Scalable Firewalls for Simplifying Cloud Tenant Network Abstractions

Emily Marx
University of California Berkeley

Yichi Zhang
University of California Berkeley

Tenzin Ukyab
University of California Berkeley

Abstract

Cloud tenants frequently use workloads that span multiple regions in the cloud and require complex network setup to connect all the regions. McClure et al. suggests cloud providers reduce this complexity by presenting tenants with a high-level virtual network API, rather than a set of low-level virtual network components. However, this requires cloud providers make all tenant endpoints publicly routable but default-off. Currently, cloud tenant endpoints are not publicly-routable, which reduces the risk of DDoS attacks. Thus, generally the only firewalls are at the endpoints. However, adding public addresses for all tenant hosts makes the cloud network vulnerable to an increased volume of resource-exhaustion attacks. To mitigate this risk, we propose adding a firewall system to each cloud point of presence (PoP), in addition to the per-endpoint firewalls. A naive solution adding a single firewall representing firewall rules for all the endpoints served by the PoP would not scale, so we propose a more sophisticated system. This new system adds firewall servers sharded by tenant to the PoP. The edge router directs traffic to the correct firewall server; the most commonly used endpoint firewall rules are also cached in the edge router. With a cloud network of size 10^{12} VMs, our design has a 0.0004% rate of leaking disallowed network traffic into the cloud network, with half the CPU usage per tenant of the status-quo VPN tunnels.

1 Introduction

Cloud tenants frequently use workloads spanning multiple physical locations in the cloud, such as on-premise data centers and multiple regions across multiple cloud providers. This requires tenants to configure virtual networks connecting these locations. An example of this can be seen in Figure 1. This is a complex and expensive task requiring significant manpower and expertise. McClure et al. [16] suggests solving this issue by having cloud providers abstract away the low-level building blocks of virtual networks and only present cloud tenants with a clean API to set up their networks. More details of this API and abstraction can be found in [16].

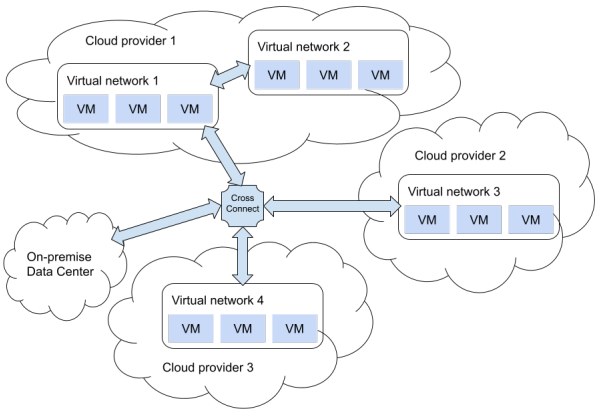

Figure 1: A tenant's virtual network spanning multiple cloud providers, regions, and an on-premise data center.

The implementation of this API requires cloud providers to make all tenant endpoints publicly-routable but default-off, meaning endpoints can be accessed by the entire Internet but traffic will be dropped unless it is explicitly permitted by tenant. This introduces security issues: The cloud network now has to support public addresses for all tenant hosts, making the network vulnerable to an increased volume of resource-exhaustion attacks.

Currently, this problem does not exist because endpoints are not publicly routable by default, as seen in Figure 2. Virtual machines (VMs) that are made public by the tenant have per-endpoint permit lists that the tenant provides at the end-host. These per-endpoint lists are implemented today by Security Groups (AWS), NSGs (Azure), etc. However, relying solely on firewalls at the endpoints leaves network resources vulnerable when

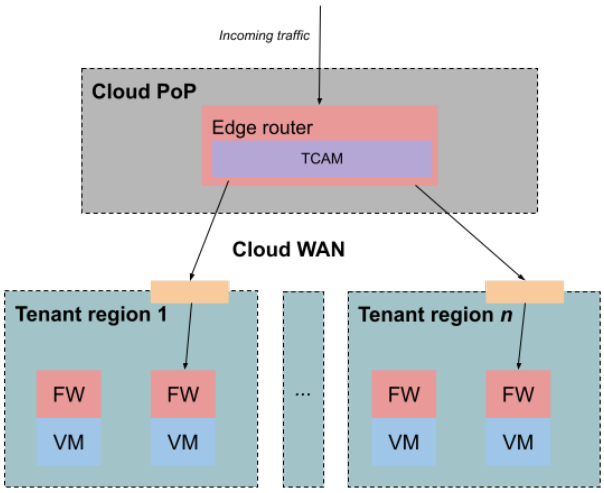

Figure 2: A single cloud provider currently has multiple points of presence (PoPs) that route public traffic into the cloud's private wide area network (WAN), toward regional data centers containing tenant virtual networks and virtual machines (VM) protected by per-endpoint firewalls (FW).

endpoints are publicly routable. Resource-exhaustion attack traffic can infiltrate the cloud network and use up expensive WAN bandwidth [13, 12, 11, 21].

Naively, software firewalls could be placed at every cloud entry point, such as edge data centers or points of presence, filtering traffic for all cloud tenants served by the entry point. Unfortunately, current firewall implementations such as Virtual Filtering Platform in Microsoft Azure [7] and Linux NetFilter would likely not be able to handle the large number of rules. Each entry point firewall would have to support $O(\text{number of endpoints} \times \text{number of external connections per end point})$, which may be up to 10^{13} in the worst case.

Instead of a single firewall at each entry point, we propose a more scalable firewall system. This design adds multiple software firewall servers to each entry point, sharded by tenant. Network traffic incoming to the entry point will first arrive at the edge router. The edge router will filter by destination address and direct traffic to the firewall server in the entry point corresponding to the destination tenant. These servers then apply the tenant's firewall rules and direct only permitted traffic over the WAN. We evaluate the performance of this design with respect to the number of rules per firewall server, finding constant CPU usage and latency of classification, and linear update latency. Additionally, we evaluate the feasibility of this design in comparison to the VPNs provided by clouds today. With a cloud network of size 10^{12} VMs, our design has a 0.0004% rate of leaking disallowed network traffic into the cloud network, with half the CPU usage per tenant of VPNs.

This paper will be organized as follows: Section 1 describes the motivation and background; Section 2 presents related works; Section 3 explains the threat model; Section 4 is a detailed description of the design; Section 5 provides the evaluation of the system; Section 6 discusses the evaluation, as well as limitations and future work; and Section 7 concludes.

2 Related work

2.1 Current Cloud Firewalls

This section will dive into the implementation of firewalls in clouds today, using as an example Microsoft's Azure cloud. Azure provides the Virtual Filtering Platform (VFP) firewall [7]. VFP implements many network tools, such as Ananta load balancing [17], ACLS, and VNets. A tenant can deploy a firewall using Azure's Firewall utility.

VFP organizes its programming model into ports, layers, groups, and rules. Each port contains layers, which are the stateful flow tables holding the rules. Rules are the filtering condition and actions. Groups are entities that manage and control rules within a layer.

VFP provides 4 classifier types to use for filtering: a compressed trie, an interval tree, a hash table, and a list [7]. The classifier used depends on the type of field in the packet the classifier is filtering on. For example, tries are used for CIDR IPs. Each data structure is optimized for the set of field types it classifies. VFP runs classification on each field in parallel and does not store a rule twice, optimizing time and space.

VFP provides firewall capabilities that are sufficient as a last line of defense for our endpoints. However, without implementation of firewalls closer to the entry points of the clouds, the cloud is vulnerable to resource-exhaustion attacks if endpoints are made publicly routable. Using a single VFP instance as an initial line of defense in the entry point would not scale for all the tenant host addresses.

Espresso [27] is Google's edge routing infrastructure that enables fast updating of Internet peering BGP FIB rules and ACL rules at points of presence. Espresso currently implements firewalling via an online traffic analysis for DoS detection. As DoS attacks are detected, rules are inserted and removed from routers to stop the attack. This paper will provide an offline solution to mitigating DoS attacks that can be used to raise the barrier of attack alongside an online solution.

2.2 Packet Classification Problem

The goal of packet classification is to classify packets by applying a set of rules to the header fields of a packet.

The dimension d refers to the number of fields in the packet P that are relevant to the classifier. Each rule R specifies d regular expressions to check against each of the fields in packet headers. The ith regular expression of R, $R[i]$, is checked against the ith field in the packet header, $P[i]$. If there is an exact, prefix, or range match, then the action, e.g. drop packet, forward packet, etc., specified for that rule is applied to that packet. If multiple rules match the packet, then either the longest prefix match or the highest priority rule is applied to the packet.

Packet classification is an old problem but is still important today due to its necessity in various applications such as QoS, firewalls, and network traffic monitoring and analysis. Classifiers are growing in size because of the aforementioned applications and need to handle higher throughput due to increasing line rate brought by fiber optics. Therefore we need packet classification algorithms with high throughput, low memory size, and fast update time as the applications change state frequently.

2.3 Ternary CAMs

A hardware solution to firewalls, also used for routing, is Ternary Content-Addressable Memory (TCAM). CAMs are a type of memory that can do line rate matching on binary words. Routing and firewall rules are stored in a match action table, where the match word is mapped to an action, such as allow, deny, or forward to a specific port. Ternary CAMs allow searches using three type of bit matches: 0s, 1s, and *s (wild cards or "don't care"). A TCAM stores the match action table in a memory array with decreasing order of priority. Input fields are checked against all entries in the memory array in parallel, allowing multiple entries to match. The address of the highest priority rule is used to index into the action SRAM memory to get the action for that rule. TCAM is valuable for its line rate classification capability; however, it is very expensive and has a limited capacity since it uses a lot of power and dissipates a lot of heat. For example, the Juniper MX10000 router has 7 million TCAM entries and can use up to 5500 W of power [1].

2.4 Bloom Filters

A Bloom filter is a space-efficient data structure used to query set membership. Rather than storing the elements of the set themselves, a Bloom filter stores bits representing them. Each element of the set is hashed using k hash functions, each of which outputs a bit index to be set to 1. When querying, the query element is hashed with the same k functions, and if all the resulting 1 bits are also 1 in the Bloom filter, the element may be in the set, but is not guaranteed to be. However, if any of the 1 bits in the query are 0 in the Bloom filter, the query is guaranteed not to be in the set. The likelihood of false positives depends on the size of the set, the size of the Bloom filter, and the number of hash functions.

Lookup time in a Bloom filter is constant with respect to the number of elements in the set, and depends only on the number of hash functions. This property, along with their space efficiency, makes them well-suited to algorithms related to packet classification, such as longest prefix match (LPM). The goal of LPM routing is to match a destination IP on the longest prefix in the forwarding table. This can be implemented by grouping prefixes by length and programming a Bloom filter for each length. Then, all the Bloom filters are queried in parallel, and the longest match is chosen [5].

3 Threat Model

In current cloud setups, cloud tenants set up virtual networks with virtual machines and can choose to connect them by VPN tunneling or by making an endpoint explicitly public and filtering at the endpoint using something similar to VPF. When making all endpoints publicly routable, the number of public IPs can increase exponentially. This widens the attack surface area for resource exhaustion attacks against the cloud and applications. Denial of service (DoS) of attacks can target a specific address or sweep an entire address space.

This increase in attack surface area makes the cloud's private wide area network (WAN) vulnerable to exhaustion. Network traffic enters the clouds at edge data centers or points of presence close to the origin of the traffic. Then it is shuttled to large regional data centers through the cloud's private WAN, which is expensive for the cloud to provision. Much work has gone into optimizing the utilization of the WAN for least cost [13, 12, 11, 21]. DoS attacks can congest this expensive WAN resource.

4 Design

4.1 Overview

Before detailing the design, we summarize the problem. A packet from the Internet arrives at a cloud entry point, such as an edge location or a point of presence (PoP), and its destination and source IP must be matched against the corresponding tenant endpoint's firewall rules. In clouds today, firewalling only occurs once the traffic reaches the endpoint. This is sufficient for traditional cloud setups since endpoints are not publicly routable, so the amount of disallowed traffic is minimal. However, making endpoints publicly routable increases the potential for DoS attacks to exhaust the cloud WAN bandwidth, since attackers are now able to direct traffic to the endpoints'

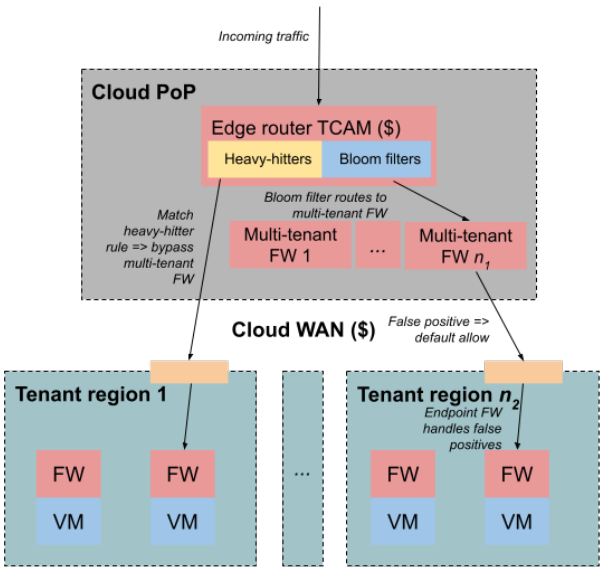

Figure 3: System design. At the edge router, the TCAM is partitioned into heavy hitters and Bloom filters pointing to multi-tenant firewalls.

public IPs. Thus, it is no longer acceptable to allow all traffic to exit the entry point before hitting a firewall: some firewalling must occur within the PoP.

The main goal of the design is to minimize the amount of disallowed traffic exiting the cloud entry point while maximizing the speed and throughput of allowed traffic. A naive solution would be to place a monolithic firewall at the PoP containing the rules of every tenant endpoint. This is infeasible because of the number of firewall rules: $O(\text{number of endpoints} \times \text{number of external connections per end point})$ is far too large to look up sufficiently quickly in a single firewall.

With these constraints in mind, the design introduces a new level of firewalling at the cloud entry point, in addition to the per-endpoint firewalls already present in traditional tenant VPCs. The overall system design can be found in Figure 3. Note that the firewalls in the PoP are the only new hardware added by the design; the TCAM and the WAN links already exist in clouds today. These new firewalls are sharded by tenant, limiting the number of rules per firewall to allow reasonable lookup time in software, using the NetFilter Linux kernel module. Then, the challenge becomes identifying which multi-tenant firewall a packet should be directed to. This must happen quickly; in particular, at line rate, so that incoming network traffic is not stalled at the router. The TCAM in the entry point's router is capable of line-rate lookups. However, edge routers today already use much of their TCAM for forwarding tables, leaving little room for additional entries related to firewalling. Thus, the entries pointing to the multi-tenant firewalls are compressed into Bloom filters, as described in section 4.2.

The multi-tenant firewalls limit the amount of traffic that would traverse the expensive cloud WAN only to be dropped at the endpoint firewall. However, they do not reduce this traffic to zero, since Bloom filters can produce false positives. In the false-positive case, a packet is directed to the wrong multi-tenant firewall. The firewall server must detect this and direct the packet over the WAN to the tenant region, since its firewall does not contain the tenant's rules and therefore the server cannot determine if the packet should be allowed or denied.

Directing traffic to the multi-tenant firewalls increases the latency of its path. Within an edge data center, latency between endpoints is on average 10 ms, so we expect about 20 ms increase. To minimize this, a small percentage of the TCAM caches a number of endpoint firewall rules. These cached rules will be the most frequently hit firewall rules or the "heavy hitter" rules. Traffic matching these rules bypasses the multi-tenant firewalls, as detailed in section 4.2.

In this design, an ingress packet traverses the cloud in the following way. First, it is matched against both partitions of the TCAM. If its source and destination IP matches one of the heavy-hitter rules, it can be directed over the WAN to the tenant region. Otherwise, if its destination matches one of the Bloom filters, it is routed to a multi-tenant server and directed over the WAN if it matches a rule in the server's firewall (or if the Bloom filter was a false positive). If the packet's destination matches no Bloom filter, it is not destined for any of the cloud's tenant endpoints and is dropped.

4.2 Router

This section details the design of the TCAM at the cloud PoP edge router, shown in Figure 4. Today, the TCAM contains a traditional forwarding table directing traffic entering and leaving the cloud to the correct interface [27]. Our design adds two additional partitions to the TCAM. The first contains a limited subset of the cloud tenants' firewall rules. This subset must be chosen carefully in order to use the expensive memory for optimal benefit. In particular, the aim is to maximize the amount of traffic matched by these rules, in other words, to choose the "heaviest-hitting" rules to cache in the TCAM.

Fortunately, clouds today perform extensive traffic analysis, and therefore have a thorough understanding of how much incoming traffic matches each firewall rule. Additionally, a number of empirical studies have found that traffic in large networks tends to exhibit a skewed distribution across destination and source IPs [10, 26, 4]. As shown in Figure 6, approximately 90% of packets are

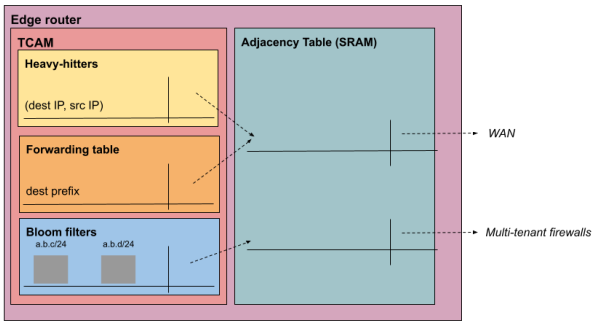

Figure 4: TCAM at the edge router is partitioned into three regions: the forwarding table; heavy-hitter rules, which route traffic directly over the WAN; and Bloom filters routing to multi-tenant firewalls. The elements of the Bloom filters are fixed-size blocks of a tenant's IP space.

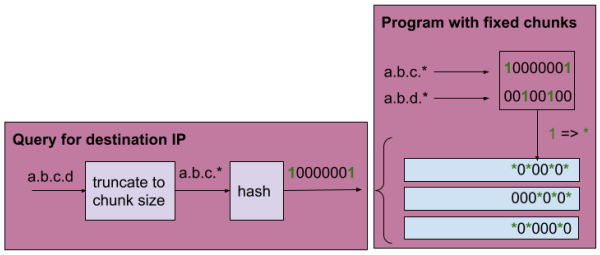

Figure 5: Chunks of tenant IP space are hashed into Bloom filters with 1 bits mapped to wildcards.

matched by only the 25 heaviest-hitting rules (note that the "permit rules" line is relevant in this context, since the endpoints are default-off. Also, the y-axis is logarithmic). Consequently, a large percentage of traffic can match the cached TCAM rules using a small amount of TCAM.

The second novel TCAM partition contains the Bloom filters that direct traffic to the PoP's multi-tenant firewalls. Each TCAM entry in this partition is a Bloom filter summarizing the IP space of multiple tenants. The space efficiency of Bloom filters allows the large number of tenants to be represented in the limited TCAM space. The design of these Bloom filters is inspired by previous work on Bloom filters for subset and superset queries [8]. Each element hashed into the Bloom filters is a fixed-size chunk of a tenant's IP space. Unlike a traditional Bloom filter, when an element hashes to a bit index, the bit is set to * rather than 1. In this way, a query destination IP matches the TCAM entry if its prefix matches one of the prefixes represented by the Bloom filter.

As a (purely illustrative) example, consider Figure 5. Here, one 8-bit TCAM line represents the prefixes $a.b.c/24$ and $a.b.d/24$, while the hash functions return bits 0 and 7 for the former and bits 2 and 5 for the latter. In this case, the TCAM line is $*0*00*0*$. The steps to querying for destination IP $a.b.c.x$ are as follows. First, the IP is truncated to the prefix length represented by each element of the Bloom filters. Then, the truncated IP is hashed by the k hash functions of the Bloom filter. Note that this will require hashing functionality in the router, possible for instance with P4 modules [14].

The query is then checked against all the TCAM entries in parallel as usual. In this example, the query would hash to 10000001, which matches the entry $*0*00*0*$ since all the indexes set to 1 in the query match the corresponding * indices in the Bloom filter. All 0 indices in the query either match 0s or *s, depending on whether other elements of the set hashed to those indices. As usual, the TCAM priority encoder chooses a single matching line out of potentially multiple matches. Thus, the probability of directing a packet to the wrong multi-tenant firewall is the false positive probability of a *single* Bloom filter. False-positive traffic is directed over the WAN and eventually reaches the per-endpoint firewalls, which act as a last line of defense in denying any false-positive traffic that would have been denied by the correct multi-tenant firewall in the PoP.

Note it is important that the chunks represented by each element of the Bloom filter have a fixed size, since each destination IP must be truncated to the same prefix length as every element. Thus, cloud providers will need to break the IP space of each tenant, which may be variable in total size, into chunks. The optimal chunk size will depend on the number of tenants, distribution of IP space size across tenants, and the amount of TCAM allocated to Bloom filters. Choosing a smaller chunk size increases the false positive probability, since the Bloom filter contains more elements, but reduces the over-allocation of IP space to each tenant that may occur with less granular chunks.

In addition to choosing the Bloom filter chunk size, cloud providers face several other tradeoffs in choosing the parameters of their firewall system in this design. One is the relative allocation of TCAM to heavy-hitter rules versus Bloom filters. Heavy-hitter rules reduce the average latency of a packet's path by allowing matched packets to bypass the multi-tenant firewalls. However, placing more heavy-hitter rules in the TCAM utilizes additional cloud WAN bandwidth by leaving less room for Bloom filters and therefore increasing the false positive rate, forcing the PoP firewall servers to default-allow more traffic. Thus, tuning the TCAM partitioning allows cloud providers to explore a tradeoff between latency and bandwidth. The optimal choice may be impacted by the relative importance of latency and bandwidth to the cloud's tenants, as well as the traffic distribution and

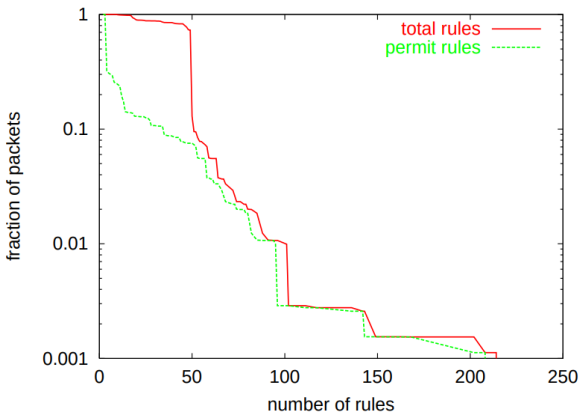

Figure 6: Traffic distribution across firewall rules (from [26]). The y-axis is the fraction of packets not matched by the corresponding number of rules.

total tenant IP space.

4.3 Firewall Manager

In order to manage and update rules for each virtual machine, a firewall manager service will need to be implemented by the cloud providers. Most cloud providers already implement similar services, such as AWS Security Groups. These services can be extended to manage the router and firewall servers at each point of presence.

In order to ease updates to the firewall servers, the design aggregates all rules for a particular tenant onto one server. This design aligns well with the contiguous address blocks assigned to each tenant. Updating the NetFilter firewall servers is linear with the number of rules in the table, as found in Section 5.2. Thus, the firewall manager can batch updates to the servers in time increments in order to handle a high frequency of updates from tenants.

Deleting from Bloom filters is not possible. Every time a Bloom filter has to be updated, it needs to be recreated from scratch. Therefore, instead of updating the Bloom filter every time the address space is reallocated, the firewall manager can batch updates to the filters once a threshold of delayed updates is reached. The design is able to handle incorrectly directed traffic due to its multi-level firewalling and ability to handle Bloom filter false positives, so the incorrectly directed traffic due to stale Bloom filter entries will not be an issue.

5 Evaluation

In evaluating this new firewalling system, we compare its drawbacks against its benefit. The drawbacks fall into two categories: the edge router TCAM and the multi-tenant firewalls. We evaluate these in sections 5.1 and 5.2, respectively. Some of the drawbacks regarding the router TCAM involve tradeoffs which cloud providers may explore by tuning parameters. That is, cloud providers may prefer to sacrifice more of one metric for another, based on their environment and priorities. To inform such decisions, we also explore these tradeoffs in section 5.1.

In quantifying the benefit of the system, we revisit the motivation for the publicly-routable endpoints requiring us to design new firewalling. To review, configuring virtual networks today is a complex and error-prone process. Thus, to evaluate the benefit of rethinking this abstraction, we must quantify the complexity saved. One metric of complexity is the reduction in number of network boxes that cloud tenants must manage in the redesigned virtual cloud. These boxes fall into several categories; the first is VPCs and their associated gateways. The number of VPCs and gateways per tenant is a multiplicative factor of several network components. Each of the tenant's cloud regions generally has a small number of domains, such as one for development, one for production, and one at the edge. In turn, each domain usually has a small number of VPCs. Thus, the total number of VPCs eliminated per tenant is $O(\text{cloud regions} \times \text{domains per region} \times \text{VPCs per domain})$. Removing these VPCs from a tenant's view also eliminates the associated gateways, of which each VPC has one or a few. The second type of virtual network device this system abstracts away is transit peerings between tenants' cloud regions. These peerings are often pairwise and unidirectional, making the number $O(\text{cloud regions}^2)$. To make these expressions more concrete, a recent survey reports that over 88% of surveyed enterprises use two or more cloud providers, with an average of 16 providers [20]. A tenant may use multiple regions within each provider. As an example, consider Aviatrix, a tenant with two cloud regions. For Aviatrix, the proposed virtual networking design reduces the number of network boxes configured by 12 VPCs, 16 gateways, and 2 peerings [15].

5.1 Router

To evaluate the performance of the TCAM design, the two main metrics of interest are the increase in WAN utilization due to Bloom filter false positives (Figure 7) and the amount of traffic that can bypass the multi-tenant firewalls (Figure 8). The former metric is important because WAN bandwidth is expensive, and the latter because the path through the multi-tenant firewalls has increased latency, as discussed in section 5.2.

In generating these figures, we assume a TCAM of

Table 1: Cloud statistics used to evaluate the scaling of our design. * This statistic was calculated based off the others.

Statistic	Value
Number of tenants	100k - 1,000k
VPCs per tenant	10s - 100s
Number of VMs per VPC/tenant	10s
Width of TCAM line	288
TCAM size	7,000,000
Forwarding table size	900,000
Total number of VMs*	10e7 - 10e13

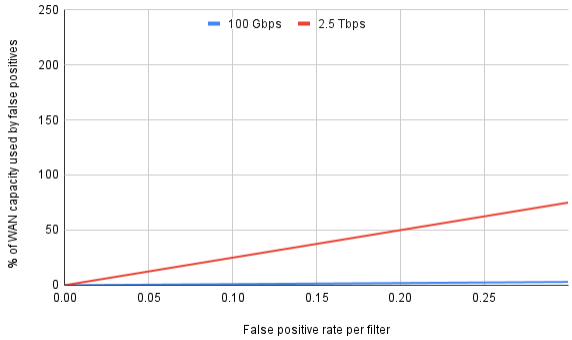

Figure 7: WAN utilization increase vs. Bloom filter false positive rate

width 288 bits, with 7 million entries, 900,000 of which are occupied by the forwarding table. The fixed-size chunks used to divide tenant IP space are of size 16,000 IPs, or a /18. The following equation is used for the Bloom filter false positive rate:

$$P = (1 - [1 - \frac{1}{m}]^{kn})^k$$

where P is the false positive rate, m is the number of bits in the Bloom filter (here, the width of the TCAM), k is the number of hash functions, calculated as $\frac{m}{n} \ln 2$, and n is the number of elements hashed into the filter (here, the number of fixed-size tenant IP chunks per TCAM line). We estimate the scale of the number of rules using ranges of statistics from cloud providers. These statistics can be found in Table 1. We calculated the maximum total number of VMs by multiplying the maximum number of VPCs per tenant, number of tenants, and number of VMs per VPC; the minimum was calculated analogously.

As shown in Figure 7, WAN utilization due to false positives increases linearly with the false positive rate of the Bloom filters that direct traffic to software firewalls in the PoP. This metric is calculated as a percentage of

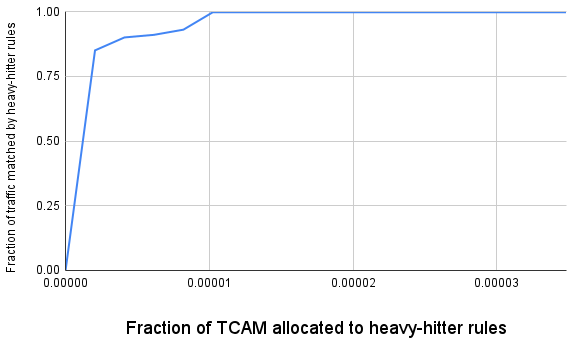

Figure 8: Traffic bypassing PoP FWs vs. TCAM heavy hitter fraction

WAN capacity. The baseline for this metric is zero: In clouds today no WAN is utilized due to Bloom filter false positives, since clouds today have no software firewalls in the PoP.

The figure shows two scenarios, corresponding to different levels of severity of DDoS. The steeper line shows an attack of 2.5 Tbps, and the other line 100 Gbps. For context, the worst DDoS attack on record was 2.5 Tbps, and most do not exceed 10 Gbps [2].

As shown in the figure, even at a false positive rate of 0.3 (30%), only 3% of WAN capacity is consumed by false-positive traffic in the 100 Gbps scenario (which is still an order of magnitude worse than most DDoS attacks), and 75% in the 2.5 Tbps scenario. For context, WAN utilization today is generally only 40-60% [11], so WAN links are likely to handle a small percentage increase well under average conditions.

We discuss the factors affecting false positive rate below (in Figures 9 and 10), and show that the rate is likely to be on the lower end of Figure 7.

Figure 8 shows the fraction of traffic that can bypass the multi-tenant firewalls, taking the fast path. This traffic is that which matches one of the TCAM heavy-hitter rules. Clouds today have no multi-tenant firewalls in the PoP, as that is a new component proposed by this design, so no traffic takes a path through firewalls in the PoP. Thus, the baseline for this figure is 100% of traffic taking the fast path. (However, as discussed, this is unacceptable with publicly-routable endpoints, due to the vulnerability of WAN to resource exhaustion attacks.) As shown in the figure, nearly all traffic is matched by the heavy-hitter rules with only a very small fraction of TCAM allocated to the rules. This is encouraging, since TCAM is a precious resource. In generating this figure, we used the empirical traffic distribution in the graph shown in Figure 6, and kept the number of endpoints constant at the average of the range discussed in 5.1.

The small amount of TCAM needed to put most traffic

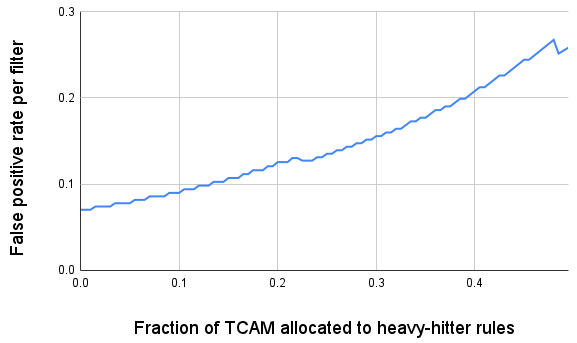

Figure 9: Bloom filter false positive rate vs heavy hitter fraction

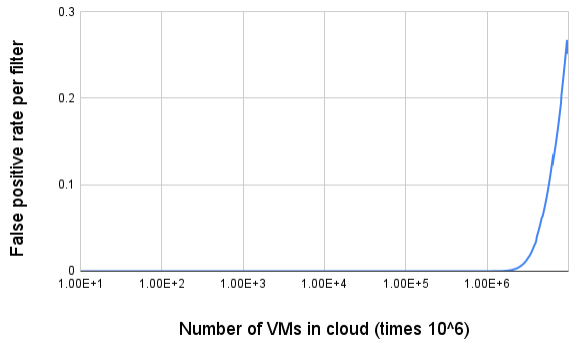

Figure 10: Bloom filter false positive rate vs. number of VMs

on the fast path has positive implications for the Bloom filter false positive rate, as shown in Figure 9. Decreasing the fraction of TCAM allocated to heavy-hitter rules decreases Bloom filter false positive rate, since more TCAM can be allocated to the Bloom filters. This presents an interesting tradeoff to cloud providers: allocating more TCAM to heavy-hitters reduces traffic latency by putting more packets on the fast path, but increases WAN bandwidth by increasing Bloom filter false positives.

A final factor affecting the Bloom filter false positive rate is the overall size of a cloud region, as measured by the number of tenant endpoints, shown in Figure 10. As the number of endpoints increases, the Bloom filters are more likely to have false positives, since more chunks are hashed into the TCAM space. The number of endpoints in a cloud depends on the provider; the range here was chosen based on conversations with a large cloud provider. As shown, the false positive rate is near zero until the very top end of the range.

5.2 Multi-Tenant Firewall

To evaluate the multi-tenant firewall servers, we measure performance as the number of rules per server increases and compare our results with the status-quo VPN server. Currently a tenant would connect two cloud deployments using two VPN servers at each end, and establish a tunnel in between. All traffic between the two clouds would go through the VPN servers; therefore, they serve a similar purpose as the multi-tenant firewall servers. Cloud users currently bear the cost of the VPN servers explicitly, since they need to maintain the servers on their own. Changing to the new design means that we are moving the computation currently done at the VPN servers of each tenant to the firewall servers at the edge locations. This increases the cost of compute managed by the cloud provider; however, the provider is likely to pass the cost back to the tenants, so the economic impact is likely to be unchanged.

We conducted several micro-benchmarks to measure the CPU usage, latency, and bandwidth of traffic against the number of rules installed in the multi-tenant firewalls. These firewalls are implemented using NetFilter. It is worth noting that NetFilter represents rules as "chains", which are linked lists where packet matches are specified. NetFilter also has a built-in notation of set, which is a hashed set of elements - in our case, IP addresses - and rules can be specified to match upon a certain set. Therefore, instead of maintaining the whitelist of addresses on the chain, we do it using a set and have only one rule on the chain that matches on the set. It is unacceptably slow if we maintain the addresses using multiple rules instead of one set. In the following paragraphs, we use "number of rules" and "number of elements" interchangeably.

We set up the benchmarks on two instances of the t3.large machines on AWS EC2 in the same datacenter, each having 2 virtual CPU cores, 8 Gigabytes of RAM, and a 5 Gigabit connection bandwidth. They are very underpowered compared to the machines we expect to see in actual deployments, but we also load them with much fewer rules and lighter traffic than we expect for the actual machines. For the baseline VPN measurements, we establish an IPSec tunnel between the two machines, and send traffic through the tunnel. For our measurements, we designate one machine as the server and put addresses into the whitelist mentioned before, and we use the other machine to send TCP traffic. We run the `tcp_bw` and `tcp_lat` test in `qperf` for 60 seconds, and gather CPU utilization using `mpstat`. We report the CPU utilization using the data during the bandwidth test, because that gives off a heavier load to the machines and stresses the CPU more than the latency test does.

As shown in Figure 11, as the number of rules installed increases, the CPU utilization on the firewall server holds

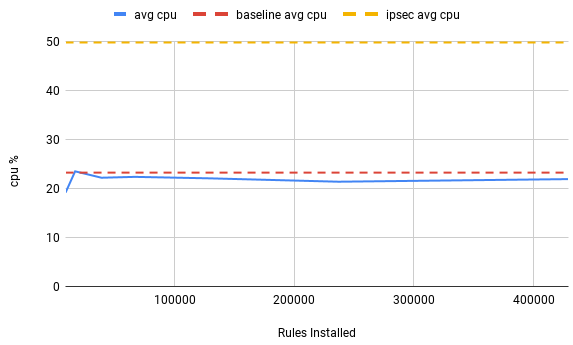

Figure 11: CPU usage during qperf bandwidth test

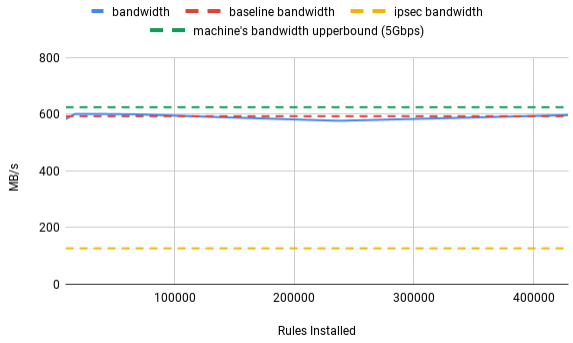

Figure 12: Bandwidth comparison

constant. The aggregate CPU usage of the status-quo VPN servers is the yellow line, or 50% of a single core, multiplied by the number of tenants. On the other hand, the aggregate usage for our design is the blue line, approximately 25%, multiplied by the number of firewall servers. The number of servers depends on the degree of sharding we land on, but it will be smaller than the number of tenants. Thus, the overall CPU usage of the design is lower than clouds today.

Figure 13 shows that our approach takes a roughly constant time to match a packet to a rule, because the elements are gathered in a hash set. This latency is slightly lower than the VPN tunnel, likely because it eliminates encryption and decryption. As shown in Figure 12, our approach gets very close to the theoretical bandwidth upper bound of 5Gbps, whereas sending traffic through the VPN tunnel yields a much lower bandwidth.

Storing elements in the hash set takes, unsurprisingly, linear space. With the RAM usage information shown in Figure 14, we can fit a linear model. In a machine with 1024 Gigabytes of RAM, we can put in about 8.7 million elements.

We also measured latency to update the ruleset. Unfortunately, as the number of rules installed increase, the

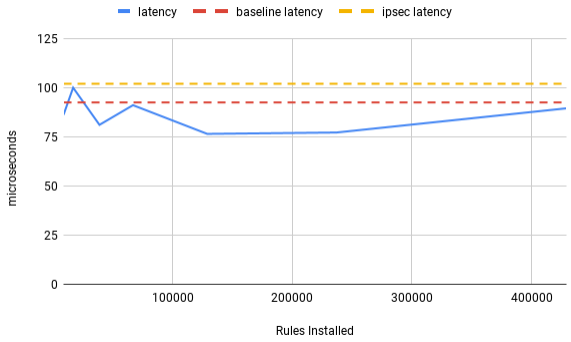

Figure 13: Latency comparison

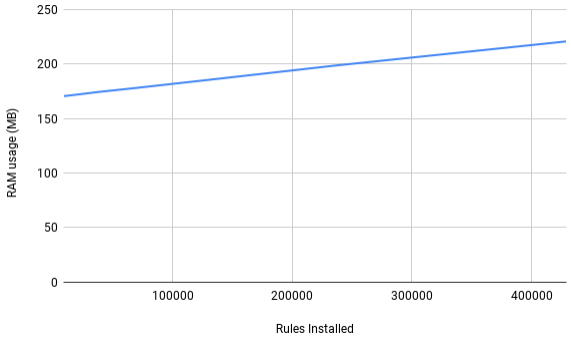

Figure 14: Memory usage

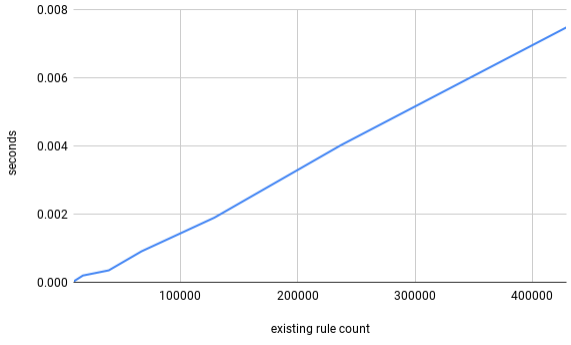

Figure 15: Time to add one element

time to install one more rule scales linearly as shown in Figure 15. Because there is a non-trivial overhead in both the addition of the rules and the round trip time between the firewall manager and the firewall server, batch update of the rules will certainly be beneficial. We suspect the overhead of adding rules comes from rehashing of existing elements, since we observe a high CPU utilization during the time.

6 Discussion

6.1 Limitations

The Bloom filter design requires hashing an incoming packet's destination IP. This hashing must be done in hardware in order to maintain line rate classification at the edge router. Current routers do not support hashing out of the box. However, routers capable of Programming Protocol-independent Packet Processors (P4) can be programmed with hashing hardware modules [14].

In designing the Bloom filters, we have assumed that a tenant's IP space is contiguous enough to be broken up into chunks of size /16. We consider this a reasonable assumption, since tenant IP space is generally contiguous within each VPC belonging to a tenant, and often across VPCs. However, if a cloud provider chooses to allocate IP space less contiguously, they may need to choose a smaller chunk size, which increases Bloom filter false positive rate.

6.2 Future work

Currently the multi-tenant firewall server design uses the software firewall implementation, NetFilter, that can be found by default in the Linux kernel. Instead of using the default Linux firewall implementation, other packet classification algorithms and data structures can be implemented in the kernel using eBPF [24], which allows running sandboxed programs in the kernel. Potential packet classification data structures include tries [19, 23], trees [3, 6], spatial decomposition methods [9, 18, 25], cross-producting [23] and tuple space search [22]. These may give us faster classification latency, faster update times, and improved memory utilization.

Bloom filters are not capable of deleting elements, since it is not possible to ascertain which bits were set by the element to be deleted versus other elements in the set. This is likely not a significant problem for most cloud providers, since tenant IP space is not likely to be frequently reallocated. However, for cloud providers who often find themselves removing tenants or reducing tenants' IP space, more complex TCAM data structures could be considered. There has been previous work on deletion-capable Bloom filters, such as counting Bloom filters [8, 5].

As discussed in sections 4.2 and 5.1, cloud providers control a number of parameters that impact the performance of the firewall system in various ways. While cloud providers often have extensive systems for monitoring the metrics involved in these tradeoffs, they may benefit from new systems to better understand how these metrics impact the firewall's performance. This is particularly true since the relationships between many of the parameters are intertwined. Future work may include building automated models of these relationships. This may also be helpful when facing decisions about whether to acquire additional amounts of very expensive resources used by the firewall, such as TCAM and WAN bandwidth.

Many cloud providers are concerned about isolation between tenants. When combining tenants in both the Bloom filters and the multi-tenant firewalls, providers will likely want to choose carefully in order to prevent heavy traffic to one tenant, or changes in the traffic distribution to a tenant, from impacting the latency or bandwidth observed by another tenant. This will involve balancing load and number of rules across the multi-tenant firewall servers, as well as assigning tenants to Bloom filters such that each tenant sees a similar false positive rate. The latter point may be another reason to consider Bloom filters with deletion, as discussed above.

6.3 Extensions

This sharded firewall design can be used in other situations where there is a large scale of rules. ISPs implement some simple firewalling for customers. If they want to provide firewalls as a service they may need to support rules at the scale of cloud providers. In Internet of things settings, such as smart homes or offices, devices may need network security. For example, for obvious physical security reasons a fire alarm may need to be DoS protected. A firewall system such as our design may need to be implemented at the gateway router if there are many devices within a large building.

6.4 IPv6

Given the number of VMs that require pubicly-routable IPs in this design, we will likely need IPv6 to be deployed on a larger scale. This brings both potential improvements and possible drawbacks. On the negative side, the size of filters will increase by a factor of about 4, since a whole v6 address is 4 times the size of a v4 address. This reduces the number of rules that can fit in a fixed allocation of TCAM and PoP firewall servers. However, IPv6 provides the opportunity to have a less

fractured address space. With 128-bit addresses, we can afford to assign much larger blocks of addresses to each tenant, reducing the concern for overallocation in determining the number of elements hashed to the Bloom filters.

7 Conclusion

Making all addresses publicly routable but default off in a cloud increases the surface of attack for resource exhaustion, by making more of the cloud's addresses accessible. Currently, cloud providers place firewalls at the tenant endpoints but not at the cloud PoP, so introducing publicly routable endpoints allows attackers to congest the expensive private WAN network of the cloud. In order to protect the private WAN, we propose a design that places a scalable firewall system at every PoP or edge data center. In order for the firewall to scale to the large number of rules, we have sharded the firewall by tenant. The router at the PoP also needs to handle the scale of the rules, so we dedicated a part of the TCAM in the router for Bloom filters that match destination addresses to forwarding actions that route to the PoP firewalls. Bloom filters allow the entire public address space of the cloud to fit in the limited TCAM space, since multiple address spaces can be mapped to one line and multiple tenants can be aggregated onto one firewall server. Additionally, Bloom filters cannot support variable prefix length matching. So, the address space of the cloud provider is cut into blocks to enable fixed prefix length matching. We also dedicate a part of the TCAM for frequently hit rules. This enables a large portion of the valid incoming network traffic to be forwarded directly into the WAN toward its destination instead of having to go through the multi-tenant firewalls.

In our evaluation, we analyzed how the false positive nature of the Bloom filters in the TCAMs affect the WAN bandwidth. Since each line in the TCAM is evaluated in parallel, it is possible to match on multiple lines. The TCAM chooses one of the lines as the best match. The action for that line might not point to the correct multi-tenant firewall due to the false positive nature of the Bloom filters. So, if a multi-tenant firewall receives packets that do not match its tenants address spaces, then it will default-allow the traffic. Therefore with a larger number of VMs, our false positive rate increases, which directly relates to an increase in WAN utilization. However, our evaluation shows that by allocating only a small portion of the TCAM to heavy hitters, a large portion of the network traffic can bypass the multi-tenant firewalls. We also evaluated the feasibility of this design for a cloud provider to implement. We found that the firewalls in the PoP are comparable to or better than status-quo VPN tunnels in terms of CPU, bandwidth, and classification latency.

8 Acknowledgements

We would like to thank our advisor Professor Sylvia Ratnasamy and our collaborators Sarah McClure and Zeke Medley for their support. We would also like to acknowledge our course instructors Professor John Kubiatowicz, Professor Natacha Crooks, and teaching assistant Stephanie Wang.

References

[1] MX10000 Modular Universal Routing Platforms Datasheets | Juniper Networks.

[2] Famous ddos attacks — the largest ddos attacks of all time, 2021.

[3] BUDDHIKOT, M. M., SURI, S., AND WALDVOGEL, M. Space Decomposition Techniques for Fast Layer-4 Switching. In *Protocols for High-Speed Networks VI*, J. D. Touch and J. P. G. Sterbenz, Eds., vol. 31. Springer US, Boston, MA, 2000, pp. 25–41. Series Title: IFIP Advances in Information and Communication Technology.

[4] COHEN, E., AND LUND, C. Packet classification in large isps: Design and evaluation of decision tree classifiers. *ACM SIGMETRICS Performance Evaluation Review 33*, 1 (2005), 73–84.

[5] DHARMAPURIKAR, S., KRISHNAMURTHY, P., AND TAYLOR, D. E. Longest prefix matching using bloom filters. *IEEE/ACM Transactions on Networking 14*, 2 (2006), 397–409.

[6] FELDMAN, A., AND MUTHUKRISHNAN, S. Tradeoffs for packet classification. In *Proceedings IEEE INFOCOM 2000. Conference on Computer Communications. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies (Cat. No.00CH37064)* (Mar. 2000), vol. 3, pp. 1193–1202 vol.3. ISSN: 0743-166X.

[7] FIRESTONE, D. VFP: A Virtual Switch Platform for Host SDN in the Public Cloud. 14.

[8] GOEL, A., AND GUPTA, P. Small subset queries and bloom filters using ternary associative memories, with applications. *ACM SIGMETRICS Performance Evaluation Review 38*, 1 (2010), 143–154.

[9] GUPTA, P., AND MCKEOWN, N. Packet Classification using Hierarchical Intelligent Cuttings. 9.

[10] HAMED, H., AND AL-SHAER, E. Dynamic rule-ordering optimization for high-speed firewall filtering. In *Proceedings of the 2006 ACM Symposium on Information, computer and communications security* (2006), pp. 332–342.

[11] HONG, C.-Y. Achieving High Utilization with Software-Driven WAN. 15.

[12] HONG, C.-Y., MANDAL, S., AL-FARES, M., ZHU, M., ALIMI, R., B., K. N., BHAGAT, C., JAIN, S., KAIMAL, J., LIANG, S., MENDELEV, K., PADGETT, S., RABE, F., RAY, S., TEWARI, M., TIERNEY, M., ZAHN, M., ZOLLA, J., ONG, J., AND VAHDAT, A. B4 and after: managing hierarchy, partitioning, and asymmetry for availability and scale in google's software-defined WAN. In *Proceedings of the 2018 Conference of the ACM Special Interest Group on Data Communication* (New York, NY, USA, Aug. 2018), SIGCOMM '18, Association for Computing Machinery, pp. 74–87.

[13] JAIN, S., KUMAR, A., MANDAL, S., ONG, J., POUTIEVSKI, L., SINGH, A., VENKATA, S., WANDERER, J., ZHOU, J., ZHU, M., ZOLLA, J., HÖLZLE, U., STUART, S., AND VAHDAT, A. B4: Experience with a Globally-Deployed Software Defined WAN. 12.

[14] KFOURY, E. F., CRICHIGNO, J., AND BOU-HARB, E. An exhaustive survey on p4 programmable data plane switches: Taxonomy, applications, challenges, and future trends. *IEEE Access* (2021).

[15] MCCLURE, S. Rethinking networking abstractions for cloud tenants (the fun new cloud).

[16] MCCLURE, S., RATNASAMY, S., BANSAL, D., AND PADHYE, J. Rethinking networking abstractions for cloud tenants. In *Proceedings of the Workshop on Hot Topics in Operating Systems* (New York, NY, USA, June 2021), HotOS '21, Association for Computing MFachinery, pp. 41–48.

[17] PATEL, P., BANSAL, D., YUAN, L., MURTHY, A., GREENBERG, A., MALTZ, D. A., KERN, R., KUMAR, H., ZIKOS, M., WU, H., KIM, C., AND KARRI, N. Ananta: Cloud Scale Load Balancing. 12.

[18] QI, Y., XU, L., YANG, B., XUE, Y., AND LI, J. Packet Classification Algorithms: From Theory to Practice. In *IEEE INFOCOM 2009* (Apr. 2009), pp. 648–656. ISSN: 0743-166X.

[19] QIU, L., VARGHESE, G., AND SURI, S. Fast firewall implementations for software and hardware-based routers. In *Proceedings Ninth International Conference on Network Protocols. ICNP 2001* (Nov. 2001), pp. 241–250.

[20] RAMEL, B. D., AND 10/21/2019. Research Brief Summarizes Trends in Multi-Cloud Deployments -.

[21] SINGH, R., BJORNER, N., SHOHAM, S., YIN, Y., ARNOLD, J., AND GAUDETTE, J. Cost-effective capacity provisioning in wide area networks with Shoofly. In *Proceedings of the 2021 ACM SIGCOMM 2021 Conference* (New York, NY, USA, Aug. 2021), SIGCOMM '21, Association for Computing Machinery, pp. 534–546.

[22] SRINIVASAN, V., SURI, S., AND VARGHESET, G. Packet Classification using Tuple Space Search. 12.

[23] SRINIVASAN, V., VARGHESET, G., SURIS, S., AND WALDVOGELG, M. Fast and Scalable Layer Four Switching. 12.

[24] THE LINUX FOUNDATION. eBPF, 2021.

[25] VAMANAN, B., VOSKUILEN, G., AND VIJAYKUMAR, T. N. EffiCuts: optimizing packet classification for memory and throughput. 12.

[26] WALLERICH, J., DREGER, H., FELDMANN, A., KRISHNAMURTHY, B., AND WILLINGER, W. A methodology for studying persistency aspects of internet flows. *ACM SIGCOMM Computer Communication Review 35*, 2 (2005), 23–36.

[27] YAP, K., MOTIWALA, M., RAHE, J., PADGETT, S., HOLLIMAN, M., BALDUS, G., HINES, M., KIM, T., NARAYANAN, A., JAIN, A., LIN, V., RICE, C., ROGAN, B., SINGH, A., TANAKA, B., VERMA, M., SOOD, P., TARIQ, M., TIERNEY, M., TRUMIC, D., VALANCIUS, V., YING, C., KALLAHALLA, M., KOLEY, B., AND VAHDAT, A. Taking the edge off with espresso: Scale, reliability and programmability for global internet peering.