests to the portal. μServiceBench [13] is a publicly available benchmark that mimics a set of micro-services running a shopping site. Here, we use synthetic load generators and inject a wide-range of attacks [15]. Note, this cluster has much more traffic between VMs and so many more edges in the hourly IP-graph relative to the number of nodes. K8sPaas cluster is a production kubernetes-as-a-service cluster; that is, customers can deploy their helm charts and pods while the cloud provider manages the cluster. Note, the many more VMs here and the correspondingly larger graph sizes. We use this cluster as the default for all our analyses. KQuery is a cluster that runs SQL-like queries on (mostly) memory-resident datasets at a large cloud provider.

In §2, we describe our initial attempts at three kinds of graph analyses. We list open issues. In some cases, while the requirements and potential approaches are clear, we do not yet have a full practical realization. Prior work, on social network and product recommendation graphs, describes some similar graph analyses [65]. However, as we discuss further in §2, cloud communication graphs present a sharp contrast because: (a) one communication trace may be represented as many different communication graphs based on whether nodes should be IPs, services or {IP,port} tuples for example and while prior work can analyze a particular graph, choosing which graph to construct requires networking insights and (b) the graph analyses methods appear to require domain customization because different features are available in networking graphs and the underlying phenomena that gives raise to the learnable patterns is different.

To sum, our thesis is that complete and dynamic cloud communication graphs can be feasibly constructed at scale and in a cost-effective, private and secure manner. Having these graphs uniquely enables some novel analyses and, importantly, lets us build novel security primitives such as segmenting cloud graphs to enhance security.

2 Graph Analyses

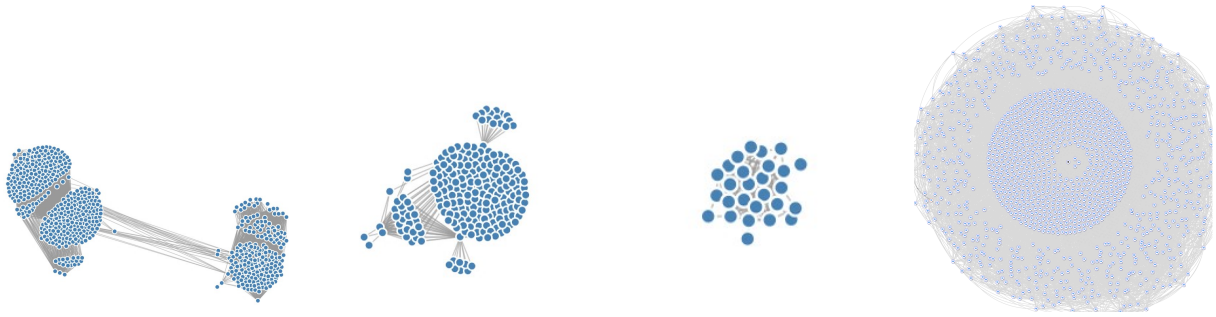
We discuss a few techniques that are uniquely enabled by complete and dynamic communication graphs.

2.1 MicroSegmentation

Micro-segmentation refers to fine granular protection of resources inside cloud subscriptions. Since a subscription can have multiple VMs, databases and other resources, we aim to mitigate the blast radius when any one of those resources is breached. That is, we seek to limit which other resources may become vulnerable due to the breach. Our approach is to divide resources into, so-called, μsegments and author reachability policies between the μsegments. A pair of resources can communicate with each other only if explicitly allowed by the policies; i.e., the default will be to deny [31, 37]. By doing so, the blast radius of breaching a resource reduces to only those that the resource must communicate with during normal operation. A few vendors have released μsegmentation offerings [27, 39, 53, 70] and the market for μsegmentation is estimated to be a few billion dollars in annual revenue and growing at +20% year-over-year [26].

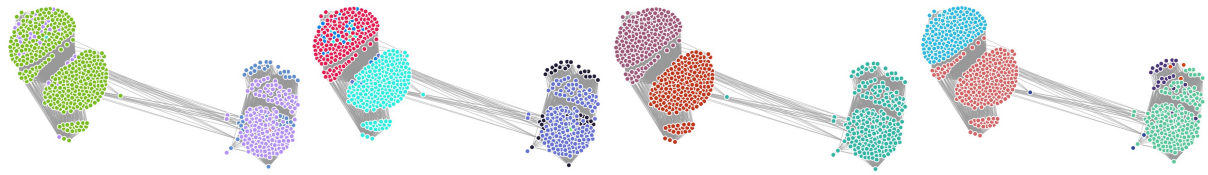
Micro-segmentation is challenging due to a few reasons. Consider the (unsegmented) IP-level communication graphs

[1] cost of goods and services
[2] 8 core VM at Azure say
[3] based on a pricing analysis of Illumio [53] and others [27, 39, 70].



(a) K8s PaaS IP-graph (b) Portal IP-graph (c) μserviceBench IP-graph (d) KQuery IP-graph

Figure 2: Unsegmented IP-graphs for the three datasets in Table 1



(a) Simrank segmentation (b) Simrank++ segmentation (c) Conn.-weight. modularity (d) Byte-weighted modularity

Figure 3: Applying other segmentation strategies, inspired by prior work, on K8s PaaS's IP-graph

for each of the three examined clusters in Figure 2.[4] The state-of-art invites a human administrator to manually tag each node with a μsegment label. It is not apriori clear how well a human would perform on this task. Worse, when the role of a resource changes– for example, when pods in kubernetes migrate or scale up or down [14] or when a software change causes VMs to behave differently– the μsegment labels must keep up-to-date.

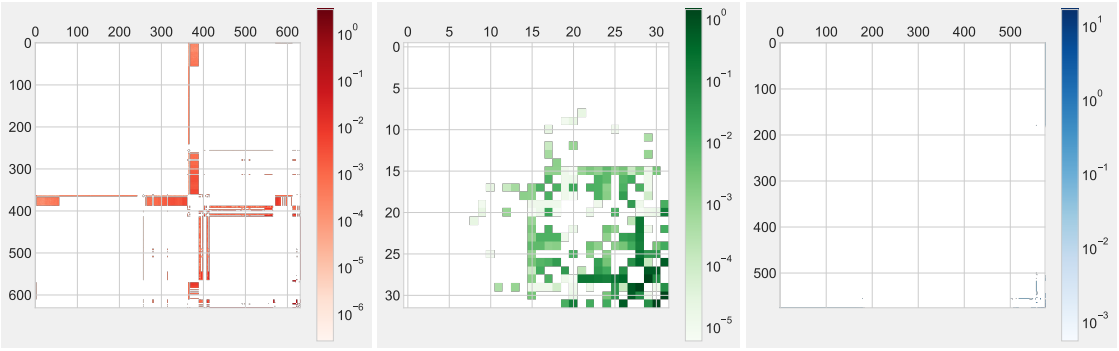
Could an algorithm assist with μsegment labeling? Fundamentally, there are many fewer roles than resources in a cloud setting because, for redundancy and scalability, it is quite common to have multiple resources play the same role. Moreover, we find that the role of a resource can be inferred based on its communication, such as which neighbors a resource communicates with and the nature of the conversation (e.g., the time series and numbers of bytes, packets and connections exchanged with each neighbor). Figure 1 shows the result of a simple segmentation.[5] In Figure 1, nodes that share a color have the same role and can be placed into a μsegment.

Open issues: The auto-segmentation logic used above has super-quadratic complexity. Furthermore, developer interviews show that while the labels are a good start there are key mistakes. How can we improve accuracy and reduce cost? Intuitively, auto-segmentation is akin to the role inference problem in graph mining literature [51] and Figure 3 shows the results from other popular techniques. Comparing with Figure 1, the results clearly differ. The reason is that, intuitively, modularity-based clustering [33] groups nodes that exchange a lot of data with each other but, in communication graphs, nodes with the same role such as the *front-end VMs* may never talk to each other. Other clustering metrics have similar downsides. SimRank [54] is a recursive algorithm wherein the similarity score for a pair of nodes is a weighted sum of the similarity of the nodes' neighbors. Uniquely, recursive techniques can learn roles that are not immediately obvious based on the communication of individual nodes. SimRank++ [28, 60] extends SimRank to the case of weighted edges. Both SimRank and SimRank++ have higher complexity than the simple segmentation above but, in our initial experiments, did not yield higher quality results. Overall, we believe the auto-segmentation logic that is best suited to cloud communication graphs remains an open question due in part to the following concerns. (1) The underlying phenomena that leads to role similarity in cloud communication graphs – same code running in multiple resources for redundancy and scalability – is not a consequence of human dynamics unlike the case of social graphs and product recommendation graphs [51]. (2) Resources may have multiple roles, for e.g., a VM may run multiple services. Thus, segmenting IP-port graphs may be more useful but these graphs can be much larger than IP-graphs. (3) The ideal auto-segmentation algorithm may vary across subscriptions or as roles evolve over time and so a learnt algorithm that adapts based on feedback may yet outperform all static segmentations.

Setting aside how μsegments are labeled, enforcing reachability policies between μsegments imposes additional stress on

[4] IP-port graphs are at least one order of magnitude larger.

[5] We score each pair of nodes based on the overlap in their neighboring sets (Jaccard score [35, 45]). Then, we cluster the clique where edges are weighted by the overlap score using the Louvain method [33].



(a) K8s PaaS (b) μserviceBench (c) Portal

Figure 4: For the datasets in Table 1, adjacency matrix representations of the bytes exchanged between IP addresses. The matrix entries are byte counts normalized and color-coded in log scale.

network virtualization systems. Clouds today limit the number of rules that can execute on the path in and out of each VM (e.g., no more than 10^3 rules at a VM) and naïvely unrolling reachability rules between μsegments into reachability rules between IP addresses, which current clouds support, can lead to rule explosion. Adding dynamic tags into packets and extending the network virtualization layer to enforce policies on tags is a potential solution. Tags may also help reduce churn and lag when μsegment labels change.

Although today's cloud providers only support reachability policies, using higher order policies can lead to fewer false positives and better outcomes. For example, suppose a code change causes VMs in a μsegment to begin speaking with a new service that they did not speak with before. A reachability policy will flag this as a violation. However, noticing that all of the VMs in the μsegment continue to exhibit similar behavior, even though the behavior has changed, may avoid the false positive. We call these *similarity*-based policies. Analogously, *proportionality*-based policies, which consider the amount of traffic between different pairs of μsegments, can distinguish between changes that are explainable due to a flash-crowd (e.g., more traffic to the backend when more requests are incoming) versus other changes (e.g., more traffic to the backend by itself). It is not yet clear to us which kinds of policies are worthwhile to use and efficient to implement.

2.2 Succinct Summaries

We find that cloud communication graphs are exceedingly sparse. Using principal component analysis (PCA), we can sparse transform a matrix by just using the first k eigen vectors. That is, for a square matrix M, PCA computes matrices E and D which represent the eigen vectors and eigen values on the diagonal such that $M = EDE^\top$. Now, if M has n rows, let E_k be the $n \times k$ matrix containing just the first k eigen vectors, D_k be the $k \times k$ matrix with just the first k eigen values, then we denote the k'th sparse transform as $M_k = E_k D_k {E_k}^\top$. We compute reconstruction error $\mathsf{ReconErr}(M, M_k)$ as the normalized absolute sum of entries in $M - M_k$. Clearly, $M_n = M$, that is, using all eigen vectors will perfectly recover a matrix. However we find that many fewer eigen vectors suffice for a low reconstruction error in the considered communication graphs. For example, in the K8s PaaS dataset, using just $k = 25$ eigen vectors ($n > 500$ in this case) leads to a less than 0.05 error. This means that on average each entry in the reconstructed matrix is within 5% of its true value.[6]

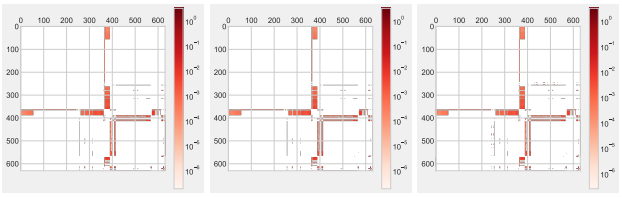
The adjacency matrices in Figure 4 show that cloud communication graphs exhibit some clear patterns. Rows and columns are IP addresses and the color of the matrix entries, in log scale, represent the amount of bytes that are exchanged. We call out a few patterns:

- **Chatty cliques:** subsets of nodes that exchange large amounts of data among each other.
- **Hub and spoke:** some nodes exchange a large amount of data with many other nodes. Hubs are likely to be control plane components such as job managers, k8s api servers [14], cloud stores or telemetry sinks.

Similar patterns exist in graphs from different subscriptions likely because these patterns arise fundamentally from software practices.

Open issues: Beyond the simple PCA and visual analyses above, we ask whether it may be possible to learn a generalizable model to classify cloud communication patterns. That is, a model pre-trained over many communication graphs which a customer can apply off-the-shelf on their communication graph to identify the canonical patterns in their network. A key advantage from such a model could be to offer executive summaries such as '80% of the bytes in your network are doing X'. Such generalizable models exist for images (e.g., RESNET[50])

[6] Similar results hold when using independent components, e.g., FastICA [8], instead of PCA's eigen vectors.



(a) Hour +1 (b) Hour +2 (c) Hour +3

Figure 5: Timelapse of bytes exchanged on the K8s PaaS dataset; the matrices here represent communication in three consecutive hours immediately after the hour shown in Figure 4(a).

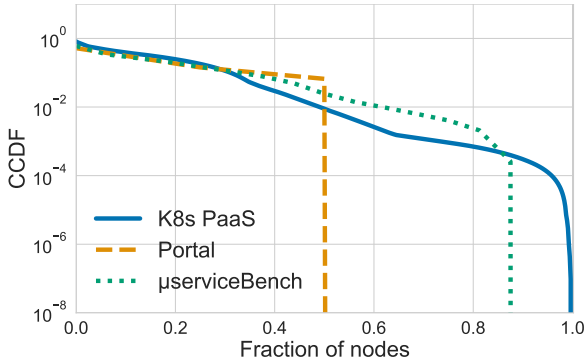


Figure 6: Where to invest more capacity? This CCDF of bytes exchanged in a communication graph versus the number of nodes participating in the exchange shows that a few nodes account for most of the traffic.

and text (e.g., BERT[44]) but we are unaware of any in the networking context.

We ask whether it may be possible to convert such a summarization model into an anomaly detector. That is, a model that can capture the key patterns may also be able to identify when the patterns change. Consider Figure 5 which shows the matrices for three subsequent hours after the matrix in Figure 4(a) on the K8s PaaS dataset. While there are some changes– some bands shrink or grow in intensity and a few appear only during some hours, many patterns are consistent.

How may we build such a general summarization model? Using graphical neural network based auto-encoders is a possibility. Generalizing across clouds can be a challenge because their graphs can have very different sizes and node degrees and one must quantize carefully because a generalizable model takes fixed sized inputs [44, 50].

2.3 Counterfactual Analyses

Notice that the connection summaries can be converted into distributions of flow sizes and inter-arrival times (quantized to the frequency of summaries). Thus, the dataset enables rich

Time	Local		Remote		#Packets		#Bytes	
	IP	Port	IP	Port	Sent	Rcvd.	Sent	Rcvd.

Table 2: Schema of connection summaries.

	Azure	AWS	GCP
Name	NSG Flow Logs	VPC Flow Logs	
Agg. Intrvl	1 min	1 min	5s or higher
Content	As illustrated in Table 2		
Sampling	N/A	N/A	3% of Pkts, 50% of Flows
Price	about 0.5$/GB to collect		

Table 3: Details on available connection summaries at three large cloud providers.

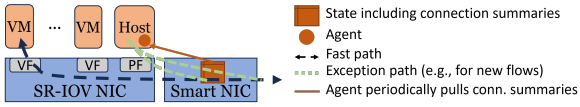


Figure 7: Illustrating how one can capture connection summaries with zero impact to VMs in a public cloud.

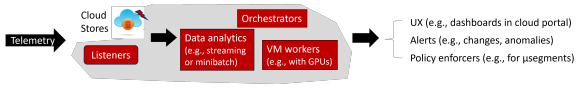


Figure 8: An example analytics system architected as a software-as-service (SaaS) that can adapt to load.

counterfactual reasoning. For example, [71] learns a mathematical model that can offer flow completion time distributions given flow size and arrival information. Figure 6 describes a simpler analysis: where are the communication bottlenecks in a cloud subscription? This analysis can show an administrator where to invest more capacity (by changing the VM SKU). As well, an admin can relocate VMs that exchange a lot of data into the same availability zone or a proximity group to improve performance [34, 38].

3 Low COGS telemetry, analytics

3.1 Telemetry Source

At three large cloud providers, we can collect connection summaries periodically with minimal impact to customer workloads [9, 11, 18]. Tables 2 and 3 show the data format and some salient details. Figure 7 describes a method to collect such summaries with low impact; we are unaware of any prior description of how to obtain such telemetry.

The key idea, as shown in Figure 7, is to record connection summaries on the programmable NICs that are directly attached to all hosts in public clouds. These programmable NICs serve various network functions today [4, 5, 12, 32, 57, 64]. Connection summaries can be recorded in smartNIC memory and an agent on the host can periodically pull and forward summaries to a cloud store or a service endpoint. The size of the logs and the memory footprint is proportional to the number of concurrent flows. These cards already maintain per-flow state [24, 32] and so recording a few counters is a

small additional burden. The same functionality can also be implemented in the software stacks that implement network virtualization [43]. Some providers sample packets and flows to further reduce cost [11]. The telemetry can be stored privately. and customers cannot tamper with collection which, crucially, ensures that telemetry is usable even when VMs are breached. Importantly, observe that there is little impact to customers; the possible performance interference from updating a few counters is negligible relative to all of the other network function processing that happens in network virtualization [24, 25, 43, 48].

Relative to capturing packet traces [21], logging in the kernel [56, 66] and sampling packets on switches [22], the method in Figure 7 has a few advantages. To collect packet traces, the network stack must copy packets and processing packets is costly especially at NIC speeds going to 400Gbps. Packet processing elements in switches typically do not have memory to spare to record per-flow state [16, 17] and so, they sample packets and summarize at the control CPUs [22, 59]. Network stack hooks such as eBPF are useful except that deploying agents on every VM in a public cloud is daunting because customers may use different OS types and providers have limited control on guest OSes. Even if such agents were deployed, their telemetry can be tampered by malicious code in the VM and cannot be relied upon during breaches.

Open Issues: The telemetry here is blind to application-level information such as process name or service identity. While such information is useful (e.g., to analyze inter-service communication), it is not clear how to capture with low impact.

Also, pushing sketches into programmable NICs may be needed to capture information that is absent in a connection summary such as burst statistics.

3.2 Analytics System

Using systems that are available in most large public clouds today, we ask if one can build an analytics system that can analyze roughly 1000 VMs worth of telemetry (e.g., connection summaries at one minute granularity) using a handful of VMs worth of resources? This is roughly a 0.5% surcharge, if you will, and low COGS is crucial for practical viability.

The telemetry volume (e.g., #records/min in Table 1) can be supported by streaming systems [36] or by using mini-batches [2, 3, 6]. A key issue is to factor the graph analyses in §2 into parallelizable in-memory execution plans and add GPU support for the learnt components. Figure 8 sketches a potential SaaS architecture.

As a case-study, consider generating communication graphs from a live stream of connection summaries. Naïvely, this is a group-by-aggregation query. That is, accumulate the byte, packet, and connection counts between pairs of nodes. The memory need is proportional to the number of node pairs in the graph. As we saw in Figure 4, connectivity is sparse but there may be many remote IPs. Also, IP-port graphs will have more nodes. One potential mitigation is to focus on the heavy hitters. That is, remote IPs and ephemeral ports that do not individually account for a sizable share of traffic are collapsed together. In fact, the graph sizes in Table 1 collapse IPs contributing less than 0.1% of bytes, packets or connections into one node in the IP-graph. With this approximation and other salient changes, we can construct communication graphs on subscriptions with 1000s of VMs in realtime using just a few machines worth of resources.

Open Issues: While graph generation appears viable, can complex analyses be factored appropriately to meet the COGS constraints? Could offloading the analyses into programmable hardware on the network path help?

4 Related work

Our key contributions are as follows:

- analyses over network-wide time series of communication graphs and
- a low COGS and impact telemetry analytics system.

To our knowledge, both aspects are novel. While a large body of prior work exists on temporal graph mining [63, 65], only a few consider the graphs that arise in the context of communication networks [61]. We consider novel challenges in the networking context, such as learning novel security policies and using multi-faceted graphs, i.e., nodes being IPs, services or IP-port pairs.

We already mentioned a few systems that analyze passively collected network-wide telemetry [40, 56, 58, 66]. However, they have yet to have a wide, sustained deployment or net positive revenue. Moreover, PingMesh [49] uses active probes to help detect performance problems on the network, whereas MALT [62] organizes topology connectivity and configuration information in a graphical format. Neither measures the actual communication on the data path.

Extensive existing work examines packet traces from one or a few carefully chosen collection points [29, 46, 52, 55, 61, 67]. However, it is unclear if these techniques will generalize or scale to network-wide communication summaries.

5 Conclusion

Using connection level summaries that are available today in public clouds, we argue that complete and dynamic cloud communication graphs can be feasibly constructed at scale and in a cost-effective, private, and secure manner. We also described several novel analyses over these graphs. While gaps remain, such analyses can be an important step towards novel security primitives such as segmenting cloud graphs.

The use of production traces in this paper was governed by an institutional privacy review.

References

[1] Advanced persistent threats. https://www.cisa.gov/topics/cyber-threats-and-advisories/advanced-persistent-threats.

[2] Amazon redshift. https://aws.amazon.com/redshift/.

[3] Apache spark. https://www.databricks.com/spark/about.

[4] AWS: Data Transfer Costs for Common Architectures. https://go.aws/3cg5J3O.

[5] Azure: Bandwidth Pricing. https://bit.ly/3Cou81Z.

[6] Azure synapse analytics. https://learn.microsoft.com/en-us/azure/synapse-analytics/.

[7] Cisco Adaptive Security Virtual Appliance (ASAv) Data Sheet. https://bit.ly/3UldTJq.

[8] Fast ica. https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.FastICA.html.

[9] Flow logging for network security groups. https://learn.microsoft.com/en-us/azure/network-watcher/network-watcher-nsg-flow-logging-overview.

[10] FortiGate-VM on Amazon Web Services. https://bit.ly/3Bp5qwb.

[11] Gcp: Vpc flow logs. https://cloud.google.com/vpc/docs/flow-logs.

[12] Google Cloud: Bandwidth Pricing. https://bit.ly/3Cw83i9.

[13] Google cloud platform microservices demo. https://github.com/GoogleCloudPlatform/microservices-demo.

[14] Horizontal pod autoscaling. https://kubernetes.io/docs/tasks/run-application/horizontal-pod-autoscale/.

[15] Infection monkey - breach and attack simulation. https://www.akamai.com/infectionmonkey/breach-and-attack-simulation.

[16] Intel tofino. https://intel.ly/3wxWT8w.

[17] Intel tofino 2. https://intel.ly/3QTeD6F.

[18] Logging ip traffic using vpc flow logs. https://docs.aws.amazon.com/vpc/latest/userguide/flow-logs.html.

[19] A new walmart 'cloud factory' will accelerate digital innovation, boost business efficiency. https://shorturl.at/amwHI.

[20] Palo Alto Networks VM-Series Firewall. https://docs.paloaltonetworks.com/vm-series.

[21] tcpdump. http://ee.lbl.gov/tcpdump.tar.Z.

[22] Using netflow filtering or sampling to select the network traffic to track. https://rb.gy/83mcu.

[23] What is an advanced persistent threat (apt)? https://www.cisco.com/c/en/us/products/security/advanced-persistent-threat.html.

[24] Vfp: A virtual switch platform for host sdn in the public cloud. In *NSDI*, 2017.

[25] Azure accelerated networking: Smartnics in the public cloud. In *NSDI*, 2018.

[26] Microsegmentation - global strategic business report. https://www.researchandmarkets.com/report/microsegmentation, 2023.

[27] Akamai. Akamai Guardicore Segmentation. https://www.akamai.com/products/akamai-guardicore-segmentation.

[28] I. Antonellis, H. G. Molina, and C. C. Chang. Simrank++: Query rewriting through link analysis of the click graph. In *VLDB Endowment*, 2008.

[29] P. Bahl, R. Chandra, A. Greenberg, S. Kandula, D. Maltz, and M. Zhang. Towards Highly Reliable Enterprise Network Services via Inference of Multi-level Dependencies. In *SIGCOMM*, 2007.

[30] J. Bailey and B. Jensen. Walmart and azure. https://shorturl.at/vMTY0.

[31] H. Ballani, Y. Chawathe, S. Ratnasamy, T. Roscoe, and S. Shenker. Off by default! In *HotNets*, 2005.

[32] D. Bansal, G. DeGrace, R. Tewari, M. Zygmunt, J. Grantham, S. Gai, M. Baldi, K. Doddapaneni, A. Selvarajan, A. Arumugam, B. Raman, A. Gupta, S. Jain, D. Jagasia, E. Langlais, P. Srivastava, R. Hazarika, N. Motwani, S. Tiwari, S. Grant, R. Chandra, and S. Kandula. Disaggregating stateful network functions. In *NSDI*, 2023.

[33] V. D. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre. Fast unfolding of communities in large networks, 2008.

[34] P. Bodík, I. Menache, M. Chowdhury, P. Mani, D. A. Maltz, and I. Stoica. Surviving failures in bandwidth-constrained datacenters, 2012.

[35] A. Broder. On the resemblance and containment of documents. In *Proceedings of the Compression and Complexity of Sequences 1997*. IEEE Computer Society, 1997.

[36] P. Carbone, A. Katsifodimos, S. Ewen, V. Markl, S. Haridi, et al. Apache Flink: Stream and batch processing in a single engine. In *ICDE*, 2015.

[37] M. Casado, M. J. Freedman, J. Pettit, J. Luo, N. McKeown, and S. Shenker. Ethane: Taking Control of the Enterprise. *ACM SIGCOMM Computer Communication Review*, 37(4):1, Oct. 2007.

[38] N. M. M. K. Chowdhury and R. Boutaba. Network Virtualization : State of the Art and Research Challenges. *IEEE ComSoc*, 2009.

[39] Cisco. Cisco Tetration. https://www.cisco.com/c/en_sg/products/data-center-analytics/tetration-analytics/index.html.

[40] C. Cranor, Y. Gao, T. Johnson, V. Shkapenyuk, and O. Spatscheck. Gigascope: High performance network monitoring with an sql interface. In *SIGMOD*, 2002.

[41] C. Cranor, T. Johnson, O. Spataschek, and V. Shkapenyuk. Gigascope: A stream database for network applications. In *SIGMOD*, 2003.

[42] C. Cranor, T. Johnson, O. Spatscheck, and V. Shkapenyuk. The gigascope stream database. *IEEE Data Eng. Bull.*, 2003.

[43] M. Dalton et al. Andromeda: Performance, Isolation, and Velocity at Scale in Cloud Network Virtualization. In *NSDI*, 2018.

[44] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2019.

[45] O. Ertl. Superminhash – a new minwise hashing algorithm for jaccard similarity estimation. https://arxiv.org/pdf/1706.05698.pdf.

[46] C. Estan, S. Savage, and G. Varghese. Automatically Inferring Patterns of Resource Consumption in Network Traffic. In *SIGCOMM*, 2003.

[47] B. Evans. Walmart cio: We picked microsoft for huge cloud deal to accelerate digital transformation. https://shorturl.at/aABI3.

[48] A. Gember-Jacobson, R. Viswanathan, C. Prakash, R. Grandl, J. Khalid, S. Das, and A. Akella. Opennf: Enabling innovation in network function control. In *SIGCOMM*, 2014.

[49] 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 *SIGCOMM*, 2015.

[50] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition, 2015.

[51] K. Henderson, B. Gallagher, T. Eliassi-Rad, H. Tong, S. Basu, L. Akoglu, D. Koutra, C. Faloutsos, and L. Li. Rolx: Structural role extraction & mining in large graphs. In *KDD*, 2012.

[52] M. Iliofotou, P. Pappu, M. Faloutsos, M. Mitzenmacher, S. Singh, and G. Varghese. Network monitoring using traffic dispersion graphs (tdgs). In *IMC*, 2007.

[53] Illumio. Zero Trust: the security paradigm for the modern organization. https://www.illumio.com/solutions/zero-trust.

[54] G. Jeh and J. Widom. Simrank: A measure of structural-context similarity. In *KDD*, 2002.

[55] S. Kandula, R. Chandra, and D. Katabi. What's Going On? Learning Communication Rules in Edge Networks. In *SIGCOMM*, 2008.

[56] S. Kandula, S. Sengupta, A. Greenberg, P. Patel, and R. Chaiken. The Nature of Datacenter Traffic: Measurements & Analysis. In *IMC*, 2009.

[57] T. Koponen, K. Amidon, P. Balland, M. Casado, A. Chanda, B. Fulton, I. Ganichev, J. Gross, P. Ingram, E. Jackson, A. Lambeth, R. Lenglet, S.-H. Li, A. Padmanabhan, J. Pettit, B. Pfaff, R. Ramanathan, S. Shenker, A. Shieh, J. Stribling, P. Thakkar, D. Wendlandt, A. Yip, and R. Zhang. Network virtualization in multi-tenant datacenters. In *NSDI*, 2014.

[58] A. Lakhina, M. Crovella, and C. Diot. Mining anomalies using traffic feature distributions. *SIGCOMM CCR*, 2005.
[59] Y. Li, R. Miao, C. Kim, and M. Yu. Flowradar: A better netflow for data centers. In *NSDI*, 2016.
[60] D. Lizorkin, P. Velikhov, M. Grinev, and D. Turdakov. Accuracy estimate and optimization techniques for simrank computation. In *VLDB Endowment*, 2008.
[61] Y. Mirsky, T. Doitshman, Y. Elovici, and A. Shabtai. Kitsune: An ensemble of autoencoders for online network intrusion detection. In *NDSS*, 2018.
[62] J. C. Mogul, D. Goricanec, M. Pool, A. Shaikh, D. Turk, B. Koley, and X. Zhao. Experiences with modeling network topologies at multiple levels of abstraction. In *NSDI*, 2020.
[63] B. Perozzi, R. Al-Rfou, and S. Skiena. Deepwalk: Online learning of social representations. In *KDD*, 2014.
[64] B. Pfaff, J. Pettit, T. Koponen, E. Jackson, A. Zhou, J. Rajahalme, J. Gross, A. Wang, J. Stringer, P. Shelar, K. Amidon, and M. Casado. The design and implementation of open vSwitch. In *NSDI*, 2015.
[65] A. Rawashdeh and A. Ralescu. Similarity measure for social networks – a brief survey. volume 1353, 2015.
[66] A. Roy, H. Zeng, J. Bagga, G. Porter, and A. C. Snoeren. Inside the social network's (datacenter) network. In *SIGCOMM*, 2015.
[67] S. Singh, F. Baboescu, G. Varghese, and J. Wang. Packet Classification Using Multidimensional Cutting. In *ACM SIGCOMM 2003*.
[68] Verizon. 2023 data breach investigations report. https://www.verizon.com/business/resources/reports/dbir/.
[69] Verizon. Data breach investigations report: 2008 – 2022. https://www.verizon.com/business/resources/reports/2022/dbir/2022-data-breach-investigations-report-dbir.pdf.
[70] VMWare. VMware NSX. https://www.vmware.com/products/nsx.html.
[71] K. Zhao, P. Goyal, M. Alizadeh, and T. E. Anderson. Scalable tail latency estimation for data center networks. In *NSDI*, 2023.