USENIX

THE ADVANCED COMPUTING SYSTEMS ASSOCIATION

Cost-effective and Reliable Global Internet Peering with Programmable Switches

Congcong Miao, *Tencent and National University of Singapore;* Zhiyi Yao, Jianchao Lv, and Jinglin Wang, *Tencent;* Shihan Lin, *University of Michigan;* Xinyi Zhang, *CNIC CAS, China;* Yunming Xiao, *The Chinese University of Hong Kong, Shenzhen;* Wei Guo, Jiwu Bu, Yachen Wang, and Xianneng Zou, *Tencent;* Yong Jiang, *Tsinghua Shenzhen International Graduate School;* Marco Canini, *KAUST;* Gaogang Xie, *CNIC CAS, China*

https://www.usenix.org/conference/nsdi26/presentation/miao-peering

This paper is included in the Proceedings of the 23rd USENIX Symposium on Networked Systems Design and Implementation.

May 4–6, 2026 • Renton, WA, USA

ISBN 978-1-939133-54-0

Open access to the Proceedings of the 23rd USENIX Symposium on Networked Systems Design and Implementation is sponsored by

جامعة الملك عبدالله للعلوم والتقنية

King Abdullah University of Science and Technology

Cost-effective and Reliable Global Internet Peering with Programmable Switches

Congcong Miao[1,2], Zhiyi Yao[1], Jianchao Lv[1], Jinglin Wang[1], Shihan Lin[3], Xinyi Zhang[4], Yunming Xiao[5], Wei Guo[1], Jiwu Bu[1], Yachen Wang[1], Xianneng Zou[1], Yong Jiang[6], Marco Canini[7], Gaogang Xie[4]
[1]*Tencent,* [2]*National University of Singapore,* [3]*University of Michigan,*
[4]*CNIC CAS, China,* [5]*The Chinese University of Hong Kong, Shenzhen,*
[6]*Tsinghua Shenzhen International Graduate School,* [7]*KAUST*

Abstract

Large-scale cloud providers always deploy peering routing system at the Internet's peering edge to route traffic between the cloud and the Internet. Traditional router-based peering systems fail to pace up to the fast-changing application requirements, while the recent host-based approach is not cost-effective and struggles to address malicious traffic from the Internet. In this paper, we advocate for a radical new peering architecture to introduce programmable switches at the peering edge. We propose and implement a first-of-its-kind system, called Janus, to simultaneously handle inbound and outbound network traffic and significantly reduce network hardware cost. The core of Janus's approach is to leverage a traffic dispatch module to offload most of the outbound traffic to switches to enhance system's scalability. Janus offloads all inbound traffic to switches and redirects potential malicious traffic to the anti-DDoS service to enhance system reliability. Furthermore, Janus introduces a fast route convergence mechanism to effectively handle Internet-scale route updates. We have gradually deployed Janus at the edge of our production network over the past years. Evaluation results show that Janus can reduce the average hardware cost by 78%, as compared to existing systems while gracefully handling DDoS issues. Meanwhile, Janus can reduce the route convergence within seconds under failure scenarios, which is orders of magnitude faster than existing approaches.

1 Introduction

The Internet's peering edge is vital in enabling the exchange of Terabits/s of Internet traffic among cloud providers and partner Internet service providers (ISPs). Large-scale cloud providers commonly deploy peering routing systems at point-of-presence (PoP) across the world where Internet users from ISPs can access cloud services through the nearby edge. The Border Gateway Protocol (BGP) [17] is a widely used routing mechanism in peering routing systems to establish inter-domain routes on the Internet. As an edge exchanging point, the peering routing system plays an important role. On the one hand, as there are many peers providing various quality of service (QoS) classes with different bandwidth costs, it should effectively route the *outbound* traffic (i.e., traffic from cloud applications to the Internet) to the desired peers. On the other hand, malicious traffic is widespread on the Internet. The peering routing system can mitigate malicious attacks, especially by blackholing Distributed Denial of Service (DDoS) attacks from the *inbound* traffic (i.e., traffic from the Internet to cloud applications).

Traditionally, peering routing systems have relied on router-based architectures [18], where commercial routers run BGP to establish inter-domain routes among partner ISPs. However, due to inherent limitations, including low feature velocity and high complexity for managing and configuring commercial routers, this approach does not effectively meet the requirements of fast-evolving cloud services. Recently, host-based peering routing [25] has emerged as a promising alternative that offers flexibility in rapid feature development and provides a more granular routing mechanism for network traffic. However, it still faces significant challenges in effectively handling the demands of cloud edge peering. On the one hand, peering on the Internet faces frequent attacks. Our production measurements during a three-month period reveal that our peering edge encounters over 4,200 DDoS attacks, averaging 47 attacks per day. Among these attacks, 15.5% caused processing capacity overflow, resulting in potential packet loss for cloud applications. On the other hand, the processing capacity of a single host is limited by the bandwidth of its network interface cards (NICs), typically 200Gbps, and the packet processing capabilities of its CPUs. When processing hundreds of Tbps of traffic from the Internet, the host-based system requires thousands of servers, which incurs significant capital and operational expenditures, in the order of millions of dollars (§ 2).

To address the aforementioned issues, we advocate for a radically new peering architecture built upon programmable switches at the peering edge, leveraging the powerful forwarding capability of switching ASICs. We propose and implement a first-of-its-kind system, called Janus, to simultaneously handle inbound and outbound network traffic and significantly

reduce the network hardware cost. Janus addresses several challenges: at the system level, it combines switches and hosts to achieve cost-effective routing while managing the switching ASICs' limited resources; at the protocol level, it designs efficient route synchronization among all peering devices and realizes a fast route convergence mechanism to accelerate the Internet-scale route update process (§ 3).

As nowadays there are about one million routes in the Internet, a key challenge we need to overcome is the limited number of entries in switching ASICs (i.e., 320K), which prevents us from storing the entire Internet-scale forwarding table. Our measurement of a typical PoP for two years shows that 1.7% of routes forward more than 80% of network traffic. This observation motivates us to design a hierarchical traffic dispatch architecture: the outbound traffic is processed by an internal peering switch (PS) and then dispatched to either host servers or an external peering switch based on the destination IP addresses. All switches and hosts run BGP to calculate entire Internet-scale routes. A small fraction of routes that are responsible for most of the traffic are offloaded to the internal PS ASIC, which directly forwards the corresponding traffic to the external PS. The remainder of the traffic, which is smaller in volume, is handled by hosts, greatly reducing network hardware costs. Furthermore, by leveraging host memory to store entire routes, we introduce a two-level classifier to achieve a more precise routing mechanism that enables application-aware routing (§ 4.2, § 4.3).

Another critical challenge is high reliability, particularly in addressing DDoS attacks from the Internet. We direct all inbound traffic from the external PS to the internal PS; this avoids overloading the hosts with malicious traffic. Traffic regarded as potentially malicious is redirected to an anti-DDoS service, which comprises a cluster of servers for further inspection. The anti-DDoS service drops the malicious traffic and forwards application traffic back to the internal PS. Furthermore, once certain traffic is identified as malicious, we migrate the corresponding ACL rules from the anti-DDoS service to the external PS for blackholing. In turn, this reduces the traffic handled by the anti-DDoS service (§ 4.4).

Under failure scenarios such as link failure causing millions of routes to be updated, the traditional approach will take minutes to converge at the Internet scale, causing packet loss during the convergence. To address this, we design a fast route convergence mechanism to accelerate the Internet-scale route withdrawal process. Specifically, we devise remote link sense where the firstly affected peering switch will send a notification message to the routing reflector and the reflector will broadcast the failure information to all the peering devices. Then, we use a carefully-constructed linked list to identify and withdraw the affected routes as a batch, which further accelerates the route update process (§ 4.5).

Finally, we set up a testbed to conduct a stress test. Using this testbed, we demonstrate the capacity of a switch to store and process Internet-scale routes within its CPU subsystem and we determine the forwarding capacity of a host at 120Gbps as a safe threshold. We then gradually deployed Janus at the edge of our production network over the past year. Evaluation results show that Janus can reduce the average hardware cost by 78%, as compared to existing systems while gracefully handling DDoS attacks. Janus can achieve route convergence within seconds under failure scenarios, which is orders of magnitude faster than existing approaches. Furthermore, our application-level routing in Janus enables application traffic to be routed to a specific peer while meeting its demand and reducing the bandwidth cost by up to 50% as compared to our previous approach (§ 5).

This work **does not** raise any ethical issues.

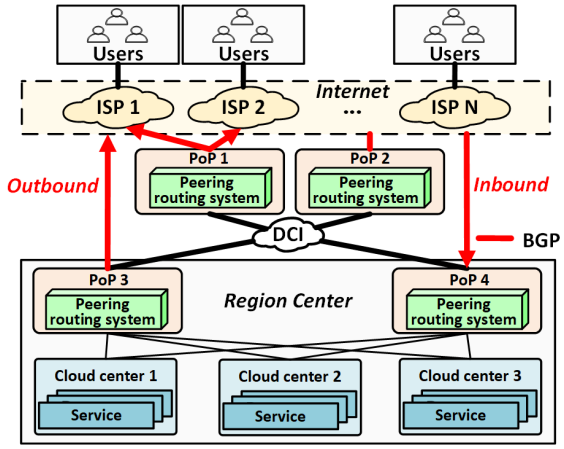


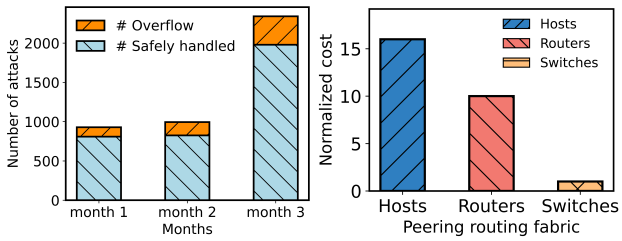
Figure 1: Cloud peering system.

2 Background and Motivation

2.1 Cloud edge peering

As illustrated in Figure 1, the Internet's peering edge is crucial for large-scale cloud providers, enabling the exchange of terabits per second of Internet traffic with partner ISPs and other cloud providers [11]. To support this, cloud providers deploy peering routing systems at Points of Presence (PoPs) distributed globally, allowing Internet users from ISPs to access diverse cloud applications in regional centers via the nearest edge. These regional centers are interconnected through data center interconnection (DCI) links. The Border Gateway Protocol (BGP) [17] is the standard routing protocol used in these systems to establish inter-domain routes across the Internet. As an edge exchange point, the peering routing system is responsible for directing outbound traffic (from cloud applications to the Internet) to appropriate peers, while also defending against malicious inbound traffic (from the Internet to cloud applications).

Commercial peering routers. Traditionally, cloud providers have relied on Internet-scale commercial peering routers (PRs) as the backbone of their peering routing systems to exchange data with other autonomous systems (ASes). PRs are capable of supporting millions of entries in their forwarding information bases (FIBs) and handling terabits per second



(a) The number of DDoS attacks. (b) Cost.

Figure 2: (a) The number of DDoS attacks our production network encounters within three months. (b) The normalized cost of peering devices providing 100Gbps bandwidth capacity.

of Internet traffic at scale.

While these routers provide essential functionalities such as BGP peering and Internet-scale FIBs, they lack the flexibility to accommodate customized routing for emerging demands. For instance, deploying a new feature like application-aware routing, i.e. customizing routing paths based on application-specific requirements, including QoS priority, latency sensitivity, in production typically requires a turnaround time of two weeks. However, upgrading routers to support such features often takes months or even years due to the time-intensive nature of hardware development. This significant delay makes it challenging for routers to keep pace with rapidly evolving product requirements. Additionally, given the substantial management complexity of Internet-scale routers, cloud providers tend to minimize their usage at peering edges, favoring a smaller number of large, high-capacity routers.

Host-based peering routing. Recent advancements have shifted Internet-scale routing to host servers, introducing host-based peering routing [25]. In this system, packet processing and forwarding are implemented using the Data Plane Development Kit (DPDK), while Internet-scale FIB routes are constructed by running the BGP stack on the hosts. This approach externalizes BGP from the peering fabric (PF) to software processes running on the hosts.

Leveraging the programmability of commercial CPUs, host-based peering routing enables efficient customization of packet forwarding based on services, users, or other policies. Additionally, it facilitates the integration of Software-Defined Networking (SDN) technologies, such as traffic engineering, into Internet peering. This architecture also enhances testability for management and configuration, supporting rapid deployment of new service features.

2.2 Limitations of existing peering systems

While the recently proposed host-based peering routing offers a more granular mechanism for network traffic, such as application-aware routing, our deployment experience reveals significant challenges that hinder its large-scale adoption. Below, we detail these critical issues.

Vulnerability to DDoS attacks. The Internet is rife with malicious traffic, making the protection against Distributed

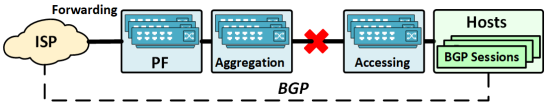


Figure 3: A typical example of host-based peering routing that leads to high maintenance complexity.

Denial of Service (DDoS) attacks a critical objective for peering systems. However, host-based peering routing systems are particularly vulnerable to such attacks, leading to potential packet loss for cloud application traffic.

To better understand this issue, we mirrored Internet traffic at the peering edge over a three-month period and analyzed the characteristics of DDoS attacks. Figure 2(a) illustrates the number of DDoS attacks encountered each month. Our observations reveal that the peering edge faced over 4,200 DDoS attacks during this period, averaging 47 attacks per day. A deeper analysis of the traffic patterns shows that 15.5% exceed the packet processing capacity of a single CPU core, leading to packet loss for the application flows that are assigned to the same core for forwarding.

Although these attacks are eventually detected and mitigated by the security team, the detection process often takes tens of seconds. During this time, the attack traffic is forwarded and processed by the hosts, consuming CPU resources. In severe cases, this can exhaust the CPU's capacity, leaving it unable to handle legitimate cloud application traffic, resulting in significant packet loss and degraded quality of service.

Poor scalability. With the emergence of applications such as high-definition video and AI-generated content (AIGC) services, the network traffic passing through edge peering systems has increased dramatically. However, the total network throughput a single host can handle is constrained by the bandwidth of its NICs and the processing capacity of its CPU cores. Our testbed evaluation shows that the forwarding capacity of a typical server is at most 200Gbps, and is capped at 120Gbps when considering operational margins (§ 5.1). To forward 10Tbps of Internet traffic, a peering routing system would require at least 50 hosts. Considering the deployment across tens of PoPs and the need for redundancy to ensure fault tolerance, host-based peering routing often necessitates thousands of servers, representing assets worth tens of millions of dollars. As illustrated in Figure 2(b), the cost of peering host servers to provide 100Gbps of bandwidth capacity can be significantly higher—ranging from 1.6× to 16×—compared to routers and switches used in our production environment. Consequently, host-based peering routing is costly and struggles to scale with the drastically-growing bandwidth demands of cloud applications.

Maintenance complexity. Host-based peering routing externalizes BGP from the peering fabric (PF) to software processes running on host servers, significantly increasing maintenance complexity. Figure 3 illustrates a typical example of this architecture. In this setup, Internet-scale FIB routes

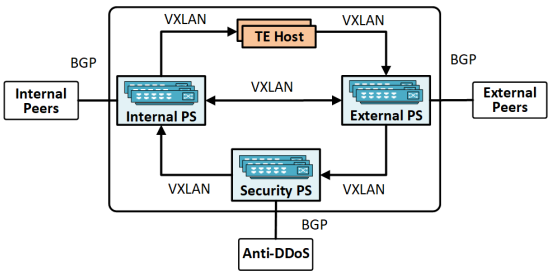


Figure 4: High-level architecture of Janus.

are constructed by running the BGP stack on the host, with BGP sessions established through intermediate switches (e.g., aggregation and access switches). These switches are responsible for forwarding packets to and from peers. However, the introduction of these forwarding switches complicates link failure detection. For instance, if a link failure occurs between the aggregation and access switches, the ISP router's link layer wouldn't detect the link failure, since its connection to the PF router is still active and operational. As a result, significant traffic loss can occur before the failure is identified and resolved.

3 Rethinking the Peering Architecture

We propose a fundamentally different peering architecture to address the limitations of existing approaches. First, we outline the design principles guiding our proposal (§ 3.1). Next, we introduce an architecture that leverages the capabilities of programmable switches (§ 3.2). Finally, we discuss the key challenges in realizing this proposal (§ 3.3).

3.1 Design principles

Unlike prior work [18, 25], our system design adheres to the following basic principles.

- **Reliability.** As a critical component in the architecture of large-scale cloud systems, the peering routing system must ensure high reliability. Given the prevalence of malicious traffic on the Internet, such as DDoS attacks, the system should effectively protect cloud application traffic from being disrupted by these threats.
- **Cost-effectiveness.** The network hardware cost per 100Gbps in the peering routing system should scale efficiently, remaining manageable even as service demands grow rapidly. Otherwise, the hardware cost of the peering system could escalate to tens of millions of dollars in the near future, making it unsustainable.
- **Fast convergence.** Network failures, such as link failures, are inevitable among Internet peers, causing all routes to the affected peer to become invalid. With millions of routes from a single peer, the peering routing system must converge as quickly as possible to minimize packet loss and ensure uninterrupted application traffic.
- **Flexibility.** The system should support customized routing strategies tailored to specific business traffic requirements, such as service type, locality, and priority. Additionally, these strategies must be capable of rapid iteration and deployment to adapt to evolving production needs.

3.2 Switch-based peering routing

We present Janus, a novel peering architecture that elegantly adheres to the aforementioned design principles.

Leveraging programmable switches at the Internet's peering edge. Figure 4 shows the high-level architecture of Janus. The key innovation that sets Janus apart from existing approaches is the introduction of programmable switches at the peering edge, functioning as peering switches to interconnect various peers. These switches leverage the powerful forwarding capabilities of switching ASICs to efficiently route Internet traffic. The peering switches (PS) are categorized into external peering switches, internal peering switches, and security peering switches, which respectively peer with ISPs on the Internet, internal cloud data centers, and anti-DDoS service hosts. Equipped with a CPU subsystem capable of supporting the computation and storage required for peering protocols, these programmable switches directly run the BGP stack. Consequently, each PS can independently execute the BGP speaker daemon to compute Internet-scale routes and directly forward packets to and from other peers. Moreover, the programmability of the ASIC subsystem provides flexible forwarding actions among the three types of switches and the hosts, and thus the switches and hosts collaborating to process traffic effectively and reliably.

By leveraging peering switches with several Tbps of forwarding capability, Janus significantly reduces hardware costs compared to host-based systems, as the majority of Internet traffic is forwarded by the switches. For traffic requiring customized forwarding strategies, we utilize a cluster of TE hosts to enable flexible and tailored traffic forwarding while employing asymmetric routing approaches to minimize the scale of the cluster (§ 4.2, § 4.3). With external PS forwarding Internet traffic to bypass hosts, Janus effectively prevents malicious traffic from directly attacking the hosts, redirecting only potential threats to anti-DDoS services, thereby greatly enhancing system reliability (§ 4.4). Additionally, we extend BGP to achieve fast route convergence through the design of a remote-link-sense mechanism and caching mechanism (§ 4.5).

3.3 Challenges

While the introduction of programmable switches at the peering edge offers significant advantages, deploying such a system at scale in production presents several critical challenges.

Effectively combining the strengths of hosts and switches. programmable switches excel in forwarding performance but are limited by memory constraints, while hosts offer abundant memory but lack high forwarding efficiency. Designing an efficient routing and forwarding plane that seamlessly integrates these two components into a unified architecture is challenging due to the added system complexity. Furthermore,

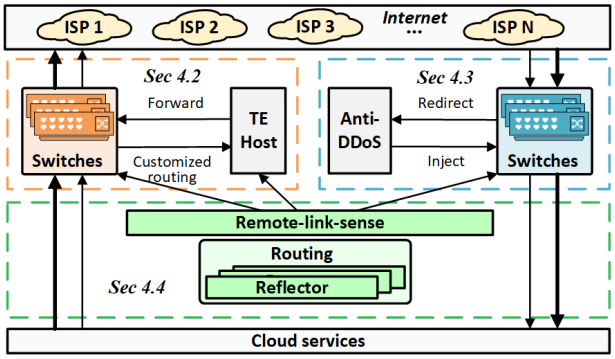


Figure 5: Janus overview.

enabling application-aware routing for traffic with customized forwarding requirements further complicates the design.

Limited entries in switching ASICs for Internet-scale FIBs. Internet peers typically advertise millions of routes, but the memory capacity of switch ASICs is constrained, supporting only a limited number of forwarding table entries (e.g., 320K). It is challenging to determine which subset of traffic should be forwarded by switches to optimize hardware utilization and minimize costs.

Efficiently handling malicious traffic. Handling malicious traffic directly at edge host servers is challenging, as certain attacks can exhaust CPU resources, leading to application traffic loss before detection. Leveraging high-performance packet forwarding infrastructure to bypass the hosts is crucial for mitigating such threats. However, it is essential to manage the responsibilities between the forwarding infrastructure and hosts so that this approach maintains network performance and introduces minimal additional costs.

Slow route convergence. Cloud services, including finance, gaming, and AIGC, are highly sensitive to packet loss, making route convergence time critical to their availability. Traditional failure recovery methods used in data centers or local peering are inadequate for Internet-scale cloud peering systems, as edge peering systems must handle millions of route updates. Consequently, it is imperative to design a mechanism to achieve faster route convergence.

4 System Design

We begin by presenting an overview of Janus (§ 4.1), a novel Internet peering routing system. Next, we delve into three key components of Janus: switch-based traffic dispatch with application-aware routing support (§ 4.2, § 4.3), efficient DDoS traffic redirection (§ 4.4), and accelerated route convergence (§ 4.5).

4.1 Janus overview

Existing peering routing systems face significant challenges in effectively addressing malicious traffic and maintaining cost efficiency. The objective of Janus is to enable robust and efficient cloud peering routing while overcoming these limitations. Figure 5 provides an overview of Janus, which efficiently manages both inbound and outbound network traffic while substantially reducing network hardware costs. For outbound traffic, we employ a traffic dispatch module to route traffic to either hosts or switches, enabling cost-effective and customized routing.

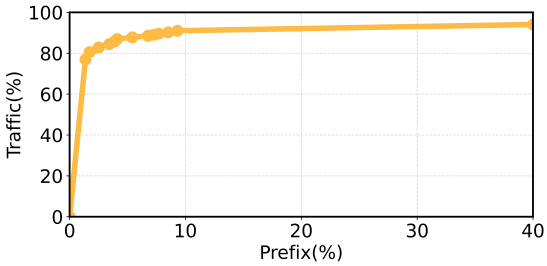


Figure 6: Few routes can carry most of the traffic.

More specifically, we run BGP sessions to compute Internet-scale routes on both switches and hosts, while asymmetrically offloading them to their FIBs based on their individual capacities. On the forwarding plane, the majority of traffic processing is offloaded to the peering switches, with only a small portion handled by the hosts. This significantly reduces the number of hosts required for packet processing compared to existing approaches (§ 4.2 & § 4.3). For inbound traffic, the system fully offloads processing to switches, mitigating the risk of hosts being targeted by attacks. The system identifies potential malicious traffic and redirects it to an Anti-DDoS service, a dedicated cluster designed to filter out malicious traffic, which reinjects legitimate traffic back to the switches for further forwarding (§ 4.4). Additionally, to enhance system reliability, we implement a fast route convergence mechanism to accelerate the Internet-scale route convergence process (§ 4.5).

4.2 Outbound traffic offloading

A key design choice in Janus is the introduction of programmable switches at the peering edge to harness the powerful forwarding capabilities of switching ASICs for routing Internet traffic. However, the limited memory capacity of switching ASICs prevents them from storing the entire Internet-scale forwarding table. This limitation prompted us to analyze specific routes and their associated traffic patterns. Figure 6 presents a two-year measurement study, revealing that a very small proportion of routes handle the majority of Internet traffic. Specifically, the top 1.7% of routes account for 80% of the total traffic, while 7.6% of the routes carry over 90% of the traffic. We find that this phenomenon is primarily driven by large-scale video services in our production network, which account for the majority of network traffic across the global cloud network. This observation inspired us to offload a small subset of routes to switches to handle most of the traffic, leaving hosts to process only a minimal portion of the network traffic. However, effectively integrating both switches and hosts into a unified architecture poses significant challenges in achieving a seamless routing and

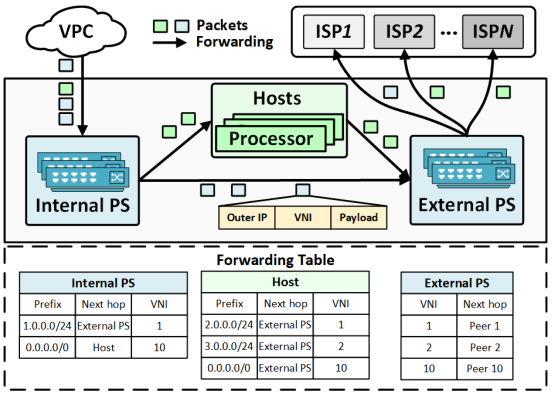


Figure 7: Traffic dispatch module.

forwarding plane. Next, we will present the detailed design of Janus forwarding and routing plane.

Traffic dispatch module. As mentioned before, we need to use both peering switches and hosts to forward the packets. As shown in Figure 7, we present a forwarding plane consisting of two peering switches (PS) and several hosts. The internal PS receives the packets from cloud services and redirects them to the external PS or hosts according to the forwarding policies in the internal PS. The packets directed to the hosts are either those whose corresponding entries are not offloaded to the internal PS's small FIB, or those requiring finer-grained policies for application-aware routing. The hosts store the completed FIBs and can forwards the packets redirected from the internal PS. The external PS directly connects with the peers on the Internet, and can forward every packet from the internal PS or hosts to the corresponding peers.

Moreover, the forwarding policy in the internal PS is periodically updated to accommodate to the changing traffic patterns and ensure most traffic is forwarded by the switches. Specifically, a traffic monitoring system (outside the scope of Janus) will mirror the traffic and keep statistics on the traffic volume by Internet IP prefixes. It will determine prefixes with large traffic volumes and instruct the internal PS to offload their routing entries. If the application's requirements change—for example, to support application-aware routing—we can shift the forwarding entry from the switch to the host. On the other hand, once the traffic on the host servers exceeds the predefined threshold, we shift the route from the host to the switch to prevent the host from becoming overloaded.

Asymmetrical FIBs. A crucial challenge in Janus is the limited memory capacity of switch ASICs, which cannot accommodate the entire Internet-scale FIB. Janus addresses this by asymmetrically synchronizing the FIBs computed by the BGP speaker daemon to switches and hosts.

Specifically, Janus runs BGP sessions on all peering devices (i.e., switches and hosts). For both internal and external PS, the BGP session calculates the entire Internet-scale routes in the CPU subsystem, but only a small subset of these routes is synchronized to the FIB in the ASICs. In contrast, hosts, with their larger memory capacity, offload the entire routing table

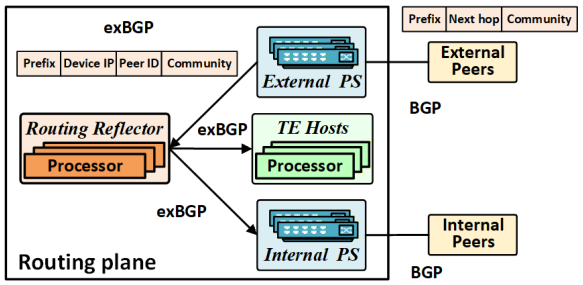


Figure 8: Routing information synchronization with exBGP.

to their FIBs. This allows hosts not only to forward traffic that is unmatched in the internal PS but also to customize routes for specific traffic to meet the requirements of cloud services. Additionally, the number of forwarding entries in the switch ASICs can be dynamically adjusted to control the traffic volume handled by the switches.

Routing information synchronization. Synchronizing routing information, including the device IP (identifying the local forwarding device) and peer id, across multiple devices (e.g., internal PS, external PS, hosts) within a peering routing system can be difficult. These devices are abstracted as a single large BGP speaker that communicates with peers through BGP. However, BGP itself cannot synchronize routing information among the devices or coordinate forwarding actions between them.

To address this, we extend BGP with a protocol we call *exBGP*. As illustrated in Figure 8, when a new route is updated at the external PS, exBGP enhances the traditional BGP announcement by including two additional attributes: device IP and peer ID, alongside the standard IP prefix and community. Here, the device IP refers to the local IP address of switches or hosts, while the peer ID identifies the Internet peer. Both attributes can be pre-configured. The enhanced announcement is then sent to the routing reflector, which propagates it to the remaining devices within the peering system. Upon receiving the announcement, these devices update their routing information accordingly.

This mechanism ensures efficient synchronization of routing information, enabling seamless integration of new devices or updates to routes within the peering system. Moreover, the peer ID and device IP in exBGP contribute to our design for fast route convergence, which we elaborate on in § 4.5.

Packet forwarding workflow. Figure 7 illustrates the detailed packet forwarding workflow along with the forwarding tables in the devices, enabled by the programmability of the PS forwarding plane. When the internal PS receives a packet from cloud services, it first looks up its forwarding table, encapsulates a VXLAN header with the VNI identifier indicating the peer ID, and then forwards the packet to the next hop (i.e., external PS or host). If the IP prefix of the packet is not found in the forwarding table, the default policy forwards the packet to the hosts, ensuring that all packets are processed and forwarded appropriately. Upon reaching the host, the packet's

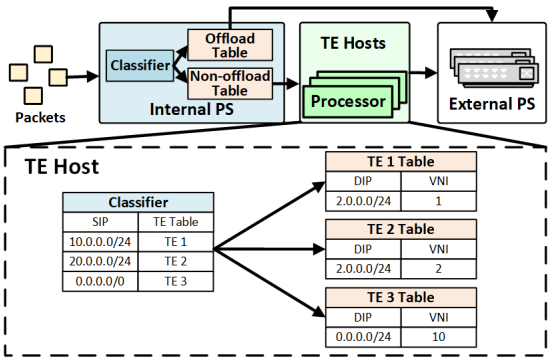


Figure 9: Classifier in the TE host.

VNI is updated according to customized routing strategies to enable application-aware routing. The host then forwards the packet to the external switch for further processing. Finally, when the external PS receives the packets, it looks up its forwarding table to map the VNI identifier to the specific peer, ensuring that each packet is forwarded to the correct peer.

4.3 Application-aware routing

Since packets sent to hosts are processed with finer granularity, we now present our design for supporting application-aware routing. We highlight the aspects that distinguish our approach from traditional host-based routing systems [25].

Two-level traffic classifier. Cloud services impose new requirements for quality of service, particularly in traffic engineering (TE), and demand high feature velocity deployment. To keep pace with these rapidly evolving needs, we introduce a classifier module to identify traffic in both internal PS and hosts, as shown in Figure 9. Specifically, we add a new forwarding table, called the non-offload table, in the internal PS to handle traffic for cloud services with specific requirements. Packets arriving at the internal PS are classified to determine which forwarding table to use, and packets requiring TE are sent to the hosts for further processing.

In hosts, we establish multiple TE tables to forward packets to different peers, thereby meeting the specific requirements of services. For example, for end users with a destination IP (DIP) of 2.0.0.0/24, when accessing different cloud services (i.e., different source IPs), the packets originating from the cloud will be forwarded to the users through distinct VNI identifiers, influencing the path selection. Note that the VNI identifier in the header is used to indicate the peer ID. The classifier can incorporate various packet attributes, such as IP prefix or differentiated services code point (DSCP), enabling the host to forward packets based on specific requirements.

Overall, this two-level approach incorporates a coarse classifier in the internal PS and a finer classifier in the hosts, enabling rapid adaptation to service demands.

Elephant flow offloading and hash-based identification. Although most traffic is offloaded to the switches, an elephant flow whose route entry has not yet been offloaded may arise. This can occur, for example, when a client that is usually served with small volumes of traffic temporarily experiences a large volume of traffic. In such cases, elephant flows from the cloud services are forwarded to the hosts, potentially overwhelming the CPU processing capacity.

A promising approach is to identify the elephant flow and redirect them to the switches. However, it is resource-consuming to count the traffic volume of each flow to identify the elephant flow in the expensive serves. Therefore, we design a simple but efficient two-level hash-based mechanism. Specifically, we first hash the flows to a few flow buckets based on the hash value of their five-tuples. Therefore, the host only needs to count the traffic volume of each bucket, which is much less than the number of flows. Given that the traffic forwarded to the host is considerably less than that to switches, the existence of even a single elephant flow will greatly increase the traffic volume in a bucket. Once a bucket is identified, we then dig deeper to count the traffic volume of each flow in this bucket to identify the exact elephant flow. Upon identifying this flow, the IP prefix and the corresponding forwarding entry of this flow are offloaded to the internal PS whose next hop is the external PS to bypass the host, enhancing system robustness.

This approach still enforces application-aware routing for elephant flows. The offloaded forwarding entry is computed based on the application-specific policy in effect at the onset of an elephant flow. Because elephant flows are typically short-lived (90% of them last $<$10s), we expect the offloaded elephant flows to continue following the host policies. After an elephant flow ends, the offloaded entry will be retired during the periodic offload-table update, and subsequent flows are processed in the hosts.

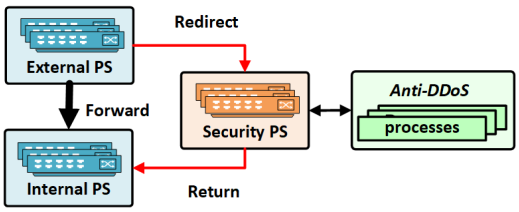


Figure 10: Security peering switches are introduced to redirect and inject inbound traffic.

4.4 Handing DDoS of Inbound Traffic

The peering routing system must ensure high robustness to defend against frequent malicious traffic at the edge, particularly DDoS attacks originating from the Internet [25]. Existing host-based peering systems struggle to mitigate DDoS attacks, significantly impacting cloud services. By leveraging the powerful forwarding capabilities of programmable switches, we demonstrate how Janus effectively addresses DDoS attacks.

Two-stage architecture bypassing hosts. Figure 10 illustrates the architecture of Janus for inbound traffic. We introduce a new switch, referred to as the security peering switch (PS), which connects the external PS to the internal PS.

Unlike outbound traffic, which requires millions of entries for packet forwarding, inbound traffic only requires approximately 50K routes. This reduction is achieved through precise IP address planning within our cloud network. This approach can also be applied to other cloud networks, even for larger clouds, since the number of inbound routes can be precisely controlled and planned. Consequently, we can offload all inbound traffic to the switch, leveraging its powerful forwarding capabilities without involving hosts, thereby mitigating the risk of malicious traffic directly targeting them.

Specifically, similar to the routing information synchronization for Internet peers, the routing information of internal peers is updated using exBGP (Figure 8). The external PS offloads all inbound traffic IP prefixes to its ASICs. Application traffic is directly forwarded to the internal PS to minimize packet latency. However, potential DDoS traffic is first redirected to the security switch. The security switch connects to the anti-DDoS service (typically a cluster of host servers), which filters out malicious packets and re-injects legitimate application traffic back to the security switch.

Three-layer DDoS scrubbing mechanism. We employ a three-layer DDoS scrubbing mechanism in Janus, to tailor for filtering out most malicious traffic in the switches.

At the first layer, the external PS identifies suspicious DDoS traffic according to IP prefixes, which are periodically updated by the anti-DDoS service. The suspicious traffic is redirected to the security PS.

At the second layer, the security PS applies ACL rules to filter out simple types of DDoS traffic, such as ICMP flood and TCP-flood. The ACLs are simple and use L3 and L4 information (*e.g.* five tuples), and they can be implemented on programmable switches. Traffic that is not filtered by the security PS ACLs is forwarded to the anti-DDoS service for deeper inspection.

The anti-DDoS service is the final layer of scrubbing. To mitigate subtle DDoS traffic (*e.g.* L7 DDoS), it implements complex and flexible rules, such as deep packet inspection and fingerprint detection. The traffic that passes the rule is considered normal application traffic and is re-injected into the security PS and then into the internal PS. The anti-DDoS service also periodically updates its detection rules and offloads the relevant prefixes and simple ACL rules to the external PS and the security PS via exBGP. While the detailed implementation of the anti-DDoS service is beyond the scope of this work, we focus on its functionality in efficiently handling malicious traffic from the Internet.

Blocking rules in external PS. To further reduce the traffic volume processed by the security PS and the anti-DDoS service, the anti-DDoS service can instruct the external PS to block specific prefixes when necessary, thereby effectively filtering out a substantial portion of malicious traffic at hardware.

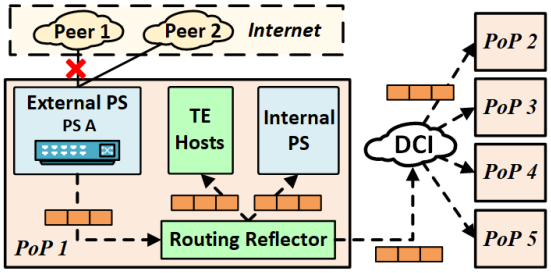


Figure 11: Remote link sense where a routing reflector broadcasts the failure messages to all devices in the peering systems.

4.5 Fast route convergence

In cloud peering systems, failures such as local link failures, BGP connection teardown events, and peer failures are inevitable, as edge peering interacts with numerous peers on the Internet. In the conventional approach, upon detecting a failure, the peering devices recalculate the routes locally. Once a new route is computed, the device immediately propagates a BGP withdrawal message to all other devices within the system. These devices then recalculate their routes based on their respective routing strategies. However, since all switches and hosts run BGP speakers to compute Internet-scale routes, a single failure can trigger updates for millions of routes. The convergence speed of this approach is limited to tens of thousands of routes per second, resulting in convergence times that can span several minutes. This prolonged convergence leads to packet loss, which is particularly detrimental to loss-sensitive applications. To address this, we present our key design for achieving fast route convergence.

Remote link sense. To accelerate route convergence in cloud peering, we introduce a remote-link-sense (RTS) mechanism. The core idea of RTS is to propagate detailed failure information to all devices in the network using exBGP (see Figure 8). Specifically, as illustrated in Figure 11, if the link between external PS A and peer1 fails, external PS A reports the link failure information to the routing reflector. The failure message includes the device IP, the peer ID associated with the failed link, and the event details. The routing reflector then broadcasts this failure information to all devices in the peering system. In distributed peering systems, the reflector also disseminates failure messages through the DCI.

Upon receiving the failure message, each device examines its routing table for entries involving external PS A and peer1, and updates its routes accordingly. This proactive approach enables devices to update their routes promptly when a failure occurs anywhere in the network.

Batched route update. While the RTS mechanism allows remote peering devices to detect link failures, updating routes sequentially in an Internet-scale forwarding table can be time-consuming. To address this, we introduce a cascaded forwarding table deletion mechanism that enables batch updates, significantly accelerating convergence. Specifically, we organize the forwarding table into linked lists, where each list

corresponds to a unique combination of device IP and link. When a route is updated, the new route entry is appended to the appropriate linked list. This structure ensures that all route entries associated with the same device IP and link are grouped together.

In the event of a failure, peering devices can quickly locate the relevant linked list using the device IP and link information. They can then delete all affected entries in a single batch operation, eliminating the need for a sequential scan of the entire forwarding table.

Cached route calculation. To further enhance routing update efficiency, Janus leverages caching to store the results of complex route processing in both switches and hosts. Specifically, upon receiving a route message, the peering device performs extensive calculations on community attributes, such as AS numbers, to determine the routing outcome (e.g., whether the route should be updated in the routing table). Since multiple route messages often share identical community attributes, recalculating these attributes repeatedly for millions of routes is time-consuming and significantly impacts convergence time.

To address this, Janus caches the calculation results. When processing a route message, the peering device can directly retrieve the precomputed results from the cache, bypassing redundant calculations and substantially reducing convergence time. By default, the cache is configured with a fixed capacity and employs a least recently used (LRU) strategy.

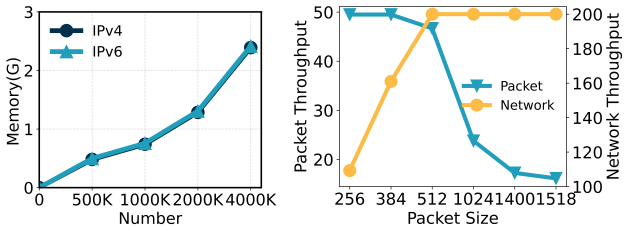
5 Evaluation

Janus is designed to carry all of cloud peering traffic. We firstly set up a testbed to conduct the pressure test of a single programmable switch and host server (§ 5.1). We then gradually introduce network traffic to Janus. We study the efficacy of our proposed switch-based peering routing (§ 5.2, § 5.3, § 5.4). Finally, we provide the convergence time of Janus as compared to existing open-sourced systems (§ 5.5), as well as the application-aware routing performance (§ 5.6)

5.1 Testbed evaluation

Experiment setup. The programmable switch is combined of a CPU subsystem equipped with an Intel X86 CPU featuring 4 cores and 16GB of memory and a switching ASIC subsystem from Broadcom Trident3 with a maximum of 3.2Tbps forwarding capability and 320K forwarding table entries. The main system module in the programmable switch is implemented with NPL due to internal deployment constraints. As for the host server, it is equipped with an Intel R Xeon(R) CPU featuring 48 cores and 320GB of memory, along with a dual 100 Gbps port NIC, using an in-house user space networking stack based on data plane development kit (DPDK) to process the packet. All hardware devices and software installed are identical to the production network, demonstrating the consistency of our experimental results.

Programmable switch. Figure 12(a) shows the memory usage when varying the number of routes in the switch CPU



(a) Switch (b) Host

Figure 12: Pressure test of switch and host.

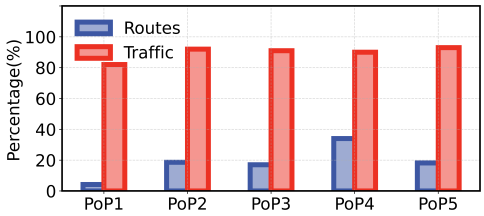


Figure 13: The proportion of routes in the ASIC and traffic forwarded by the ASIC.

subsystem. We observe an approximately linear relationship between memory usage and the number of routes and it takes 2.5 GB memory to store the number of routes up to 4M. Based on our experience, the maximum number of routes from our peering is about one million, indicating that the CPU subsystem has enough memory to store all the peering routes. Another observation is that the memory usage for IPv4 routes is the same as the IPv6. This is attributable to our choice to malloc a block with the maximum memory required for a single route to reduce the memory fragmentation. We then frequently add and delete a batch of routes to test the CPU utilization. We find the utilization is 20% at the peak, indicating that it can sufficiently handle the route update.

Host server. Since the host server uses DPDK to process the packet, a natural question we ask is how much traffic can be processed by a host server at a safe threshold. We generate VXLAN packets and the DPDK decapsulates the packet header of the arrival packet, looks up the TE table based on the header, and encapsulates the packet header. As shown in Figure 12(b), the packet throughput decreases when the packet size increases. When the packet size is larger than 512B, the host can forward at a line rate of 200Gbps. However, it achieves 50Mpps for packets of 384B with a forwarding rate of 160 Gbps. Generally, the average packet size on the Internet is about 384B. We should reserve 20% of capacity for traffic bursts and packet pattern changes and finally set the forwarding performance of the host to 120Gbps.

5.2 Outbound traffic offloading

A small part of routes forward most of network traffic, which motivates us to offload part of the routes to the AISCs, leveraging its high forwarding capability. Here, we simply configure the switches uniformly to offload 160K routes that encounter most traffic to the ASIC. On the one hand, the number of

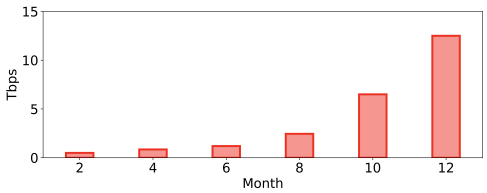


Figure 14: The total traffic carried on Janus over time.

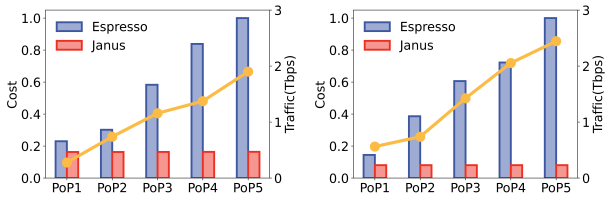
routes from the peer is one million, indicating 100K forwarding entries have forwarded most of the traffic (see Figure 6). Even if we increase the entries in ASICs to their limit, the total traffic offloaded will only increase by 5%. On the other hand, we always do not use all capabilities in the ASIC, leaving room for other operations. Figure 13 shows the traffic offloaded to the ASIC and the proportion of routes in the ASIC we measured from five PoPs. We observe that the number of routes in the ASIC only takes about less than 10% of total routes, but account for at least 82% of network traffic. There is a tradeoff between the number of routes in the ASIC and the portion of traffic offloaded. For a more precise control for each PoP, network operators can change the number of entries in the ASIC to change the offloaded traffic volume. For example, if network operators increase the number of routes in the ASIC in PoP1, more network traffic will be forwarded by the ASIC.

5.3 Network hardware cost

Janus is designed to carry all Internet traffic, including inbound traffic and outbound traffic. During the process of large-scale deployment, we started with the lower-priority network traffic for gray-box testing before Janus was fully deployed in production. With time it is now carrying high-priority network traffic. Figure 14 depicts the total traffic carried by Janus. We observe that traffic grows exponentially. For last four months, network traffic on Janus grew 3× more than the total.

Figure 15 depicts the total cost (normalized) of network hardware at five PoPs. We compare our proposed Janus with a recent benchmark scheme namely Espresso [25], which only uses the host servers to achieve global Internet peering. The cost is calculated by normalizing the hardware cost (hosts and switches) per 100 Gbps of traffic. Since Janus consists of both host servers and switches, we weight the cost according to the proportion of traffic handled by each, respectively.

When the outbound traffic is 400Gbps in Figure 15(a), the cost of Espresso and Janus is quite close since Espresso only needs to use a small number of host servers to handle the outbound traffic, while Janus takes advantage of both switches and host servers to process them. However, as the number of traffic grows up to 2500Gbps, the cost between Espresso and Janus becomes large that Janus can reduce the hardware cost by 84%. Espresso requires a large number of servers to handle the outbound traffic, resulting in a quite large network cost. However, Janus only uses a small number of host servers



(a) Outbound (b) Inbound

Figure 15: The outbound and inbound network cost and traffic volume (yellow line) carried by Janus in five PoPs.

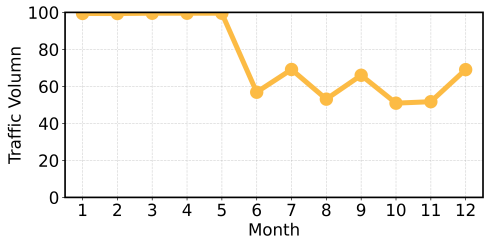


Figure 16: Internet traffic (normalized) reduced due to ACL rules in the external peering switch.

to handle a small part of traffic since most of the traffic is offloaded to the switch. Meanwhile, the more outbound traffic Janus carries, the more cost Janus saves. As for inbound traffic in Figure 15(b), the cost for Janus is also substantially smaller than Espresso. Janus can reduce the cost by more than 90% for the inbound traffic because all inbound traffic in Janus is offloaded to the switch and the cost of switch per 100Gbps is less than one-tenth of the server (see Figure 2(b)). Overall, Janus can reduce the average network hardware cost by 78% over these five PoPs.

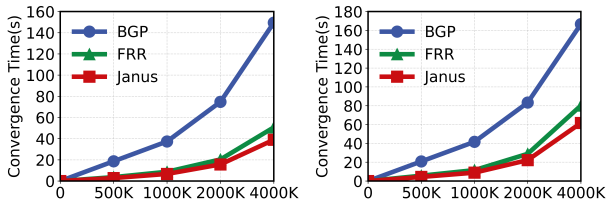
5.4 Inbound DDoS traffic offloading

As shown in § 4.4, part of (potential) malicious traffic is forwarded to the Anti-DDoS system for DDoS mitigation. For our initial version, all malicious traffic will be sent to the Anti-DDoS system (i.e., always a group of servers) for DDoS mitigation. It consumes a lot of server resources to handle malicious traffic. Therefore, we migrate the ACL rules at the external peering switch to block malicious traffic. Figure 16 depicts Internet traffic handled by the Anti-DDoS system measurement for 12 months. After the deployment of the ACL rules in May in the external peering switch, the total network traffic handled by the Anti-DDoS system is reduced by at most 50%. This indicates that we can further reduce the cost of the expensive server at Anti-DDoS system.

5.5 Comparison of BGP Speakers

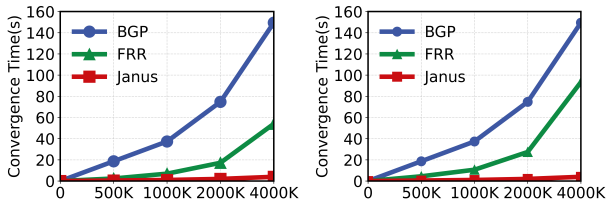
In the early time, we had to choose between an open-source BGP stack, e.g., Quagga[1] [1], FRR [2]. We settled on FRR as the leading open-source candidate and conducted a detailed comparison to the traditional BGP stack on a widely used commercial router. The main reason is that a lot of effort

[1] Quagga project is deprecated now.



(a) IPv4 (b) IPv6

Figure 17: BGP advertising convergence time for IPv4 and IPv6.



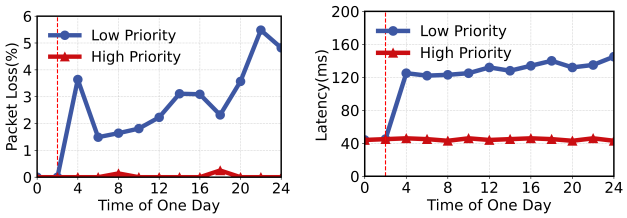
(a) IPv4 (b) IPv6

Figure 18: BGP withdrawal convergence time for IPv4 and IPv6.

has been spent on optimizing FRR in our cloud. The main metric is the convergence time for both BGP advertising and withdrawal for both IPv4 and IPv6 routes. We create route sets of different sizes from 500 thousand to 4 million and study the convergence time with different BGP stacks.

For the BGP advertising, we created route sets of different sizes from 500 thousand to 4 million (maximum routes from a peer). As shown in Figure 17(a), we observe that the traditional BGP stack takes about 150 seconds to complete convergence. The main reason is that a large amount of time is consumed to process complex routing policies with different attributes. For FRR, it takes 50 seconds to complete convergence due to its comprehensive software BGP stack. Janus can further reduce the convergence time by 20%. This result is attributed to our design that we introduce a cache mechanism during the process of the routing policy. When routes with the same routing attributes which have been processed before, we can directly obtain the result from the cache without processing the routes. The result is similar for IPv6 shown in Figure 17(b).

As for the BGP withdrawal in Figure 18, we observe a similar result for these traditional BGP stacks. An important observation is that Janus shows a promising result that the convergence time does not change as the number of routes increases. Even if the number of routes reaches 4 million, Janus takes 4 seconds to complete convergence, which is orders of magnitude faster than existing approaches. This is because Janus leverages a remote-link-sense mechanism and batched routing update to achieve fast convergence. Once a link is down, causing millions of routes to be updated, Janus can fastly react to the failure by deleting a set of corresponding forwarding entries in the table.



(a) Packet Loss (b) Latency

Figure 19: The packet loss and latency for both low and high priority traffic.

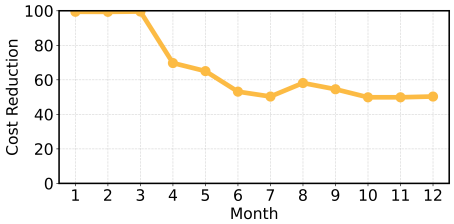


Figure 20: The bandwidth cost of a specific PoP.

5.6 Application-aware Routing

Finally, we discuss the application-aware routing. Janus connects to tens of peers at the edge with different peers providing multiple quality of service (QoS) classes. Some peers provides high-availability or low-latency links along with higher bandwidth cost, while some peers provide low-availability or high-latency links along with low bandwidth cost. As for the network traffic, it can be classified into several classes [6, 14, 23]. For example, higher-QoS traffic is always loss-sensitive and time-sensitive and lower-QoS traffic is always loss-tolerant. For the traditional router-based system, it can only route the application traffic to a specific peering once configured. In our previous deployment, both low and high priority traffic will be routed to the peering ensuring high QoS to meet the requirements of the high priority traffic. However, this approach requires high bandwidth cost. Upon the deployment of Janus at 2 am in Figure 19, we observe that packet loss of low priority traffic increases to a high value, ranging from 1.5% to 5.5%. Meanwhile, the latency of high priority traffic is reduced by 3×. This indicates a precise control of low priority traffic which is routed to the peering with low QoS. Figure 20 depicts the total cost of a PoP measured for a year. We observe after the deployment of Janus in March, the bandwidth cost is reduced by up to 50%. This is attributable to the precise routing mechanism that our Janus can effectively route the low priority traffic to the lower-cost peers, and thus the bandwidth cost for these applications is greatly reduced.

6 Lessons learned

In the early time, we had to choose between the host server-based infrastructure (i.e., Espresso [25]) or the commercial routers (e.g., Edge Fabric [18]) to meet the QoS demand and reduce the network bandwidth cost. Our first version of Janus is similar to Espresso which relies on the host server

to establish BGP peering since it makes the edge network more flexible to meet application QoS requirements. A lot of lessons have been learned during the initial deployment of our switch-based architecture at the Internet peering edge and iterating on the implementation based on actual experience. In this section, we outline some of important lessons.

Mitigating DDoS from Internet. We initially considered routing the outgoing traffic to the peering meeting the QoS requirements. However, during the deployment of Janus, we observed that Janus was not able to effectively address the DDoS attack from the Internet. When the DDoS traffic is processed at the edge server, it will always exceed the processing capability of a single CPU core, causing the packet loss for the high-priority network traffic which is sensitive to the packet loss. This issue is mitigated by designing a DDoS scrubbing mechanism where the internal traffic gradually decreases during our iterative deployment.

Establishing BGP peering at peering switch. Since our edge network needs to connect to thousands of peers all around the world, this approach should add new protocols between the server and the peering for network debugging and failure location, greatly increasing the network complexity and the workload and difficulty for network operators. Meanwhile, different peers may not always support the network protocol used in our production network. Therefore, we directly run the BGP at the peering switch at the edge and the peering switch directly connects to the peering switches to establish the BGP connection, greatly reducing the complexity of the network.

Disaggregating routing from controller. We have initially considered the SDN based centralized controller to calculate and synchronize routing information to the edge peering switch. However, we realized that the centralized control encountered several issues in the global network. On the one hand, once the controller calculates the routing information and needs to synchronize it to all PoPs, the different latencies across the world result in routing information configured asynchronously. On the other hand, such SDN based centralized control greatly increases the failure domain. Therefore, we adopt a disaggregated approach where the controller is only responsible for device configuration, topology management, detection, health monitoring, etc. Routing information is complete through exBGP. The routing reflector leverages exBGP to synchronize routing information among devices. Once the device updates routes, the change should be propagated by routing reflector, therefore reducing the failure domain.

7 Related Work

Cloud Internet BGP peering. BGP has long attracted research for many related topics. Cloud providers have been one of the spots for the BGP research community given their scale and connectivity to multiple ASes. Existing research has explored BGP configuration verification [5, 8, 26], the connectivity of cloud providers [4, 12, 27], the routing policies [21, 22], inter-domain routing failures [9, 13], BGP security vulnerabilities [15, 19, 20], and more. As one of largest cloud provider, our work focused on the Internet peering edge.

There have been several proposals focusing on Internet peering edge [18, 25] to route application traffic at the Internet peering edge to meet the QoS demand of each applications. Edge Fabric [18] relies on commercial routers with the SDN-based controller to perform traffic scheduling by synchronizing routes to the routers through the controllers. However, due to the inherent defects (e.g., low feature velocity, substantial management and configuration complexity) of commercial routers, it is not able to pace up to the fast changing product requirements. Espresso [25] relied on the host server and SDN controller to achieve fine-grained traffic engineering. However, on the one hand, this approach is difficult to effectively deal with malicious traffic (e.g., DDoS), which further hinders it being deployed in a large scale. On the other, this approach is not cost-effective for the network hardware. In contrast, Janus is a radical new infrastructure which uses switches to offload the majority of network traffic to reduce the network cost and the DDoS problem with a lightweight component.

Cloud gateway. A lot of efforts have been devoted to study the design of various cloud gateway for various network functions, such as virtual router [21], virtual private network [3], load balancer [7], and network address translation [25]. In recent years, with high packet forwarding capability of (programmable) switch, some work always incorporated the hardware device into their cloud gateway [10, 16, 24]. However, these works only focused on the internal network. In contrast, the edge network differs greatly with the internal network, which is responsible for connecting tens of peers with million of routes. Therefore, we propose combining the advantage of both switch and host servers where the majority of traffic is forwarded with switches with a small number of routing tables, while a small part of traffic is forwarded with host servers with a large number of routing tables.

8 Conclusion

We introduce Janus, a radical peering architecture that deploys programmable switches at the edge to intelligently offload both inbound and outbound traffic via a novel dispatch module. Meanwhile, Janus enhances security by redirecting threats to anti-DDoS services and delivers orders of magnitude faster route convergence. Evaluation through testbed and production experiments, Janus significantly improves routing efficiency, reduces latency, enhances security, and outperforms traditional solutions in scalability, fault tolerance, and response time under various network conditions.

Acknowledgment

We sincerely thank our shepherd Michio Honda and anonymous NSDI reviewers for their insightful comments. We also thank the teams at Tencent for their advice on Janus. Gaogang Xie is the corresponding author.

References

[1] GNU Quagga Project, 2010. www.nongnu.org/quagga/.

[2] The FRRouting Protocol Suite, 2022. https://github.com/FRRouting/frr.

[3] M. T. Arashloo, P. Shirshov, R. Gandhi, G. Lu, L. Yuan, and J. Rexford. A scalable vpn gateway for multi-tenant cloud services. *ACM SIGCOMM Computer Communication Review*, 48(1):49–55, 2018.

[4] T. Arnold, J. He, W. Jiang, M. Calder, Í. Cunha, V. Giotsas, and E. Katz-Bassett. Cloud provider connectivity in the flat internet. In *IMC '20: ACM Internet Measurement Conference, Virtual Event, USA, October 27-29, 2020*, pages 230–246. ACM, 2020.

[5] M. Brown, A. Fogel, D. Halperin, V. Heorhiadi, R. Mahajan, and T. Millstein. Lessons from the evolution of the batfish configuration analysis tool. In *Proceedings of the ACM SIGCOMM 2023 Conference*, pages 122–135, 2023.

[6] M. Denis, Y. Yao, A. Hatch, Q. Zhang, C. L. Lim, S. Zhang, K. Sugrue, H. Kwok, M. J. Fernandez, P. Lapukhov, et al. Ebb: Reliable and evolvable express backbone network in meta. In *Proceedings of the ACM SIGCOMM 2023 Conference*, pages 346–359, 2023.

[7] D. E. Eisenbud, C. Yi, C. Contavalli, C. Smith, R. Kononov, E. Mann-Hielscher, A. Cilingiroglu, B. Cheyney, W. Shang, and J. D. Hosein. Maglev: A fast and reliable software network load balancer. In *Nsdi*, volume 16, pages 523–535, 2016.

[8] A. Fogel, S. Fung, L. Pedrosa, M. Walraed-Sullivan, R. Govindan, R. Mahajan, and T. Millstein. A general approach to network configuration analysis. In *12th USENIX Symposium on Networked Systems Design and Implementation (NSDI 15)*, pages 469–483, 2015.

[9] A. Haeberlen, I. C. Avramopoulos, J. Rexford, and P. Druschel. Netreview: Detecting when interdomain routing goes wrong. In *Proceedings of the 6th USENIX Symposium on Networked Systems Design and Implementation, NSDI 2009, April 22-24, 2009, Boston, MA, USA*, pages 437–452. USENIX Association, 2009.

[10] D. Kim, Z. Liu, Y. Zhu, C. Kim, J. Lee, V. Sekar, and S. Seshan. Tea: Enabling state-intensive network functions on programmable switches. In *Proceedings of the Annual conference of the ACM Special Interest Group on Data Communication on the applications, technologies, architectures, and protocols for computer communication*, pages 90–106, 2020.

[11] C. Labovitz, S. Iekel-Johnson, D. McPherson, J. Oberheide, and F. Jahanian. Internet inter-domain traffic. *ACM SIGCOMM Computer Communication Review*, 40(4):75–86, 2010.

[12] R. Landa, L. Saino, L. Buytenhek, and J. T. Araújo. Staying alive: Connection path reselection at the edge. In *18th USENIX Symposium on Networked Systems Design and Implementation, NSDI 2021, April 12-14, 2021*, pages 233–251. USENIX Association, 2021.

[13] C. Miao, Y. Xiao, M. Canini, R. Dai, S. Zheng, J. Wang, J. Bu, A. Kuzmanovic, and Y. Wang. Tensor: Lightweight bgp non-stop routing. In *Proceedings of the ACM SIGCOMM 2023 Conference*, pages 108–121, 2023.

[14] C. Miao, Z. Zhong, Y. Xiao, F. Yang, S. Zhang, Y. Jiang, Z. Bai, C. Lu, J. Geng, Z. He, et al. Megate: Extending wan traffic engineering to millions of endpoints in virtualized cloud. In *Proceedings of the ACM SIGCOMM 2024 Conference*, pages 103–116, 2024.

[15] A. Mitseva, A. Panchenko, and T. Engel. The state of affairs in BGP security: A survey of attacks and defenses. *Comput. Commun.*, 124:45–60, 2018.

[16] T. Pan, N. Yu, C. Jia, J. Pi, L. Xu, Y. Qiao, Z. Li, K. Liu, J. Lu, J. Lu, et al. Sailfish: Accelerating cloud-scale multi-tenant multi-service gateways with programmable switches. In *Proceedings of the 2021 ACM SIGCOMM 2021 Conference*, pages 194–206, 2021.

[17] Y. Rekhter, T. Li, and S. Hares. A border gateway protocol 4 (bgp-4). Technical report, 2006.

[18] B. Schlinker, H. Kim, T. Cui, E. Katz-Bassett, H. V. Madhyastha, I. Cunha, J. Quinn, S. Hasan, P. Lapukhov, and H. Zeng. Engineering egress with edge fabric: Steering oceans of content to the world. In *Proceedings of the Conference of the ACM Special Interest Group on Data Communication*, pages 418–431, 2017.

[19] P. Sermpezis, V. Kotronis, K. Arakadakis, and A. Vakali. Estimating the impact of BGP prefix hijacking. In *IFIP Networking Conference, IFIP Networking 2021, Espoo and Helsinki, Finland, June 21-24, 2021*, pages 1–10. IEEE, 2021.

[20] P. Sermpezis, V. Kotronis, A. Dainotti, and X. A. Dimitropoulos. A survey among network operators on BGP prefix hijacking. *Comput. Commun. Rev.*, 48(1):64–69, 2018.

[21] H. Shao, X. Wang, Y. Lu, Y. Yu, S. Zheng, and Y. Zhao. Accessing cloud with disaggregated {Software-Defined} router. In *18th USENIX Symposium on Networked Systems Design and Implementation (NSDI 21)*, pages 1–14, 2021.

[22] H. Svens and L. Hellberg. Analysis of bgp routing for major cloud service providers: A characterization of growth and accessibility, 2022.

[23] Y. Xia, Y. Zhang, Z. Zhong, G. Yan, C. L. Lim, S. S. Ahuja, S. Bali, A. Nikolaidis, K. Ghobadi, and M. Ghobadi. A social network under social distancing:{Risk-Driven} backbone management during {COVID-19} and beyond. In *18th USENIX Symposium on Networked Systems Design and Implementation (NSDI 21)*, pages 217–231, 2021.

[24] T. Xu, B. Xue, Y. Song, X. Wu, X. Peng, Y. Lyu, X. Wang, C. Tian, B. Ye, C. Nguyen, et al. {CyberStar}: Simple, elastic and {Cost-Effective} network functions management in cloud network at scale. In *2024 USENIX Annual Technical Conference (USENIX ATC 24)*, pages 227–246, 2024.

[25] K.-K. Yap, M. Motiwala, J. Rahe, S. Padgett, M. Holliman, G. Baldus, M. Hines, T. Kim, A. Narayanan, A. Jain, et al. Taking the edge off with espresso: Scale, reliability and programmability for global internet peering. In *Proceedings of the Conference of the ACM Special Interest Group on Data Communication*, pages 432–445, 2017.

[26] F. Ye, D. Yu, E. Zhai, H. H. Liu, B. Tian, Q. Ye, C. Wang, X. Wu, T. Guo, C. Jin, et al. Accuracy, scalability, coverage: A practical configuration verifier on a global wan. In *Proceedings of the Annual conference of the ACM Special Interest Group on Data Communication on the applications, technologies, architectures, and protocols for computer communication*, pages 599–614, 2020.

[27] B. Yeganeh, R. Durairajan, R. Rejaie, and W. Willinger. How cloud traffic goes hiding: A study of amazon's peering fabric. In *Proceedings of the Internet Measurement Conference, IMC 2019, Amsterdam, The Netherlands, October 21-23, 2019*, pages 202–216. ACM, 2019.