Net-P4ct: Enhanced WAN Bandwidth Fair Sharing Using P4 Programmable Switches

Haoran Chen*, Mingwei Cui*, Yihan Zou*†, Yihang Miao*, Suhan Jiang*, Damu Ding*†, Lirong Lai*
Ming Gao*, Rui Jiang*, Shengyuan He*, Anjian Chen*, Jiaming Shi*, Junjie Wan*, Yandong Duan*
Ruomin Fang*, Hongyu Wu*, Yongping Tang*, Qiao Kang, Guangrui Wu*, Xiyun Xu*
*ByteDance

Abstract

At growing internet companies like ByteDance, Wide Area Network (WAN) bandwidth sharing across diverse services with varying SLO requirements is a fundamental challenge. Conventional host-based enforcement systems, where agents identify and throttle traffic at the server end, face practical challenges such as "blind spot" traffic, kernel-dependent operational complexity, and significant server resource overhead. To address these issues, we present Net-P4ct, an in-network bandwidth enforcement system using P4 programmable switches. Net-P4ct improves both bandwidth guarantees and fair sharing by shifting dynamic QoS control into the switch data plane. Specifically, it achieves broader traffic coverage by combining host-side traffic tagging with a P4-switch pipeline, where service classification and QoS class assignment are performed. Based on observed traffic metrics, a centralized control plane determines real-time policy updates according to the max-min fair bandwidth allocation. We demonstrate the system's benefits including improved bandwidth utilization, reduced operational complexity, and lower per-byte processing cost. Net-P4ct has been deployed in ByteDance's production WAN for nearly a year, and we hope to share our experience with the community.

1 Introduction

Driven by diversified modern-day applications including cloud services, online live-streaming, and on-demand video services, internet companies are seeing drastically increasing volumes of network traffic and a diverse range of business types [14,16]. Due to the long deployment cycles and high provisioning costs associated with WAN (Wide Area Network) expansion, network operators strive to improve bandwidth utilization and resource efficiency, in order to meet the service-level objectives (SLOs) for various applications [23, 36, 39]. To cater to the business growth, the capacity of ByteDance's self-built WAN has increased from $O(10)$ Tbps to $O(100)$ Tbps in the past few years. Today, ByteDance's backbone network connects tens of self-built data centers (DCs) across regions, utilizing multiple third-party cloud service providers (CSPs) around the world [13].

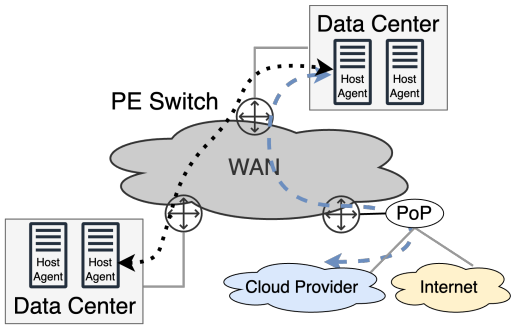


Figure 1: Existing WAN model

This hybrid inter-connect scenario imposes additional complexity and unique challenges for the network design and operations. The most common approach for bandwidth allocation in a large-scale backbone network is to deploy a massive number of host agents, which are co-located with applications/services at the server end [9, 25, 32]. Host agents are capable of monitoring host traffic, modifying packet headers, and performing traffic throttling[1], etc. Despite the ample industry deployment and experience, host-agent based solutions have some fundamental limitations, making it less than ideal in our complex WAN settings. *i)* It lacks coverage for traffic from certain "blind spots," e.g., if the traffic traverses the backbone network without going through the host, it cannot be captured by the system (see the blue dashed line in Figure 1). The influx of unregulated traffic from cloud providers and the Internet, directly destined for our remote data centers, places considerable strain on our private WAN's limited bandwidth. *ii)* It requires tremendous efforts to manage millions of hosts in mega-scale infrastructures, in addition to tens of versions of operating systems and kernels; *iii)* The total resource consumption for a large scale of host agents is significant.

†Co-corresponding authors.

[1]We use throttling and rate limiting interchangeably in this work.

In this work, we present Net-P4ct (pronounced as "net-pact"), a flexible and versatile bandwidth allocation and fair resource sharing system for WAN traffic using *P4 programmable switches* [10, 22, 29]. Net-P4ct is fundamentally different from host-based approaches, as it shifts the bandwidth enforcement and quality-of-service (QoS) control into the switch data plane. Our proposed system enhances traffic penetration by deploying P4 programmable switches (cluster) in conjunction to backbone Provider Edge (PE) switches in all regions, which covers all critical paths of WAN traffic. Acting as in-network "toll gates", P4 switches are responsible for assigning the corresponding QoS class to every packet traversing through the backbone network. We define the bandwidth request model, and work with other business teams to standardize the bandwidth sharing agreement. Based on real-time traffic metrics and resource availability from Traffic Engineering (TE) system and the inventory system, a centralized Net-P4ct controller cluster computes policy updates according to the max-min fair bandwidth allocation. The updates are pushed to P4 switches for real-time bandwidth enforcement.

There have been extensive studies on bandwidth sharing and management [21, 30, 31, 34]. However, most of the work and systems focus on sharing networks among tenants in the cloud computing environment. The WAN environment is inherently more complex due to differences in provisioning strategies, cost sensitivity, and service heterogeneity.

We briefly summarize our contributions as follows.

- We propose a generic and flexible in-network bandwidth management framework for broader coverage of WAN traffic compared to host-based solutions in hybrid scenarios.
- We implement an efficient bandwidth enforcement system that integrates a centralized controller with P4 programmable switches. The system is work-conserving, guarantees per-service minimum bandwidth, and provides max-min fairness across all services.
- We propose two enforcement policy alternatives, i.e., meter-based policy and statistics-based policy, and provide extensive numerical evaluations in real testbed.
- We share our deployment insights and operational experience in the real-world large-scale production network.

The remainder of this paper is organized as follows. Section 2 introduces background information and discusses our motivation. Section 3 presents an overview of Net-P4ct. Section 4 explains how effective bandwidth enforcement and fair bandwidth sharing are achieved with our proposed solution. Section 5 describes the implementation and interaction of all system components. In Section 6, we present the setup of our testbed and show extensive numerical results. Section 7 shows our production deployment insights. In Section 8, we review the related work. Finally, Section 9 concludes the paper.

2 Background and Motivation

In this section, we first introduce ByteDance's core network infrastructure, where a wide spectrum of services share the WAN bandwidth. We then highlight the unique characteristics of the workload and the corresponding challenges for effective bandwidth management. Finally, we summarize the key motivations behind our design.

2.1 ByteDance's cross-region services

Our backbone network provides connectivity for hundreds of applications with diverse business needs, including external clients, customer-facing applications and internal support services, such as Compute, Storage and so on. Each individual business scenario may have its distinct SLOs, e.g., bandwidth and latency. Similar to other related work [9, 25], we broadly categorize the network traffic into several QoS classes based on the business priorities and traffic patterns.

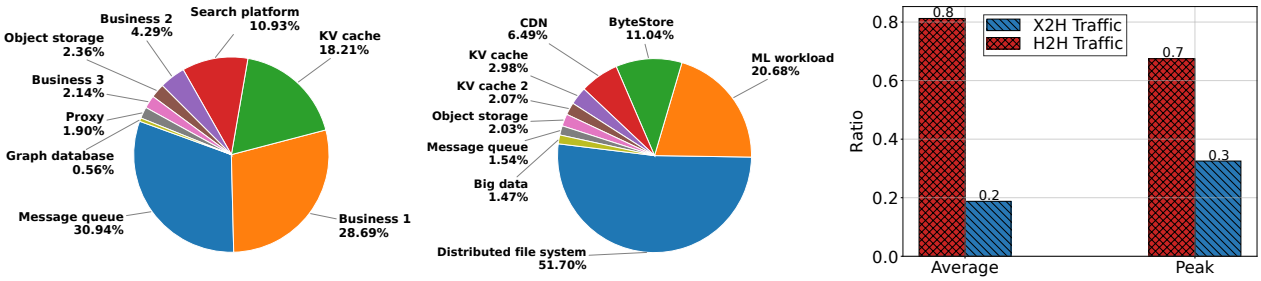
For ease of illustration, we categorize the traffic QoS classes into high priority class and low priority class. Figure 2 shows the major service traffic distribution in each priority class. *i)* We observed that each QoS class consists of diverse service types. A few infrastructure services (e.g., compute and storage) and main business applications account for majority of network traffic in each priority class. *ii)* Most business-related traffic belongs to high priority class. Even though some of them do not have large bandwidth requirement, their SLOs need to be guaranteed. *iii)* A large fraction of low-priority traffic belongs to distributed file system, which exhibits bursty traffic patterns [38]. Thus, it is an important and challenging task to enable effective bandwidth sharing with fairness for a large number of diverse services with distinct SLOs in our production system.

One unique scenario in our backbone network is the mixture of traffic originated from heterogeneous types of sources, e.g., self-managed DCs, CSPs and internet. Based on our service scenario, backbone traffic is categorized into the following two types.

Host-to-host (H2H): traffic originates from services running in internal servers (dotted line in Figure 1).

External-to-host (X2H): traffic originates from cloud services or the Internet to internal hosts (dash line in Figure 1). For instance, to improve workload flexibility, load balancers are deployed with cloud workloads. These load balancers forward traffic through our WAN to backend servers in remote data centers. This approach not only enables flexible workload placement, but also leverages our WAN infrastructure to deliver a more reliable and optimized service experience.

Figure 2(c) shows the average and peak traffic ratios of different traffic types in one of our backbone regions. Clearly, X2H traffic is an important scenario in our network, which accounts for 33% of the peak ingress WAN traffic.



(a) High Priority (b) Low Priority (c) H2H and X2H traffic ratios in one region

Figure 2: Traffic distribution in different QoS classes and different traffic types

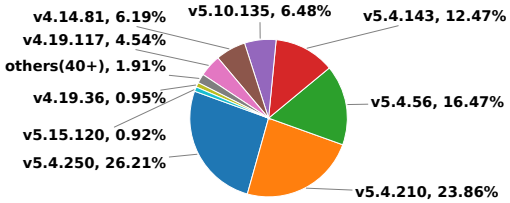


Figure 3: Linux kernel version distribution

2.2 Host-based Solutions Facing Challenges

Similarly to the Host Enforcer in [25], we have deployed a host-based daemon on every server named *NetAgent*. It utilizes eBPF (Extended Berkeley Packet Filter) in the Linux kernel to implement a set of functionalities, including host-traffic monitoring, packet header modification, and traffic throttling. However, such a traffic throttling design has encountered certain challenges in practice.

- **Traffic from "blind spots".** Due to the significant penetration of X2H traffic in our backbone network, part of the network traffic cannot be captured by host agents, causing "blind spots" in the bandwidth management system.
- **Maintenance costs of agents.** Millions of hosts in the mega-scale infrastructure, combined with mixed deployment of kernel versions, can incur tremendous operating and maintenance costs. Figure 3 shows more than 50 different kernel versions in the production. We have encountered incidents due to kernel version difference in the past.
- **Resource overhead.** Host agents are consuming resources (e.g., CPU cores) on the running hosts, especially for *throttling*. Based on our operational statistics, a million host agents will consume approximately $O(20k)$ CPU cores. This resource consumption will be billed to network team, incurring significant additional operational cost.

In practice, we have seen a few incidents caused by the above-mentioned reasons. Here, we show two examples to motivate our new design in this work.

Incident 1 - Kernel Bugs: In some rate-limiting cases, we observed that some packets are mistakenly marked with a very large timestamp value. FQ (Fair Queue) is a classless packet scheduler that is designed for flow pacing [1]. According to the Earliest Departure Time (EDT) algorithm used by NetAgent, these packets cannot be transmitted from FQ before their assigned timestamps [8]. The kernel bug caused inconsistent time bases used by the protocol stack and FQ: the protocol stack is using *CLOCK_TAI* (International Atomic Time) while FQ is using *CLOCK_MONOTONIC*. The packet sending process of some versions of the kernel, like IPVS (IP Virtual Server), will use the protocol stack's time base, resulting in a very large timestamp value. The bug was later fixed in new patches in kernel version 5.10 [5, 6].

Incident 2 - Fair Queue Saturation: In this incident, we observed that part of host's network interface cards (NICs) were not pingable after a 100Mbps rate-limiting intention. The root cause boiled down to the saturation of FQ in Linux Kernel (usually has size of $\approx 10k$ packets). Suppose the service is originally sending traffic at a 1Gbps rate and each packet's size is 1500 bytes, after NetAgent limit the traffic rate of this service, the backlog size will increase at the rate of $75k$ packets/s. As a result, the FQ will be quickly fully occupied. It is saturated with packets from the rate-limited service, whose timestamps are minutes later than the current time. Thus, these packets need to remain in the FQ for several minutes. Since FQ is a shared resource, this saturation prevents other traffic from being enqueued. This case demonstrates that improper configuration of FQ poses a risk of impairing unthrottled traffic. This issue was resolved in a later fix with a horizon configuration [7].

2.3 Need For Flexible In-network Solutions

As we mentioned above, the host-agent based solutions may not handle "blind-spot" traffic very well. One fundamental reason is that host agents are not located on the critical paths of all types of WAN traffic. For instance, X2H traffic will enter WAN from point-of-presence (PoP) without passing any hosts. In contrast, critical network devices (e.g., PE switches) are deployed directly in WAN, making them natural candidates for this purpose. However, conventional network devices can only provide limited flexibility, as switch fabrics are highly optimized for routing and traffic forwarding purposes. In ad-

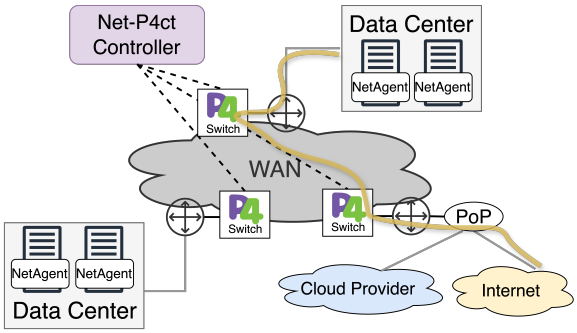


Figure 4: The overview of Net-P4ct

dition, frequently changing device configurations is not ideal for maintaining network stability.

The recent emergence of P4 programmable offers new possibilities to customize the data plane pipeline and enable more in-network functionalities. Given the advantages of P4 switches, our aim is to develop a flexible and cost-effective bandwidth sharing framework for WAN using in-network devices. Next, we will provide a high-level overview of our proposed solution.

3 Overview

In this section, we present Net-P4ct, a WAN-wide bandwidth management and enforcement system that utilizes the versatility and manageability of P4 programmable switches. Net-P4ct is designed to enable flexible QoS control and fair bandwidth sharing across all applications in geographically distributed data centers.

As shown in Figure 4, at each WAN ingress point[2], we deploy dedicated clusters consisting of P4 switches. All traffic, including that from hosts, cloud services or Internet, is processed by the attached P4 switches before being redirected to its destination over the WAN. These clusters form an entire programmable data plane that manages all outgoing traffic before it enters the backbone.

To enable per-service identification and traffic control:

- For H2H traffic, a lightweight agent (i.e., NetAgent) runs on each host and tags live packets with a service identifier. This allows P4 switches to recognize the traffic source at line rate.
- For X2H traffic, P4 switches infer the service identity directly from packet-level characteristics.

A central controller receives user-submitted bandwidth requests, allocates bandwidth resources accordingly, and computes enforcement policies based on real-time traffic metrics. These policies are then distributed to the P4 switches for in-network enforcement.

[2]WAN ingress points includes backbone PE switches at self-managed DCs and PE switches connected to CSPs and internet.

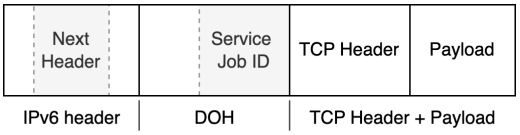


Figure 5: IPv6 destination options header for host-to-host communication

Unlike traditional systems based on rate limiting, Net-P4ct leverages P4 programmability to dynamically adjust QoS class according to the measured service rate and then maps the QoS class to the corresponding DSCP value at P4 switches. Afterwards, these DSCP values are assigned to priority classes, and packets are handled by the strict priority queuing mechanisms on WAN switches. This enables differentiated QoS class of traffic and fair bandwidth sharing among services.

The system is also designed with scalability and reliability in mind. Further implementation details, including resilience mechanisms, are described in Appendix A.

4 Bandwidth Guarantees with Fairness

In this section, we present the design of the Net-P4ct bandwidth management framework, which provides guaranteed bandwidth with fairness across services. We begin by introducing the abstraction of the service bandwidth request model and the procedure by which service teams submit bandwidth requests. These requests are then evaluated by the network team and either accepted or rejected based on current resource availability. We then describe our approach to bandwidth sharing among multiple services, followed by a discussion of the enforcement mechanisms that ensure proper bandwidth allocation and traffic prioritization.

4.1 Bandwidth Request Model Abstraction

We define a *job* as the minimal unit of bandwidth request, represented as a tuple: {*JobID*, *Service*, *Pattern*, *RegionA*, *RegionZ*, *GuaranteedBandwidth*, *Weight* }.

Each *job* corresponds to a collection of flows from a source region *RegionA* to a destination region *RegionZ*. The *Service* field identifies the business service to which the *job* belongs. A single business service typically consists of multiple jobs, each with distinct flow patterns or regions.

The *Pattern* field describes the traffic characteristics of the *job*. Based on traffic category, we classify patterns into two categories.

Host-to-host pattern. In our containerized infrastructure, multiple services share the same computing resources. To distinguish traffic at the job level, packets originating from hosts are required to carry a unique *job* identifier. This identifier can be recognized by P4 switches and used to associate the traffic with the job. Currently, IPv6 accounts for the ma-

jority of our internal traffic. For IPv6 packets, we embed the service job ID in the Destination Options Header (DOH), as shown in Figure 5. For IPv4 packets, the ID is carried in the IP Options field.

External-to-host pattern. In this communication model, source IP or the destination IP prefix corresponds to a registered job. For instance, public IP address of a cloud load balancer is used in a service job. We identify such traffic using a 3-tuple pattern: {*srcIPPrefix*, *dstIPPrefix*, *proto*}.

The *GuaranteedBandwidth* field specifies the requested bandwidth for flows from *RegionA* to *RegionZ*. This value represents a strict guarantee, and the network team ensures the bandwidth requirement is met. The *Weight* field indicates the priority level of the *job*. A higher weight indicates a higher priority in scheduling and resource allocation.

Based on our operational experience, service teams often overestimate their bandwidth needs, leading to underutilization and inefficient resource allocation. To address this, we require service teams to enter a bandwidth agreement with the network team. Under this agreement, if the usage is far less than the requested, the service team is responsible for a fixed portion of the cost associated with their requested bandwidth.

Meanwhile, the network team collaborates with service teams on a monthly basis to review and adjust bandwidth requests based on historical traffic trends, ensuring that allocations remain aligned with real-world demand.

4.2 Bandwidth Request Grant

After a bandwidth request is submitted, the approval system determines whether the request can be granted based on current resource availability. This decision depends on the capacity of underlying network. We first introduce the link capacity model, then describe how external systems (i.e., Traffic Engineering (TE) system and inventory system) coordinate to approve and allocate the guaranteed bandwidth.

4.2.1 Link Capacity Model

To ensure fault tolerance, ByteDance's backbone network is designed with built-in redundancy. A portion of the network's total link capacity is intentionally reserved as headroom to absorb failures and support disaster recovery scenarios.

Each link's capacity is divided into two classes:

- Reserved capacity: Used for high-priority traffic and guaranteed bandwidth allocations.
- Non-reserved capacity: Available for best-effort traffic and non-critical services.

For example, consider a scenario where four 100G links connect Region A and Region Z, providing a total capacity of 400 Gbps. To maintain resilience, 60% of the total capacity (240 Gbps) is reserved. The remaining 160 Gbps is classified as non-reserved capacity. When a service team requests guaranteed bandwidth, it is always allocated from the reserved capacity pool.

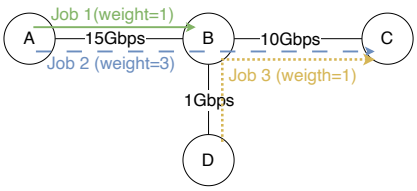


Figure 6: Bandwidth fair sharing example

4.2.2 Guaranteed Bandwidth Allocation

Bandwidth requests are evaluated through the coordination of the Traffic Engineering (TE) system and the inventory system.

The inventory system maintains an up-to-date view of the network topology by collecting information via standard protocols such as SNMP, BGP-LS, and BMP [12, 17, 27]. When a new job request is submitted, the TE system computes a set of candidate paths that satisfy the connectivity constraint. These candidate paths are then validated by the inventory system, which checks whether each link along the path has sufficient reserved capacity to fulfill the requested guaranteed bandwidth. If a feasible path is found, the bandwidth is reserved, and the request is approved. When resources are limited, the system prioritizes critical services to ensure that available capacity is allocated efficiently.

4.3 Bandwidth Sharing Policies

In Net-P4ct, service jobs share available bandwidth in a work-conserving manner: one job can use spare bandwidth beyond its minimum guarantee when the network is underutilized.

We define the concept of fair share bandwidth, allocated from the non-reserved capacity (the portion of link capacity not reserved for guaranteed traffic). While guaranteed bandwidth specifies the minimum service request, fair share bandwidth reflects the additional bandwidth a job may receive, based on its priority weight.

Net-P4ct aims to maximize overall network throughput using a weighted max-min fairness model [11, 26, 28]. We calculate fair shares using the water-filling algorithm, as outlined in Algorithm 1 in. Figure 6 presents an example of bandwidth allocation among three service jobs with weights 1, 3, and 1. The non-reserved capacities of the three links are 15Gbps, 10Gbps, and 1Gbps, respectively. We denote the minimum fair share unit as b. Bandwidth is allocated iteratively as follows.

- Round 1: Allocate initial shares as $B_1 = b, B_2 = 3b, B_3 = b$. increase b uniformly until link $l_{D->B}$ becomes saturated. This link is only used by Job 3, so its fair share is finalized as $B_3 = 1Gbps$.

Algorithm 1 Weighted max-min fairness allocation

Input:
$\mathcal{S}$: set of all services
$\mathcal{L}$: set of all links
W_i: weight of service $i \in \mathcal{S}$
C_l: non-reserved capacity of link $l \in \mathcal{L}$
$P(i,l) = \mathbf{1}_{\{\text{service } i \text{ traverses link } l\}}$ is an indicator function
Output:
B_i: fair share bandwidth of service $i, \forall i \in \mathcal{S}$
Init:
$\mathcal{S}_{\text{active}} \leftarrow \mathcal{S}$, $\mathcal{L}_{\text{active}} \leftarrow \mathcal{L}$, $\mathcal{L}_{\text{sat}} \leftarrow \emptyset$, $b \leftarrow max\_int$.
Start:
while $\mathcal{L}_{\text{active}} \neq \emptyset$ **do**
Step 1: calculate b as follows.

$$\max_{b} \quad b \tag{1a}$$

$$\text{s.t.} \quad \sum_{i \in \mathcal{S}_{\text{active}}} bW_iP(i,l) \leq C_l, \quad \forall l \in \mathcal{L}_{\text{active}} \tag{1b}$$

Step 2: find saturated links and services on these links.
$\mathcal{L}_{\text{sat}} \leftarrow \mathcal{L}_{\text{sat}} \cup \{l|$ equality is met in (1b)$\}$
$\hat{\mathcal{S}} \leftarrow \{i|P(i,l) = 1, \forall l \in \mathcal{L}_{\text{sat}}\}$
output $B_i \leftarrow b \cdot W_i, \forall i \in \hat{\mathcal{S}}$
Step 3: update corresponding variables.
$C_l \leftarrow C_l - \sum_{i \in \hat{\mathcal{S}}} b \cdot W_i, \forall l \in \mathcal{L}_{\text{active}}$
$\mathcal{L}_{\text{active}} \leftarrow \mathcal{L}_{\text{active}} \setminus \mathcal{L}_{\text{sat}}$
$\mathcal{S}_{\text{active}} \leftarrow \mathcal{S}_{\text{active}} \setminus \hat{\mathcal{S}}$
end while

- Round 2: Continue increasing b for Job 1 and Job 2. Stop when link $l_{B->C}$ becomes saturated. This link is shared by Job 2, resulting in $B_2 = 3 * 3 = 9Gbps$.
- Round 3: Finally, continue increasing b for Job 1 until link $l_{A->B}$ becomes saturated. This gives $B_1 = 6Gbps$.

Thus, the final fair share bandwidths are: 6Gbps, 9Gbps and 1Gbps for Job 1, Job 2 and Job 3, respectively.

4.4 Enforcement Policies

Our operational experience with the legacy system revealed that traditional rate limiting mechanisms often introduce throughput fluctuations. This instability occurs due to threshold-based enforcement: rate limiting is activated when link utilization exceeds an upper threshold and is released only after it drops below a lower threshold. This behavior frequently results in repetitive oscillations in throughput, reducing overall bandwidth utilization efficiency.

Figure 7 illustrates this phenomenon in our production network. Rate limiting was triggered at time t_1 when link utilization reached 80% of capacity. It remained active until t_2, when throughput fell below 65%. A similar cycle occurred from t_3 to t_4.

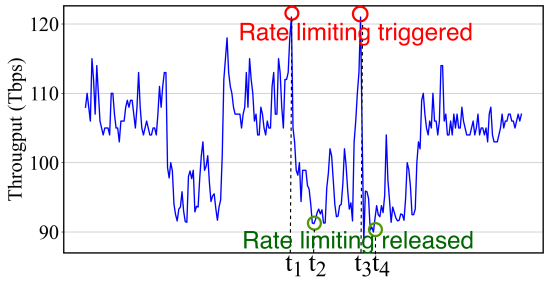


Figure 7: Throughput oscillations in legacy system

To address these limitations, Net-P4ct replaces rate limiting with DSCP remarking performed by P4 switches. Downstream WAN switches enforce QoS policies based on DSCP values using strict priority queuing. Packets are classified into three queues: B (high priority), C (medium priority), and D (low priority). The QoS class for each packet is determined by one of two enforcement policies: a meter-based policy (MBP) or a statistics-based policy (SBP).

4.4.1 Meter-based Policy

As shown in Figure 8, MBP applies a Two-Rate Three-Color Marker (trTCM) for each service job to classify packets into three priority levels: green, yellow, or red [18]. For simplicity, this mechanism can be conceptually modeled as a cascade of two token buckets. Every packet sequentially passes through both buckets. Each trTCM meter is configured with four parameters: Committed Information Rate (CIR), Committed Burst Size (CBS), Peak Information Rate (PIR), and Peak Burst Size (PBS). For clarity, we focus on the rate-based behavior and omit burst size considerations:

- If the traffic rate is within the CIR, packets are marked green.
- If the traffic rate exceeds the CIR but remains within the PIR, packets are marked yellow.
- If the traffic rate exceeds PIR, packets are marked red.

In our design, CIR is set to the job's guaranteed bandwidth, while PIR is set to the sum of guaranteed bandwidth and fair share bandwidth. Packets are then remarked with DSCP values and mapped to switch queues as follows.

- Green packets → High-priority QoS class
- Yellow packets → Medium-priority QoS class
- Red packets → Low-priority QoS class

This policy enforces strict priority while enabling efficient bandwidth utilization across services.

4.4.2 Statistics-based Policy

While the MBP is intuitive, its per-packet operation can cause packet reordering at the destination. To address this, we introduce the Statistics-based Policy (SBP), which operates at

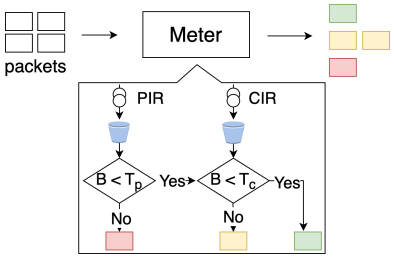


Figure 8: The working mechanism of meter-based policy

the flow group level. It works as follows: Traffic flows are mapped into a set of predefined hash buckets based on their hash values, with each bucket representing a flow group. A feedback mechanism monitors the real-time throughput of each bucket and updates its DSCP value periodically. As shown in Figure 9, SBP consists of two components: manual coloring and burst mitigation.

Manual Coloring. For each job, we allocate N hash buckets, assigning flows to buckets using a 5-tuple hash (based on source IP, destination IP, source port, destination port, and protocol). Each bucket tracks counters of incoming packets. Let t denote the previous time window, and $r^t_{b_i}$ represent the observed rate for bucket i during window t.

At time $t+1$, we select a subset of buckets such that the sum of their rates meets or just exceeds the guaranteed bandwidth, i.e., find

$$\min \sum r^t_{b_i} \quad \text{s.t.} \quad \sum r^t_{b_i} \geq GuaranteedBandwidth.$$

This process can be viewed as a variant of the knapsack problem [35]. Packets belonging to the selected buckets are marked green and assigned the corresponding high-priority DSCP values.

We then repeat this process on the remaining buckets to satisfy the fair share bandwidth, i.e.,

$$\min \sum r^t_{b_i} \quad \text{s.t.} \quad \sum r^t_{b_i} \geq FairShareBandwidth,$$

and mark packets from these buckets as yellow.

Any buckets not selected in either step are marked red. This prioritization is updated periodically based on statistics from the previous cycle.

Burst Mitigation. While manual coloring effectively eliminates packet reordering and improves traffic stability, it cannot respond quickly to short-term traffic bursts. To address this, we deploy two additional meters-one for green packets and one for yellow packets, to enable fine-grained burst detection and prevent potential overload.

For green packets, the meter is configured with both CIR and PIR set to the guaranteed bandwidth. Thus, traffic within the guaranteed rate remains green, while any excess is downgraded to red. For yellow packets, the meter is similarly configured, with both CIR and PIR set to the fair share bandwidth. Traffic within this threshold remains yellow, and any excess is marked red.

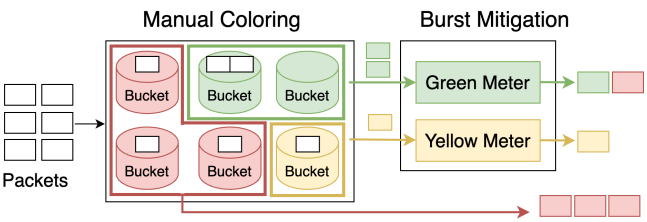


Figure 9: The workflow of statistics-based policy

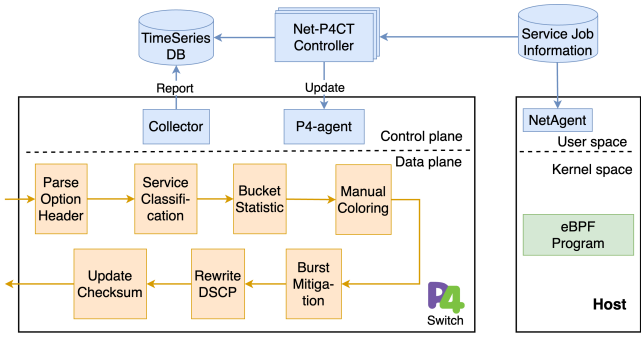


Figure 10: The architecture of Net-P4ct

This layered enforcement provides a safety net for burst handling, complementing the manual coloring with real-time rate awareness.

5 System Implementation

Net-P4ct is deployed in production networks, with each WAN ingress point equipped with a dedicated cluster of Tofino-based P4 switches [4]. As shown in Figure 10, the system architecture consists of three primary components: the host agent, the P4 pipeline, and the central controller. In the primary workflow, a service request triggers the controller to calculate the bandwidth allocation. This result is sent to the P4-agent, which then programs the policy into the P4 pipelines. Concurrently, the collector monitors system health and data plane statistics (e.g., flow counters), writing this telemetry data to a time-series database. For H2H scenarios, a host agent handles requests by configuring eBPF programs directly on the host.

5.1 Host Agent

On each host, Net-P4ct deploys two components: a user-space agent (NetAgent) and a kernel-space eBPF program that operates as part of the host's data plane.

NetAgent periodically queries a centralized service to obtain metadata including the mappings of $\{host, jobID, cgroupID\}$, enabling the association of containers with their corresponding service jobs. Once retrieved, the agent updates a BPF map with entries containing $jobID$ and $cgroupID$. The eBPF program consults this map to identify traffic originating from containers and tags outgoing

packets with the appropriate job ID before they leave the host.

This per-packet service tagging allows downstream P4 switches to classify traffic by job and apply appropriate QoS policies in the WAN.

5.2 P4 Pipeline

Net-P4ct's P4 pipeline is designed to perform high-speed classification and enforcement of bandwidth policies for all service traffic. It supports both IPv4 and IPv6, including packets with optional headers.

Upon entering the pipeline, each packet is classified at the service job level. For host-originated traffic, this is done by extracting the *jobID* embedded in the packet by the host agent. For traffic arriving from external sources, such as public cloud or Internet ingress, classification is performed using a 3-tuple match on source IP prefix, destination IP prefix, and protocol.

For the MBP, each packet traverses a trTCM meter. Meter parameters (e.g., CIR and PIR) and DSCP values are provisioned by the P4-agent in the control plane.

For the SBP, the pipeline uses a 5-tuple hash to assign flows to hash buckets. Each bucket maintains hardware counters to track traffic volume in bytes. These statistics are periodically collected by a control-plane collector within the P4 switch. The data is finally sent to a central time-series database.

Net-P4ct implements a two-stage coloring pipeline for SBP: 1) Manual Coloring. The central controller assigns a static color to all packets within a bucket, based on the collected traffic statistics. 2) Meter-based Coloring. After manual coloring, packets undergo additional rate-based coloring using trTCM. Separate meters are applied to green and yellow packets to enforce bandwidth ceilings. This ensures that any traffic exceeding guaranteed or fair share limits is downgraded to red for precise burst control.

This approach allows Net-P4ct to provide both stable assignment (via manual coloring) and dynamic enforcement (via trTCM) within the same pipeline. Based on the final color, the P4 switch updates the DSCP value in the IP header. For IPv4 traffic, the header checksum is recalculated to maintain protocol correctness.

5.3 Central Controller

The central controller orchestrates the entire Net-P4ct system. It is responsible for:

- Managing the service job lifecycle, including registration, deregistration, and bandwidth adjustments
- Calculating fair share bandwidth allocations based on user input and data from time-series database
- Generating updated enforcement policies and distributing them to P4-agent

Implemented as a cloud-native, stateless application, the central controller stores all persistent data in a distributed database. Historical traffic metrics, such as per-bucket counters, are maintained in a time-series database. The controller leverages this data to compute updated enforcement policies using the max-min fairness algorithm. Policy updates are then pushed to the P4-agent running on the P4 switches.

6 Evaluation

This section shows extensive evaluation results of our proposed design alternatives in our testbed.

6.1 Setup

Testbed Settings: We implemented the Net-P4ct P4 program and deployed it on a commodity Wedge-100BF-65X switch with a Barefoot Tofino 6.4 Tbps ASIC, supporting up to 64×100 Gbps ports. The Tofino switch has 4 pipes, and each pipe is composed of an ingress pipeline and an egress pipeline. In order to assign more available resources (e.g., stages and memory) for P4 program, the code was implemented in folded mode, that is, each ingress pipeline at the first pipe is concatenated to other three pipes in series and looped back to the egress pipeline of the first pipe. The switch supports 16 external accesses in this mode. For each service job, we allocate 32 hash buckets for fine-grained traffic partitioning, with each bucket maintaining a 64-bit counter to track byte statistics for rate estimation. The collector reports the data every 10s. The central controller calculates enforcement policy updates every 60s.

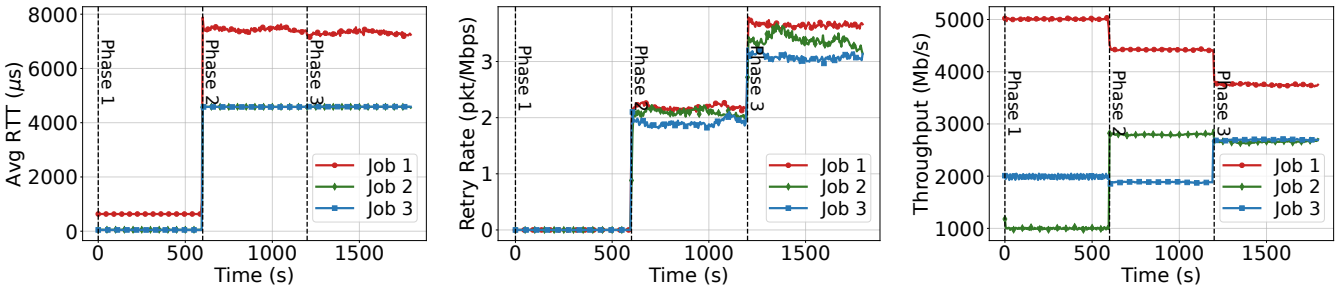
Our testbed includes 8 servers (Intel(R) Xeon(R) CPU Platinum 8260 @ 2.40GHz, 96 Cores, 1536 GB RAM), each equipped with 2×25 Gbps Ethernet interfaces. NetAgent is installed on each server. The Net-P4ct controller is deployed as 3 distributed replicas, which communicate with the P4 switches via gRPC.

Experiment Scenario: The testbed resembles a simple WAN with multiple Points of Presence (PoPs) that connect to PE switches. The focus of the experiment is on inter-PoP traffic management and bandwidth enforcement. In addition to the network inter-connectivity, each PoP is equipped with servers and P4 devices that handle essential tasks such as data processing and traffic marking, ensuring effective bandwidth management and resource sharing.

Table 1: Bandwidth Agreement for Services

Job (Service)	weight	minBW	fairShareBW
1	1	1	0.5
2	3	2	1.5
3	4	3	2

We deploy 3 jobs on separate PoPs (i.e., hosts) to simulate the traffic generation of 3 different services. Each job is as-



(a) RTT (b) Retransmission (c) Throughput

Figure 11: Baseline: no bandwidth enforcement.

Table 2: Time-varying Traffic Patterns

Phase	Time Interval (min)	Rate (Gbps)			
		Job 1	Job 2	Job 3	Total
Phase 1	0–10	5	1	2	8
Phase 2	10–20	5	3	2	10
Phase 3	20–30	5	3	3	11

signed a network path, with all paths converging on a shared 100 Gbps link. To emulate congestion, we configure access control rules (ACLs) on the network switches to drop packets once aggregate throughput exceeds 10 Gbps. The reserved capacity of the link is set to 6 Gbps and the non-reserved capacity is 4 Gbps.

The bandwidth requests for these services are shown in Table 1, with fair bandwidth values obtained from Algorithm 1. We use Job 1 to simulate services that are granted less weight and guaranteed bandwidth compared to Job 2 and 3.

Table 2 shows the time-varying traffic patterns of services. As we can see, starting from phase 2, the total transmission rate of all jobs would reach 10 Gbps, triggering congestion and packet drops.

6.2 Numerical Results

Baseline: We first show the performance of a no-action policy as a baseline. As shown in Figure 11(a), without bandwidth enforcement, round trip times (RTTs) of all jobs increase in phase 2 due to network congestion. Figure 11(b) shows that congestion also increases normalized retransmission rates (defined as the ratio between retransmission packet counts and its rate) for all jobs after phase 2. The retransmission rate in phase 3 is further increased due to more sever congestion. In Figure 11(c), as the total transmission rate increases in phase 2, each job roughly gets throughput as listed in Table 2. As Job 3 keeps increasing its sending rate in phase 3, it obtains more bandwidth at the cost of throughput losses for Job 1 and Job 2.

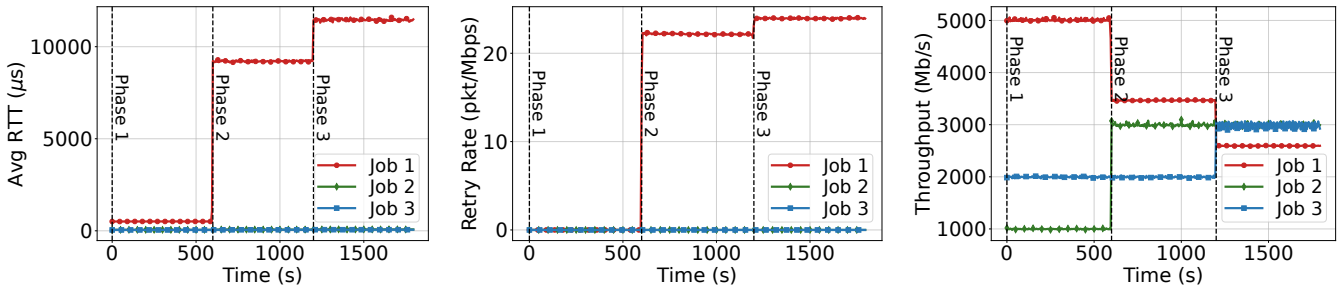
Meter-based Policy: Our first observation is that MBP ensures the guaranteed bandwidths for all jobs at all time. In phase 1, all jobs reach the throughput of their transmission rates, as no congestion is present in the network. However, as congestion occurs in phase 2, Job 1's bandwidth share is immediately suppressed, leading to increasing RTT and retransmission in Figure 12(a) and 12(b). In contrast, Job 2 and 3 consistently maintain negligible RTTs and zero retransmission counts. Figure 12(c) shows that throughputs of Job 2 and 3 always remain above their guaranteed min_BW shares (2 Gbps and 3 Gbps, respectively). In phase 3, since Job 2 and 3 do not fully use their fair-shared amount of bandwidth (3.5 Gbps and 5 Gbps, respectively), Job 1 is able to obtain approximately 2.6 Gbps bandwidth, which is higher than its fair-shared amount of 1.5 Gbps.

Statistics-based Policy: Figure 13 shows the performance of SBP. Similar to MBP, the demands of all jobs are satisfied, resulting in their average RTT and retransmission rate remaining at 0 in phase 1. However, in Figure 13(c), Job 1 obtains a higher throughput (4.1 Gbps) than the case under MBP (3.5 Gbps) in phase 2, while the rates of Job 2 and 3 stay the same. The similar trend is observed in phase 3. The results reflect that SBP significantly improves Job 1' throughput during congestion periods. Consequently, by comparing Figures 12(b) and 13(b), we observed that SBP has better performance in the retransmission count compared to MBP.

Remark. *Note that, as we discussed in Section 4.4, MBP operates at the individual packet level, while SBP assigns same priority for all packets in the same flow. As a result, even though only a fraction of packets are assigned a low priority, MBP tends to affect more flows than SBP, causing higher retransmission rate.*

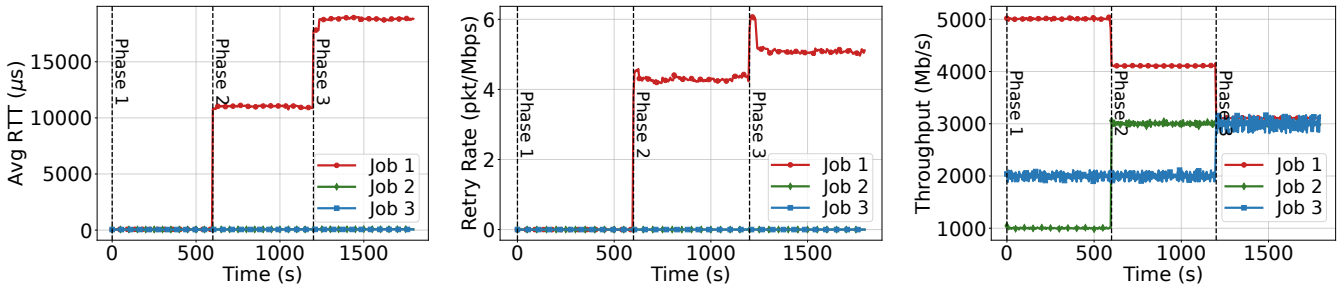
RTT & TCP Efficiency Comparison: We also analyze the cumulative distribution of RTT and retransmission rates for Job 1 under both MBP and SBP. Figure 14 shows that MBP presents a typical log-normal RTT distribution in phase 2 and phase 3. This is because MBP operates at packet level, causing each flow in Job 1 to experience packet drops with equal probability. In contrast, MBP displays a dispersed latency pattern characterized by a dichotomy: approximately 30% of flows maintain near-zero RTTs, whereas the remaining flows experience significantly higher RTTs. Figure 15 further illustrates that SBP outperforms MBP in terms of the retransmission rate. SBP also follows a log-normal distribution, with retry rates consistently higher than those with MBP.

Total Throughputs: Figure 16 compares the overall throughput with MBP and SBP. During congestion phases, SBP con-



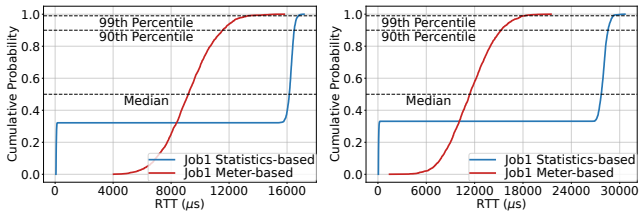
(a) RTT (b) Retransmission (c) Throughput

Figure 12: Meter-based policy



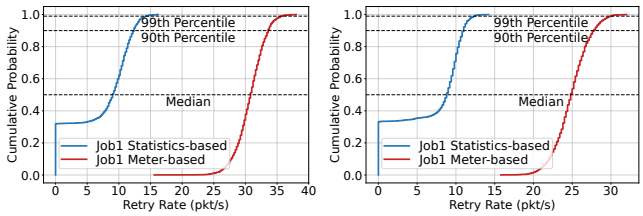
(a) RTT (b) Retransmission (c) Throughput

Figure 13: Statistics-based policy



(a) RTT CDF of Phase 2 (b) RTT CDF of Phase 3

Figure 14: RTT CDF comparison



(a) Retry Rate CDF of Phase 2 (b) Retry Rate CDF of Phase 3

Figure 15: Retransmission CDF comparison

sistently delivers significantly higher throughput than MBP. This improvement is because that a subset of Job 1's flows are not degraded (remarked as red) during congestion with SBP. In contrast, MBP tends to affect more flows of Job 1 as MBP is flow-agnostic. This results in a lower data transfer efficiency. These findings indicate that SBP is more effective at utilizing available bandwidth under congested conditions.

Flow Completion Time: Figure 17 shows the flow completion time (FCT) comparison between MBP and SBP with 50GB data transfer. The FCT of MBP is higher than that of SBP when congestion occurs (phase 2 and phase 3). This is because Job 1 under SBP can fully utilize its requested and fair share bandwidth without retransmission.

Host Agent Throttling CPU Overhead: We also evaluate the CPU usage of NetAgent in two modes: traffic tagging only (in Net-P4ct) and host-based throttling (in our legacy system). The total traffic rate is fixed at 25 Gbps. Figure 18 shows that as the number of flows increases from 2,000 to 16,000, the CPU usage for both modes increases. While Net-P4ct remains under 3% CPU usage even at 16,000 flows, the host-based throttling grows sharply, reaching nearly 15% CPU usage at the same point. This demonstrates Net-P4ct avoids the additional processing overhead incurred by in-host throttling, resulting in improved cost efficiency.

Mircoburst Impact: We evaluate SBP's effectiveness in mitigating the impact of microbursts. The bandwidth assignment for each job is consistent with that of "Phase 1" in Table 2. In this experiment, we introduce three 5-second traffic bursts starting at 30s, 60s, and 90s. The SBP policy allocation remains static during the test. As shown in Figure 19, the initial burst from Job 1 does not affect Jobs 2 and 3. This is because SBP's burst mitigation mechanism demotes it to a low priority by remarking its DSCP value. When Job 2 initiates its burst, we observe a drop in Job 1's throughput and a spike in its retransmissions. This occurs because Job 1 was consuming bandwidth beyond its fair share. We see a similar pattern when Job 3 bursts.

We also evaluated the resource consumption, as described in Appendix B.

7 Real World Production Deployment

Over the past year, we have deployed Net-P4ct across 4 data centers and 24 WAN PoPs. The initial adopters were business

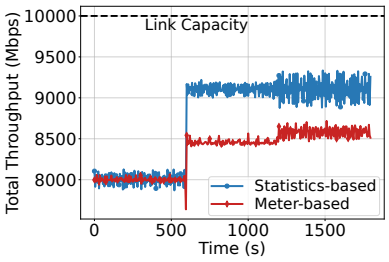


Figure 16: Overall throughput

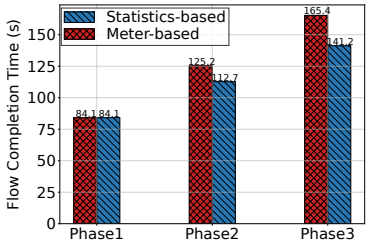


Figure 17: FCT comparison

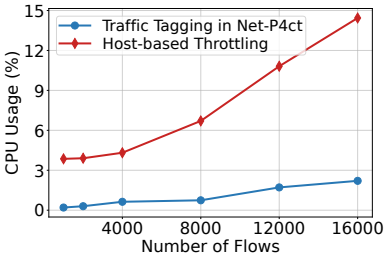
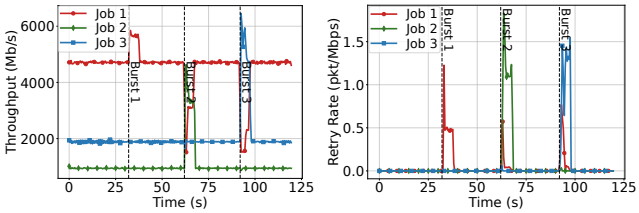


Figure 18: CPU overhead



(a) Throughpupt (b) Retransmisson

Figure 19: Microburst scenario with SBP

services with external communication requirements, including cross-region traffic and cloud workloads. Figure 20(a) illustrates the growth of jobs and deployment over time. More than 500 service jobs have been registered by service teams. As the system has been proven stable in production, we are accelerating the rollout to the rest of the data centers and PoPs.

To support large-scale configuration updates, we run 9 distributed controller replicas using 36 CPU cores. Figure 20(b) presents the configuration time of each iteration over a typical time series. The 99th percentile (p99) latency is 7.37 seconds from policy generation to enforcement.

Net-P4ct has significantly improved traffic control and bandwidth predictability. Prior to the deployment of Net-P4ct, X2H traffic from certain services could exceed requested bandwidth by over 5× due to the lack of effective enforcement, as shown in the upper plot of Figure 20(c). In the current production environment, most services now operate close to their requested bandwidth. Even for aggressive bandwidth users, their peak usage does not exceed 150%, as shown in the lower plot of Figure 20(c). This improvement results from in-network enforcement and monthly bandwidth reviews between network and service teams, enabling more accurate capacity planning and efficient bandwidth allocation.

Interestingly, deploying Net-P4ct also improves our traffic visibility. Previously, we relied on coarse-grained SNMP counters to monitor throughput on a per-port basis. In contrast, Net-P4ct enables us to perform fine-grained traffic analysis at the flow level (e.g., by IP address). It has substantially deepened our understanding of the overall network traffic distribution.

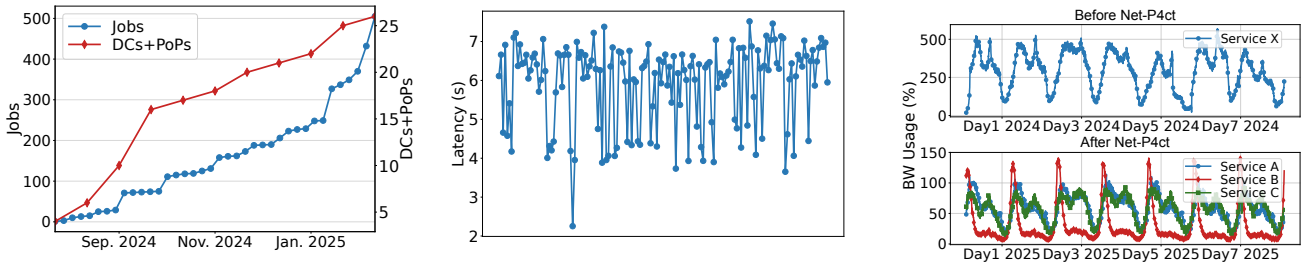
Deployment Experiences and Lessons Learned

Migrating to Net-P4ct. To ensure stability during the migration, we adopt a phased rollout strategy. In the initial canary phase, we choose to migrate low-priority services with small traffic volumes on a per-subnet basis. The process for migrating a single subnet is as follows: 1) Identify the target subnet for migration; 2) Program the P4 switches for each region via the control plane with the necessary flow rules to enable packet forwarding for the target subnet; 3) After the data plane is ready, we redirect the service traffic to the P4 switches by having them announce BGP routes for the service subnet. We then gradually expand the migration scope to other services.

Cluster Scaling-out. Our system employs a cluster of P4 switches to provide both high availability and load balancing. Incoming traffic is distributed across P4 switches within the cluster via ECMP. When a service's traffic is below a predefined threshold (empirically set to 1 Gbps), we encapsulate the packets into a VXLAN tunnel and steer them to a single, designated P4 switch. The enforcement policy is applied on this single switch. When a service's traffic exceeds the threshold, we distribute the policy enforcement across all switches to leverage the aggregate processing power of the cluster. For example, an 8 Gbps policy for a service would be implemented by enforcing a 2 Gbps on each of the four P4 switches in the cluster. This adaptive strategy allows us to balance the trade-off between simplified state management and the horizontal scalability of a distributed architecture.

Failure detection and fallback is critical. Since P4 plays an important role in Net-P4ct, to distinguish P4 failures from the physical network failures, we deploy active probes that monitor both P4 and non-P4 paths simultaneously. This allows us to quickly identify the source of failures. In the event of a P4 switch port failure or an entire P4 switch failure, the BGP routes are automatically withdrawn. The traffic is then seamlessly redirected to the other active links/devices in the cluster. In the highly unlikely event of a catastrophic P4 cluster failure, we have implemented a fail-safe mechanism to ensure basic network connectivity. This mechanism allows us to bypass the entire P4 cluster by falling back to the underlay physical network.

Bandwidth request reviews are essential. In a fast-paced environment where new services are launched frequently, oftentimes their initial traffic patterns are unclear to network operators. It is crucial for the network team to collaborate closely with service teams to continuously review and adjust bandwidth requests, especially during early deployment. This process helps the network team track the actual demand of



(a) Service Job Count

(b) Configuration Latency

(c) Service bandwidth usage (rate/requestedBW)

Figure 20: Production deployment of Net-P4ct

services and improve bandwidth utilization.

Deploying Net-P4ct outside ByteDance. The core concept of Net-P4ct is a general in-network bandwidth sharing mechanism, which we believe has broad applicability beyond our internal corporate environment. A critical factor is that Net-P4ct has no specific requirements for Tofino switches. Instead, it leverages common programmable ASIC capabilities such as flexible packet parsing/LPM lookup. Given Intel's 2024 announcement to discontinue its Tofino product line, we are actively exploring the feasibility of porting Net-P4ct to other prominent networking silicon, such as Trident5 [2, 3].

8 Related Work

There has been an extensive body of work on WAN bandwidth management, both in academia and in industry. On the high level, we can broadly categorize previous studies into two directions, i.e., throttling-based QoS management and SDN-TE based approaches.

The most related body of work is in the line of throttling-based QoS management. BwE [25] is a hierarchical bandwidth allocation system for WAN distributed computing. The service traffic is aggregated by "enforcers" at different levels, and a global enforcer is responsible for computing a network-wide bandwidth allocation decision, which will be passed down again to each level's enforcer. Ultimately, the throttling is done at the host side. BwE incorporates bandwidth functions to establish fair sharing among services. In contrast, Net-P4ct utilizes in-network enforcement to provide better coverage.

Network Entitlement [9] is the contract-based network sharing solution for Meta's backbone network. By remarking non-conforming traffic of misbehaving services to a lower priority, it protects conforming traffic and realizes service isolation. For conforming traffic, the network guarantees SLOs such as availability, clarifying the responsibilities of service teams and network teams, and facilitating accountability. However, [9] does not account for fairness among services, which is in direct contrast to Net-P4ct.

Algorithms for distributed limiters are studied in [33] to recreate the flow behavior as if it is under a single centralized rate limiter. The flow proportional share (FPS) limiter proposed in [33] is appropriate for deployment in TCP-based Web-services environments, while Net-P4ct's rate limiting is protocol agnostic, i.e., applicable to TCP and UDP.

On the other line of work, recent industrial experience including SWAN [19, 24], B4 [20, 37], and EBB [15] provide valuable insights for WAN traffic management in the production scale. However, the main focuses of above systems are on developing Software Defined Network (SDN) controllers for traffic engineering purposes. Meanwhile, Net-P4ct manages the service traffic entering the WAN, and promotes fair resource sharing. Net-P4ct also work together with our in-house TE controller for path allocation and inventory approval.

9 Conclusion

In this paper, we present Net-P4ct, a highly customizable and versatile system for allocating bandwidth and fairly sharing resources for WAN traffic. By leveraging P4 programmable switches, Net-P4ct overcomes the limitations of traditional host-based enforcement systems and improves bandwidth utilization, ensuring that traffic from different services with varying SLO requirements is efficiently managed. Our system not only improves the usage of available bandwidth, but also enhances the overall performance of WAN. Compared to kernel-dependent host-side solutions, reduced operational complexity simplifies network management and maintenance, saving both time and resources. Additionally, the lower per-byte processing cost makes it a cost-effective solution for large-scale WAN deployments. With its successful deployment in ByteDance's production WAN for nearly a year, Net-P4ct has proven its performance and reliability in a high-speed real-world network environment. It provides a reliable and effective approach for dynamic QoS control, enabling better service classification and more fair bandwidth allocation. As internet companies continue to grow and face increasing demands on their WAN infrastructure, Net-P4ct offers a promising model for future network management, setting a new standard for in-network bandwidth enforcement.

Acknowledgement

We sincerely thank the shepherd Umesh Krishnaswamy and all the anonymous reviewers for their valuable comments.

References

[1] Fair queue traffic policing. https://man7.org/linux/man-pages/man8/tc-fq.8.html.

[2] High-Capacity StrataXGS® Trident 5 Programmable Ethernet Switch Series. https://www.broadcom.com/products/ethernet-connectivity/switching/strataxgs/bcm78800.

[3] Intel Tofino products pcn-827577-00 product discontinuance tofino end-of-life. https://www.intel.com/content/www/us/en/content-details/827577.html.

[4] Intel® Tofino™. https://www.intel.com/content/www/us/en/products/details/network-io/intelligent-fabric-processors/tofino.html.

[5] ipvs: clear skb->tstamp in forwarding path. https://patchwork.kernel.org/project/netdevbpf/patch/20201013234559.15113-3-pablo@netfilter.org/.

[6] [net-next] tcp/fq: move back to CLOCK_MONOTONIC. https://patchwork.ozlabs.org/project/netdev/patch/20180928172844.182542-1-edumazet@google.com/.

[7] net_sched: sch_fq: add horizon attribute. https://git.kernel.org/pub/scm/linux/kernel/git/torvalds/linux.git/commit/?id=39d010504e6b4485d7ceee167743620dd33f4417.

[8] TCP: switch to Early Departure Time model. https://lore.kernel.org/netdev/20180921155154.49489-1-edumazet@google.com/.

[9] Satyajeet Singh Ahuja, Vinayak Dangui, Kirtesh Patil, Manikandan Somasundaram, Varun Gupta, Mario Sanchez, Guanqing Yan, Max Noormohammadpour, Alaleh Razmjoo, Grace Smith, et al. Network Entitlement: Contract-based Network Sharing with Agility and SLO Guarantees. In *Proceedings of the ACM SIGCOMM 2022 Conference*, pages 250–263, 2022.

[10] Pat Bosshart, Dan Daly, Glen Gibb, Martin Izzard, Nick McKeown, Jennifer Rexford, Cole Schlesinger, Dan Talayco, Amin Vahdat, George Varghese, and David Walker. P4: Programming Protocol-Independent Packet Processors. *SIGCOMM Comput. Commun. Rev.*, 44(3):87–95, July 2014.

[11] Zhiruo Cao and Ellen W Zegura. Utility Max-Min: An Application-Oriented Bandwidth Allocation Scheme. In *IEEE INFOCOM'99. Conference on Computer Communications. Proceedings. Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies. The Future is Now (Cat. No. 99CH36320)*, volume 2, pages 793–801. IEEE, 1999.

[12] Jeffrey D Case, Mark Fedor, Martin Lee Schoffstall, and James Davin. RFC1157: Simple Network Management Protocol (SNMP), 1990.

[13] Roger Chen and Rui Ma. How ByteDance Became the World's Most Valuable Startup, Feb 2022.

[14] Tianyu Chen, Yiheng Lin, Nicolas Christianson, Zahaib Akhtar, Sharath Dharmaji, Mohammad Hajiesmaili, Adam Wierman, and Ramesh K. Sitaraman. SODA: An Adaptive Bitrate Controller for Consistent High-Quality Video Streaming. In *Proceedings of the ACM SIGCOMM 2024 Conference*, ACM SIGCOMM '24, page 613–644, New York, NY, USA, 2024. Association for Computing Machinery.

[15] Marek Denis, Yuanjun Yao, Ashley Hatch, Qin Zhang, Chiun Lin Lim, Shuqiang Zhang, Kyle Sugrue, Henry Kwok, Mikel Jimenez Fernandez, Petr Lapukhov, Sandeep Hebbani, Gaya Nagarajan, Omar Baldonado, Lixin Gao, and Ying Zhang. EBB: Reliable and Evolvable Express Backbone Network in Meta. In *Proceedings of the ACM SIGCOMM 2023 Conference*, ACM SIGCOMM '23, page 346–359, New York, NY, USA, 2023. Association for Computing Machinery.

[16] Anja Feldmann, Oliver Gasser, Franziska Lichtblau, Enric Pujol, Ingmar Poese, Christoph Dietzel, Daniel Wagner, Matthias Wichtlhuber, Juan Tapiador, Narseo Vallina-Rodriguez, Oliver Hohlfeld, and Georgios Smaragdakis. A year in Lockdown: How the Waves of COVID-19 Impact Internet Traffic. *Commun. ACM*, 64(7):101–108, June 2021.

[17] R Fernando and S Stuart. RFC 7854: BGP Monitoring Protocol (BMP), 2016.

[18] Juha Heinanen and Roch Guerin. RFC2698: A Two Rate Three Color Marker, 1999.

[19] Chi-Yao Hong, Srikanth Kandula, Ratul Mahajan, Ming Zhang, Vijay Gill, Mohan Nanduri, and Roger Wattenhofer. Achieving High Utilization with Software-Driven WAN. In *Proceedings of the ACM SIGCOMM 2013 Conference on SIGCOMM*, pages 15–26, 2013.

[20] Sushant Jain, Alok Kumar, Subhasree Mandal, Joon Ong, Leon Poutievski, Arjun Singh, Subbaiah Venkata, Jim Wanderer, Junlan Zhou, Min Zhu, et al. B4: Experience with a Globally-Deployed Software Defined WAN. *ACM SIGCOMM Computer Communication Review*, 43(4):3–14, 2013.

[21] Vimalkumar Jeyakumar, Mohammad Alizadeh, David Mazières, Balaji Prabhakar, Albert Greenberg, and Changhoon Kim. EyeQ: Practical Network Performance Isolation at the Edge. In *10th USENIX Symposium on Networked Systems Design and Implementation (NSDI 13)*, pages 297–311, 2013.

[22] Lavanya Jose, Lisa Yan, George Varghese, and Nick McKeown. Compiling Packet Programs to Reconfigurable Switches. In *12th USENIX Symposium on Networked Systems Design and Implementation (NSDI 15)*, pages 103–115, 2015.

[23] Alexander Krentsel, Nitika Saran, Bikash Koley, Subhasree Mandal, Ashok Narayanan, Sylvia Ratnasamy, Ali Al-Shabibi, Anees Shaikh, Rob Shakir, Ankit Singla, and Hakim Weatherspoon. A Decentralized SDN Architecture for the WAN. ACM SIGCOMM '24, page 938–953, New York, NY, USA, 2024. Association for Computing Machinery.

[24] Umesh Krishnaswamy, Rachee Singh, Nikolaj Bjørner, and Himanshu Raj. Decentralized Cloud Wide-Area Network Traffic Engineering with BLASTSHIELD. In *19th USENIX Symposium on Networked Systems Design and Implementation (NSDI 22)*, pages 325–338, 2022.

[25] Alok Kumar, Sushant Jain, Uday Naik, Anand Raghuraman, Nikhil Kasinadhuni, Enrique Cauich Zermeno, C Stephen Gunn, Jing Ai, Björn Carlin, Mihai Amarandei-Stavila, et al. BwE: Flexible, Hierarchical Bandwidth Allocation for WAN Distributed Computing. In *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication*, pages 1–14, 2015.

[26] Jean-Yves Le Boudec. Rate adaptation, Congestion Control and Fairness: A Tutorial. 01 2002.

[27] J Medved, S Previdi, A Farrel, and S Ray. RFC 7752: North-Bound Distribution of Link-State and Traffic Engineering (TE) Information Using BGP, 2016.

[28] Dritan Nace and Michal Pióro. Max-min Fairness and Its Applications to Routing and Load-Balancing in Communication Networks: A Tutorial. *IEEE Communications Surveys & Tutorials*, 10(4):5–17, 2008.

[29] Tian Pan, Nianbing Yu, Chenhao Jia, Jianwen Pi, Liang Xu, Yisong Qiao, Zhiguo Li, Kun Liu, Jie Lu, Jianyuan 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.

[30] Lucian Popa, Gautam Kumar, Mosharaf Chowdhury, Arvind Krishnamurthy, Sylvia Ratnasamy, and Ion Stoica. FairCloud: Sharing the Network in Cloud Computing. In *Proceedings of the ACM SIGCOMM 2012 conference on Applications, technologies, architectures, and protocols for computer communication*, pages 187–198, 2012.

[31] Lucian Popa, Praveen Yalagandula, Sujata Banerjee, Jeffrey C Mogul, Yoshio Turner, and Jose Renato Santos. ElasticSwitch: Practical Work-Conserving Bandwidth Guarantees for Cloud Computing. In *Proceedings of the ACM SIGCOMM 2013 conference on SIGCOMM*, pages 351–362, 2013.

[32] Sivasankar Radhakrishnan, Yilong Geng, Vimalkumar Jeyakumar, Abdul Kabbani, George Porter, and Amin Vahdat. SENIC: Scalable NIC for End-Host Rate Limiting. NSDI'14, page 475–488, USA, 2014. USENIX Association.

[33] Barath Raghavan, Kashi Vishwanath, Sriram Ramabhadran, Kenneth Yocum, and Alex C. Snoeren. Cloud Control with Distributed Rate Limiting. *SIGCOMM Comput. Commun. Rev.*, 37(4):337–348, August 2007.

[34] Henrique Rodrigues, Jose Renato Santos, Yoshio Turner, Paolo Soares, and Dorgival Guedes. Gatekeeper: Supporting Bandwidth Guarantees for Multi-Tenant Datacenter Networks. In *3rd Workshop on I/O Virtualization (WIOV 11)*, 2011.

[35] Harvey M Salkin and Cornelis A De Kluyver. The Knapsack Problem: A Survey. *Naval Research Logistics Quarterly*, 22(1):127–144, 1975.

[36] Rachee Singh, Nikolaj Bjorner, Sharon Shoham, Yawei Yin, John Arnold, and Jamie Gaudette. Cost-Effective Capacity Provisioning in Wide Area Networks with Shoofly. In *Proceedings of the 2021 ACM SIGCOMM 2021 Conference*, SIGCOMM '21, page 534–546, New York, NY, USA, 2021. Association for Computing Machinery.

[37] Chi yao Hong, Subhasree Mandal, Mohammad A. Alfares, Min Zhu, Rich Alimi, Kondapa Naidu Bollineni, Chandan Bhagat, Sourabh Jain, Jay Kaimal, Jeffrey Liang, Kirill Mendelev, Steve Padgett, Faro Thomas Rabe, Saikat Ray, Malveeka Tewari, Matt Tierney, Monika Zahn, Jon Zolla, Joon Ong, and Amin Vahdat. B4 and After: Managing Hierarchy, Partitioning, and Asymmetry for Availability and Scale in Google's Software-Defined WAN. In *SIGCOMM'18*, 2018.

[38] Xinchun Zhang, Aqsa Kashaf, Yihan Zou, Wei Zhang, Weibo Liao, Haoxiang Song, Jintao Ye, Yakun Li, Rui Shi, Yong Tian, Wei Feng, Binbin Chen, Zuzhi Chen,

Tieying Zhang, and Yongping Tang. ResLake: Towards Minimum Job Latency and Balanced Resource Utilization in Geo-Distributed Job Scheduling. *Proc. VLDB Endow.*, 17(12):3934–3946, August 2024.

[39] Hang Zhu, Varun Gupta, Satyajeet Singh Ahuja, Yuandong Tian, Ying Zhang, and Xin Jin. Network Planning with Deep Reinforcement Learning. In *Proceedings of the 2021 ACM SIGCOMM 2021 Conference*, SIGCOMM '21, page 258–271, New York, NY, USA, 2021. Association for Computing Machinery.

A Redundancy and Scalability

Net-P4ct is designed with high availability and failover mechanisms to ensure reliable operation in production.

P4 Switch Resilience: Each data center deploys a cluster of P4 switches, all directly connected to the WAN uplinks. In the event of hardware or port-level failures, BGP convergence automatically redirects traffic through healthy paths. For more severe failures (such as a firmware bug or full cluster outage), the system supports a failover mode in which traffic is bypassed from the P4 switches entirely and forwarded through the physical switches using predefined routes. This ensures that traffic continues to flow even when in-network enforcement is temporarily disabled.

Control Plane Resilience: Developed as a cloud-native application, the central controller runs in a stateless manner. Multiple controller replicas are deployed across data centers, and any instance can serve as the active coordinator. All persistent data (metadata, service request, traffic metrics) is stored in ByteDance's distributed infrastructure-grade database, which offers automatic replication and failover.

If communication between the controller and a P4-agent is interrupted, the affected switches continue operating using the last configurations. Once the connection is restored, updated policies are seamlessly applied. This design ensures that fairness enforcement degrades gracefully rather than disrupting service, and policy accuracy is restored without manual intervention.

Scalability: The central controller is stateless by design, allowing horizontal scaling through the deployment of additional replicas as needed. On the data plane side, Net-P4ct supports scalability by allowing multiple P4 switch clusters to be deployed across data centers and WAN ingress points. This architecture allows the system to accommodate increasing service demands.

B Switch Resource Consumption

Switch Resource Consumption: Table summarizes the additional switch resource usage introduced by Net-P4ct's new functionalities for a single pipe (Net-P4ct-related pipeline). Here, Baseline denotes the logical pipe in the basic switch that provides forwarding, port monitoring, and ACL. Because Net-P4ct reuses the baseline's classify table in another pipe, the incremental TCAM usage is 0. The results show that most resource overheads are below 5%, demonstrating mild overhead in our implementation. With all available pipeline resources, Net-P4ct can support at least 128K service jobs per switch. The main constraint is Map RAM, which is used for 64-bit counters in SBP. This can be further optimized by shortening feedback cycles or compressing to 32 bits, thereby reducing SRAM usage by half.

Table 3: Switch Resource Usage of a Single Service Pipe

Resource	Usage (%)		
	Baseline	Baseline + Net-P4ct	Increased Usage
GateWay	7.8	10.9	3.1
Hash Bit	6.6	9.8	3.2
Meter ALU	4.2	8.3	4.1
TCAM	7.3	7.3	0
SRAM	13.1	24.6	11.5
Map RAM	4.5	17.2	12.7
Stash	6.3	9.9	3.6
Stats ALU	12.5	12.5	0
Action Data Bus Bytes	7.8	10.2	3.4
Exact Match Input Xvar	4.5	7.9	3.4
Logical Table ID	18.8	22.9	4.1