Pushing the Performance Limits of Datacenter Networks with Fine-grained Priority Assignment

Rui Zhuang, Jiangping Han, *Member, IEEE*, Kaiping Xue, *Senior Member, IEEE*, Jian Li, *Member, IEEE*, David S.L. Wei, *Life Senior Member, IEEE*, Ruidong Li, *Senior Member, IEEE*, Qibin Sun, *Fellow, IEEE*, Jun Lu

***Abstract*—Priority plays a crucial role in distinguishing the diverse demands of applications and improving the performance of underlying networks. However, application-provided priority is insufficient to cope with wide variations in traffic characteristics and dynamic network conditions. When assigning priorities to flows, existing proposals are limited to constrained dimensions, rendering them inadequate for accurately and rapidly identifying flow importance. To address this problem, we present Firapam, a novel priority assignment strategy that systematically combines important traffic states to achieve fine-grained and dynamic priority assignments. Firapam employs convex optimization to adaptively respond to changes in requirements and the environment, and implements admission control for flow priority in a distributed manner. Consequently, Firapam significantly reduces the flow completion time and deadline miss rate. We analytically and experimentally demonstrate that Firapam can effectively support existing priority assignments with significant performance improvements. Compared to state-of-the-art priority assignment methods, Firapam exhibits a remarkable decrease in deadline miss rate by 14.5% to 52.3%, across diverse traffic patterns. Moreover, it reduces the flow completion tail by 17.9% and ensures a minimum reduction of 72.3% in deadline miss rate under high network loads.**

***Index Terms*—Datacenter network, traffic priority, deadline, fine-grained priority assignment**

I. Introduction

Modern datacenters provide a wide range of services, including cloud computing, virtualization, storage, and data processing [1], [2], all of which have diverse demands and are crucially reliant on the underlying network for performance. These services support two common types of applications in datacenters. The first comprises user-oriented applications such as recommendation system, web search, and social network [3]–[5]. This type of applications expands rapidly in recent years, generating a diverse mix of large and small flows with strict latency requirements (e.g., deadline). The second comprises enterprise-class applications such as parallel computing and high-performance storage [6]–[8]. This type of applications exhibits increasing demands for performance, generating flows that require ultra-low latency and high throughput to minimize flow completion times (FCTs).

With the expansion of datacenter networks (DCNs) and the enrichment of application types, service providers in datacenters face the challenge of managing incoming application streams to achieve both rapid completion (users' goal) and high resource efficiency (providers' goal) [9]. Unfortunately, service network performance currently exhibits high heterogeneity, with each of the network components/resources being dynamically shared by numerous competing services with diverse workloads, making it particularly challenging to meet the above requirements. Firstly, due to the diverse demands of applications and the strict performance requirements of critical traffic, services in DCNs must accommodate a diverse set of workloads in terms of resource and performance requirements. However, changes in network conditions may benefit some services while degrading others [10], thereby subjecting some services with unique requirements to performance limitations. Secondly, operators often lack prior knowledge of the workload and demand characteristics, thereby hindering their capability to achieve optimal traffic management or resource allocation. Consequently, as traffic complexity increases, the imperative of achieving reliable traffic classification intensifies significantly, playing a crucial role in ensuring system performance and scalability.

Priority is a powerful tool for marking traffic importance and enhancing network performance [11], [12], which significantly contributes to achieving the above goals. For example, in the design of application layer protocols, priority fields related to Quality of Service (QoS) can be added to label the packet's priority and achieve traffic classification. Under limited available capacity, applications utilize priorities to prioritize the traffic, providing the DCN with a clear understanding of flow importance and urgency [13]. By setting priorities based on the importance of traffic, switches can achieve traffic isolation and differentiated service quality according to the priorities. In this way, performance-critical traffic with high priority is permitted to move ahead of lower-priority traffic to prevent congestion from impeding the flow of vital traffic, thereby ensuring that critical applications and services obtain the bandwidth required for optimal operation.

Understanding how to prioritize and keep network traffic running smoothly for every flow is pivotal for effectively leveraging priority to differentiate critical flows from non-critical flows and enhance overall performance. This process of determining the priority is known as priority assignment (PA). Existing proposals implement PAs from different demand

R. Zhuang, J. Han, K. Xue, J. Li, Q. Sun and J. Lu are with the School of Cyber Science and Technology, University of Science and Technology of China, Hefei, Anhui 230027, China.

J. Lu is also with the Department of Electronic Engineering and Information Science, University of Science and Technology of China, Hefei, Anhui 230027, China.

D. Wei is with the Department of Computer and Information Science, Fordham University, Bronx, NY 10458, USA.

R. Li is with the Institute of Science and Engineering, Kanazawa University, Kakuma, Kanazawa 920-1192, Japan.

Corresponding Author: J. Han (e-mail: jphan@ustc.edu.cn), and K. Xue (e-mail: kpxue@ustc.edu.cn).

perspectives, which can be the flow states (e.g., flow size, deadline) [14], [15] and service level objectives (SLOs) [12]. However, existing PAs are still deficient in meeting the actual requirements of complex flows. First, existing PAs only refer to constrained dimensions of flow state, which is unable to accurately describe flow importance and urgency in traffic with mixed demands. When the selected flow state is not the best measure, PAs may assign inappropriate priorities and give rise to the priority inversion problem [12], [16], [17], leading to some flows to starve while others to be overserved. Second, existing PAs cannot swiftly identify flow importance in dynamic network environments with wide variations of traffic characteristics. Typically, existing PAs adopt fixed or heuristic methods to assign priorities, which require classifying priority levels periodically based on historical information or global traffic information. However, due to the long update cycle that surpasses the end-to-end delay, their responsiveness to network dynamics (e.g., flow distribution changes, network load changes, and link quality changes) is limited. Consequently, as traffic characteristics change within the network, their effectiveness may drop considerably over a period of time.

Our analysis reveals that existing PA methods are still far from perfection. Due to their reliance on limited traffic information and coarse-grained strategies, they are only able to provide a restricted range of service quality. To address this problem, PA methods should refer to additional traffic information, such as multi-dimensional flow state and real-time network metrics, to analyze flow importance from multiple perspectives. However, coupling this information to achieve precise and detailed priority assignment presents significant challenges. Priorities assigned from different demand perspectives may block each other. For example, large flows with higher priorities can block delay-sensitive small flows with lower priorities. Moreover, PA needs to be more responsive to network dynamics to provide reliable delay guarantees for critical flows under changing network conditions. Incorporating real-time network states into PA to optimize the proportion of high-priority flows within the network also presents additional challenges.

From the above insights, we design a novel priority assignment strategy, named Firapam, to achieve fine-grained assignment of available priorities for flows in DCNs. Firapam implements admission control on existing flow priorities and carefully divides the traffic importance based on more contextual information. By adapting the PA method to the diverse demands of traffic and the real-time state of the network, Firapam significantly enhances the service quality. We design a utility function that effectively couples the multi-dimensional state of traffic and network dynamics. The network utility captures both flow characteristics and current network states for every control action, providing rapid and accurate responses to changes in network status, traffic distribution, and requirement distribution. Firapam continuously improves the network utility through convex optimization and adjusts the admission control for flow priorities. In our work, we demonstrate that Firapam effectively prevents low-priority flows from experiencing ultra-long flow completion times (FCTs) and achieves a more equitable allocation of network resources compared to existing PA methods. Extensive simulation experiments show that Firapam outperforms state-of-the-art PA methods across key metrics, including FCT, deadline miss rate, fairness, and convergence, resulting in significant performance gains for priority-based proposals. Evaluated across bursty traffic environment and high workload environment, Firapam reduces the deadline miss rate of existing PA methods by 14.5%$\sim$52.3%, and reduces the $99th$ percentile flow completion tail by 0.2%$\sim$17.9% and deadline miss rate by 72.3%$\sim$82%, respectively.

This paper makes the following main contributions:

- We introduce an analytical framework for priority assignment in DCNs, which constructs the assignment laws of PA methods from different demand perspectives, reveals their drawbacks, and introduces the concept of *demand rate* to achieve optimal priority assignment.
- We design Firapam, a novel priority assignment strategy to optimize the performance of diverse application workloads within modern DCNs. Firapam systematically combines multi-dimensional state of traffic to conduct reliable admission control for flow priorities, and leverages convex optimization to achieve rapid response to changes in both requirements and the environment.
- We show analytically and experimentally that Firapam possesses desirable properties of stability, convergence, and fairness. The benefits of Firapam are evaluated in traditional DCNs across several workloads.

The rest of this paper is organized as follows. Section II introduces the PA methods adopted by existing priority-based proposals. Section III proposes a framework to define existing PA methods and analyze their basic problems, and provides a detailed motivation for our work. Section IV presents our fine-grained priority assignment strategy, Firapam, and proves its advantageous properties. Section V provides an in-depth performance evaluation of Firapam. Finally, Section VI concludes the paper.

II. Background and Motivation

To motivate our design, we summarize the PA methods adopted by prevailing priority-based proposals, and show the performance trade-offs with different PA methods when handling mix-flows.

A. Priority Assignment in Datacenter Networks

In numerous proposals within DCNs, the utilization of priority is crucial in effectively managing queue accumulation and achieving the lowest possible latency. These proposals include self-adjusting at endpoints [18], flow scheduling and load balancing at switches [19]–[21], in-network caching [22], and priority-based flow control [23].

The traditional priority assignment in DCNs revolves around the attributes of flows, i.e., the sizes and requirements of flows. Flow requirements typically manifest through the flow characteristics of Quality of Service (QoS), including delay, jitter, and throughput. Table I summarizes the PA methods

TABLE I
BYTES-BASED AND LATENCY-BASED PRIORITY ASSIGNMENT METHODS OF EXISTING PRIORITY-BASED PROPOSALS

Proposal	Priority at the application level	Admission control	Description
Bytes-based priority assignment			
pFabric [24]	$p = S_{remF}$	no admission control	Every packet carries a single priority set based on the remaining flow size.
PIAS [25]	$p = f_{CE}(B_{sent}, \alpha) = \begin{cases} p_1,\ B_{sent} < \alpha_1 \text{ (highest)} \\ p_i,\ \alpha_{i-1} \le B_{sent} < \alpha_i \\ p_k,\ \alpha_{k-1} \le B_{sent} \text{ (lowest)} \end{cases}$	no admission control	A Central Entity (CE) periodically (a long period of time) collects the traffic information (i.e., flow size) reported from each end host to calculate the demotion threshold set $\{\alpha\}$.
Aalo [26]	$p = f_{CE}(B_{sent}, \alpha)$	no admission control	Uses exponentially-spaced thresholds $\{\alpha\}$ set by the operator in advance.
Karuna [16]	$p = p_1$, deadline flow (highest) or $\begin{cases} SJF(S_F, \beta), \text{ know size} \\ f_{CE}(B_{sent}, \alpha), \text{ others} \end{cases}$	no admission control	A CE periodically collects traffic information to calculate threshold sets $\{\alpha\}$ and $\{\beta\}$, then distributes them to the end-host modules.
Auto [27]	$p = CDF_{CS}(S_F, \alpha)$	no admission control	A Central System (CS) collects global traffic information to optimize and set $\{\alpha\}$ by deep reinforcement learning (DRL).
Homa [14]	$p = CDF_{receiver}(S_F)$	no admission control	Each receiver determines the priorities according to the message size distribution.
Latency-based priority assignment			
DeTail [28]	$p = p(latency_{rm})$	no admission control	Only two priority levels are actually used, p is set by the application based on flow deadlines.
PASE [11]	$p = p(latency_{rm})$	$p_{admit} = \begin{cases} 2,\ ADH \le C \\ 0,\ ADH > C \end{cases}$	Arbitrators dynamically change the mapping of flows to queues and make decisions in a bottom up fashion.
Aequitas [12]	$p = p_{RPC} = \begin{cases} PC, \text{ Performance-critical} \\ NC, \text{ Non-critical} \\ BE, \text{ Best-effort} \end{cases}$	$p_{admit} = \begin{cases} \min(p_{admit} + a, 1), \\ \quad latency/S_F < target \\ \max(p_{admit} - b \cdot S_F, floor), \\ \quad latency/S_F > target \end{cases}$	Senders makes QoS admit or QoS degrade decisions for an remote procedure call (RPC) through admission control based on the satisfactory degree for RPC latency to latency target.
Poche [22]	$p = p(latency_{rm})$	no admission control	Categorizes traffic into a given number of priorities related to the latency requirements of flows.

of existing priority-based proposals, where S_F represents flow size, S_{remF} is the remaining flow size, B_{sent} is the number of bytes that have been sent, $latency_{rm}$ is the latency requirement of a flow. ADH is the sum of demands of flows with priority higher than current flow, C is the link capacity, $target$ is the SLO target of current QoS level. When $0 \le p_{admit} < 1$, the priority p of a flow is mapped to the lowest priority level with a probability of $(1 - p_{admit})$. When $1 \le p_{admit} \le 2$, p is mapped to the highest priority level with a probability of $(p_{admit} - 1)$.

The PA methods shown in Table I are fundamentally limited to one of the two dimensions (bytes or latency requirement), which can be divided into two types: bytes-based PA methods and latency-based PA methods.

1) Bytes-based PA methods: The first category of methods prioritizes traffic based on byte-related metrics, such as flow size, remaining flow size, and bytes that have been sent. Consequently, this category of methods tends to favor small flows: small flows typically receive higher priorities compared to large flows and are more likely to be placed in the first few high-priority queues. Representative proposals employing such methods include pFabric [24], PIAS [25], Aalo [26], Karuna [16], Auto [27], and Homa [14]. Bytes-based PA methods emphasize factors like throughput and fairness, making them more suitable for applications with high tolerance for latency and network quality fluctuations. Specifically, in pFabric, every packet carries a single priority set based on the remaining flow size. PIAS and Karuna both require a central entity (CE) to periodically collect traffic information (i.e., flow size) reported from each end host to calculate the demotion thresholds α. Aalo uses exponentially-spaced thresholds preset by the operator to determine the priorities of flows of different sizes. In Auto, a central system (CS) collects global traffic information to optimize α via deep reinforcement learning (DRL). Homa computes the cumulative distribution function (CDF) of flow size at the receiver to determine the range of message sizes for each priority level, so as to evenly distribute traffic among all priority levels.

2) Latency-based PA methods: The second category of methods prioritizes traffic based on latency-related metrics, such as deadline [3] and service level objective (SLO). Representative proposals employing such methods include DeTail [28], PASE [11], Aequitas [12], and Poche [22]. Latency-based PA methods focus on guaranteeing the timeliness and effectiveness of data, with the goal of reducing flow completion tail and improving worst-case application performance. Specifically, DeTail uses two priority levels and allows applications to set the priority based on flow deadlines. PASE deploys arbitrators to dynamically adjust the mapping of flow priorities to queues and make decisions in a bottom-up fashion. The source ends provide the arbitrators with two key pieces of information: criteria for scheduling (i.e., flow size, deadline or task ID) and demands (i.e., maximum rate allowed by the source end to send data). Aequitas maps RPC priority classes to network QoS levels based on the satisfactory degree of RPC latency to latency target, so as to ensure a consistently high-quality network

This article has been accepted for publication in IEEE Transactions on Network Science and Engineering. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TNSE.2024.3429402

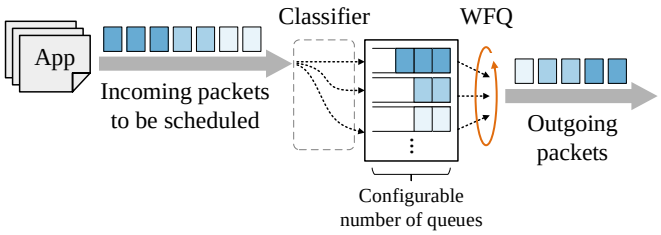
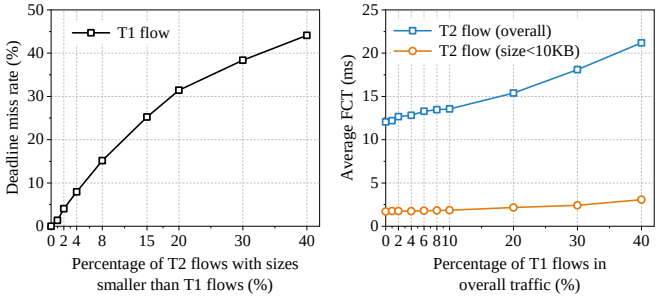


Fig. 1. The operating principle of the WFQ.



(a) Bytes-based PA hurts T1 flows (b) Latency-based PA hurts T2 flows

Fig. 2. Existing priority assignment methods lead to suboptimal performance in handling traffic with differentiated requirements.

experience for performance-critical traffic. Poche categorizes traffic into a given number of priority levels related to the latency requirements of flows.

In summary, existing PA methods mainly adopt fixed or heuristic strategies for determining the priority of each flow, with some methods refer to the global or local traffic information. However, in DCNs with dynamically changing traffic patterns, the timeliness of historical information diminishes significantly, making PAs based on historical traffic information unable to achieve accurate results.

B. Trade-offs in Different Priority Assignment Methods

Modern datacenter applications have diverse transmission requirements, with the generated traffic exhibiting varying demands for latency and bandwidth. However, traffic with differentiated requirements can conflict with each other within the network [16], posing challenges for existing priority-based proposals. Focusing solely on a certain dimension of requirements during priority assignment will inevitably lead to compromised performance for other types of traffic with dissimilar requirements.

To illustrate this problem, we conduct simulations using YAPS [29], and demonstrate from a scheduling perspective that applying bytes-based or latency-based PA methods hurts the performance of different types of flows in mixed scenarios. In our experiments, all hosts are interconnected via switches, and Weighted Fair Queuing (WFQ) is adopted on the switches for queue scheduling and congestion management. Both bytes-based and latency-based PA methods are applied at the end hosts to determine the priority of each flow. As shown in Fig. 1, flow priorities are mapped to WFQ QoS levels, and the allocation of bandwidth for each queue is subsequently refined by WFQ on the switches, thereby ensuring a greater share of bandwidth for incoming packets from high-priority flows. There are two types of flows in the simulated network: deadline flows (T1) and background flows (T2). A simulated query/response application generates queries through a Poisson process, and controls the flow sizes and deadlines of the response flows (i.e., deadline flows). Meanwhile, the background flows are generated using a similar Poisson process. The sizes of all flows are drawn from realistic traffic traces [30]–[33], and the average workload is set to 8Gbps ($0.8\times$ load).

Case against bytes-based PA: In the first experiment, we set priorities and schedule flows strictly based on their sizes. As depicted in Fig. 2(a), the implement of bytes-based PA results in a detrimental impact on the deadline miss rate of deadline flows. Specifically, as the percentage of small background flows (T2) increasing to 20%, bytes-based PA leads to over 30% of the T1 flows missing their deadlines. This is because flow size alone determines which packets to go first, which prevents deadline flows, especially the larger ones, from meeting their deadlines. Consequently, the greater the percentage of T2 flows with sizes smaller than T1 flows, the higher the deadline miss rate of T1 flows.

Case against latency-based PA: In the second experiment, we set priorities and schedule flows based on their deadlines, and give strict priority to deadline flows over background flows. Fig. 2(b) shows that the average FCT of T2 flows increases as the percentage of T1 flows in overall traffic rises. Specifically, as the percentage of T1 flows increasing from 0 to 30%, the average FCT for delay-sensitive short T2 flows (size$<$10KB) increases by 42.4%; and for all T2 flows, the average FCT increases by 50.2%. This is because T1 flows are processed with priority, allowing them to quickly take up a significant portion of the available bandwidth. Consequently, the higher the proportion of T1 flows, the worse the completion efficiency of T2 flows.

III. Analytical Framework for Priority Assignment Problems

We design an analytical framework for the priority assignment problem in DCNs and construct the assignment laws from different demand perspectives. First, we build the traffic model and requirement model of the network. Then, we construct the assignment laws of different PA methods on the basis of these models, and analytically identify the drawbacks of existing assignment laws from a theoretical perspective. Finally, we outline the fundamental design goals of our priority assignment strategy.

A. Network Traffic Model and Requirement Model

Suppose flows in the network can be divided into N classes according to their flow sizes, and the average flow size of class n is S_n. So the flow size model $\mathbb{S}$ can be expressed by a matrix:

$$\mathbb{S} = [S_1, S_2, \cdots, S_N]^\top, \tag{1}$$

where $0 < S_1 < S_2 < \cdots < S_N$.

The total number of flows in this network is f_t. Let the proportion of flows in class n to the total number of flows be G_n, and set a diagonal matrix $\mathbb{G} = \mathrm{diag}\left(G_1, G_2, \cdots, G_N\right)$, where $\sum_{n=1}^{N} G_n = 1$. So the traffic model $\mathbb{T}$ that describes the traffic distribution characteristics from the perspective of flow size can be expressed as:

$$\mathbb{T} = f_t \mathbb{G}\mathbb{S} = f_t [G_1 S_1, G_2 S_2, \cdots, G_N S_N]^\top. \tag{2}$$

Then, suppose flows in the network can be divided into M classes according to their latency requirements (e.g., deadline), and the average latency requirement of class m is L_m. So the flow requirement model $\mathbb{D}$ can be expressed by a matrix:

$$\mathbb{D} = [L_1, L_2, \cdots, L_M]^\top, \tag{3}$$

where $0 < L_1 < L_2 < \cdots < L_M$.

For a flow size class n, assume that the flows in class n have differentiated latency requirements, and the distribution of latency requirement is $K_n = [l_{n1}, l_{n2}, \cdots, l_{nM}]$, where $\sum_{m=1}^{M} l_{nm} = 1$, $l_{nm} \geq 0$. It means the proportion of flows with latency requirements of L_m in class n is l_{nm} to the number of flows in class n, and $l_{nm} \cdot G_n$ to the total number of flows. The latency requirement distribution $\mathbb{L}$ for all flows is:

$$\mathbb{L} = [K_1, K_2, \cdots, K_N]^\top = \begin{bmatrix} l_{11} & \cdots & l_{1M} \\ \vdots & \ddots & \vdots \\ l_{N1} & \cdots & l_{NM} \end{bmatrix}. \tag{4}$$

So the requirement model $\mathbb{R}$ that describes the traffic distribution characteristics from the perspective of latency requirement can be expressed as:

$$\mathbb{R} = f_t \mathbb{G}\mathbb{L}\mathbb{D}. \tag{5}$$

B. Assignment Laws of Existing Priority Assignment Methods and Their Limitations

As discussed in Section II-A, from different demand perspectives, PA methods can be broadly classified into two types: bytes-based PA and latency-based PA.

1) Assignment Laws of Bytes-based PA: Bytes-based PA methods are widely used in most proposals. This category of methods is flow-size-critical, whose assignment laws have no relation with latency requirement but flow size, and can be represented as:

$$p_S = f_S\left(S_F\right) = \left[\gamma_S\left(S_F - \alpha_S\right)\right]^{e_S} + \beta_S, \tag{6}$$

where S_F denotes flow size, $\gamma_S > 0$, $e_S < 0$, α_S and β_S jointly control the upper and lower limits of available priority levels. Parameter γ_S denotes the exponential moving average parameter related to $\mathbb{T}$. Set the standard deviation of $\mathbb{T}$ as σ, and we have:

$$\gamma_S = k\sigma \quad (k > 0). \tag{7}$$

2) Assignment Laws of Latency-based PA: Latency-based PA methods are mainly used in proposals that aim to satisfy flow deadlines or provide predictable latency guarantees. This category of methods is FCT-critical, whose assignment laws can be represented as:

$$p_L = f_L\left(L_R\right) = \left[\gamma_L\left(L_R - \alpha_L\right)\right]^{e_L} + \beta_L, \tag{8}$$

where L_R denotes latency requirement, $e_L < 0$, α_L and β_L jointly control the upper and lower limits of available priority levels. The exponential moving average parameter γ_L is irrelevant to $\mathbb{T}$.

The larger the value of p_S or p_L, the higher the priority of flows. Priority determines the scheduling opportunity or bandwidth resource available to a flow. With limited network capacity, we assume that flows with higher priority will be processed preferentially, and flows will be transmitted in the order of their priorities. So for any flow r, its average sending rate is proportional to its priority p_r, i.e., $\lambda_r = \mu p_r$, where μ is a constant that describes the proportional relationship between the two variables.

3) Limitations of Existing PA Methods: In this section, we aim to illustrate the inefficiencies of different PA methods in dealing with network dynamics and differentiated demands.

First, we solve the flow completion time (FCT) and deadline miss rate of bytes-based PA methods. For all traffic in the network (i.e., $\mathbb{T}$), their overall flow completion time FCT_{all}^B is calculated as:

$$\begin{aligned} FCT_{all}^B &= \sum_{i=1}^{f_t} FCT_i^B = \frac{\mathbb{T}}{\mu f_S\left(\mathbb{T}\right)} \\ &= \frac{1}{\mu} \cdot \sum_{n=1}^{N} \frac{S_n}{\gamma_S{}^{e_S}\left(S_n - \alpha_S\right)^{e_S} + \beta_S}, \end{aligned} \tag{9}$$

which can be expressed as a function of $\gamma_S{}^{-e_S}$:

$$FCT_{all}^B = F_1\left(\mathbb{G}\right) = F_1\left(\gamma_S{}^{-e_S}\right) = a_1 \gamma_S{}^{-e_S} + b_1, \tag{10}$$

where $a_1 > 0$, $b_1 > 0$.

The overall deadline miss rate of $\mathbb{T}$ can be expressed as a function of $\mathbb{G}$:

$$R_{DM}^B = R_1\left(\mathbb{G}\right) = \sum_{n=1}^{N} G_n \sum_{m=1}^{f_1\left(L_d, FCT_n^B\right)} l_{nm}, \tag{11}$$

where $f_1\left(L_d, FCT_n^B\right) = d$ when $L_d \leq FCT_i^B < L_{d+1}$. As $\mathbb{T}$ changes from uniform distribution to uneven distribution, σ and FCT_n^B both increase, and $\sum_{m=1}^{f_1\left(L_d, FCT_n^B\right)} l_{nm}$ increases correspondingly, resulting in increased R_{DM}^B and more flows in $\mathbb{T}$ missing their deadlines.

By combining Eq. (10) with Eq. (7), we observe that FCT_{all}^B increases as σ increases, which means FCT_{all}^B is influenced by network traffic distribution. In instances where the majority of bytes are concentrated within a few flow size classes, FCT_{all}^B will be notably higher compared to a uniformly distributed situation, even if the total number of bytes remains constant. Therefore, bytes-based PA methods are likely to experience performance degradation when a large number of similarly-sized flows burst. Moreover, bytes-based

PA methods cannot adapt well to network dynamics. Their ability to achieve optimal priority assignment mainly depends on the distribution characteristics of traffic. As network traffic characteristics change, large flows may suffer from starvation and extended FCTs due to their low priorities, consequently increasing the overall FCT and impeding delay-sensitive flows from completing in time.

Second, we solve the FCT and deadline miss rate of latency-based PA methods. The overall flow completion time FCT_{all}^{L} of $\mathbb{T}$ can be calculated as:

$$\begin{aligned} FCT_{all}^{L} &= \sum_{i=1}^{f_t} FCT_i^L = \frac{\mathbb{T}}{\mu f_L\left(\mathbb{R}\right)} \\ &= \frac{1}{\mu} \cdot \sum_{n=1}^{N} \frac{G_n S_n}{\sum_{m=1}^{M} l_{nm}\left[\gamma_L{}^{e_L}\left(L_m - \alpha_L\right)^{e_L} + \beta_L\right]}, \end{aligned} \tag{12}$$

which can also be expressed as a function of G_n and l_{nm} (i.e., $\mathbb{G}$ and $\mathbb{L}$):

$$FCT_{all}^{L} = F_2\left(\mathbb{G},\mathbb{L}\right). \tag{13}$$

Let $FCT_{S_n,L_m}^{L} = \frac{S_n}{\mu f_L(L_m)}$, the overall deadline miss rate of $\mathbb{T}$ can be expressed as a function of $\mathbb{R}$:

$$\begin{aligned} R_{DM}^{L} &= R_2\left(\mathbb{R}\right) \\ &= \sum_{n=1}^{N} G_n \sum_{m=1}^{M} f_2\left(L_m, FCT_{S_n,L_m}^{L}\right) \cdot l_{nm}, \end{aligned} \tag{14}$$

where $f_2\left(L_m, FCT_{S_n,L_m}^{L}\right) = 1$ when $L_m < FCT_{S_n,L_m}^{L}$, or else $f_2\left(L_m, FCT_{S_n,L_m}^{L}\right) = 0$.

In Eq. (12), FCT_{all}^{L} varies with the requirement distribution $\mathbb{L}$ in the same traffic model $\mathbb{T}$. And in Eq. (14), because $f_2\left(L_m, FCT_{S_n,L_m}^{L}\right)$ is known for L_m and S_n, R_{DM}^{L} is mainly affected by traffic distribution $\mathbb{G}$ and latency requirement distribution $\mathbb{L}$. R_{DM}^{L} increases both when the proportion of large flows increases or the overall latency requirement decreases, and flows with larger sizes in $\mathbb{T}$ are more likely to miss their deadlines. Due to the finite priority queues and buffer resources within DCNs, if some large flows are set with short latency requirements, although their FCTs will decrease correspondingly, the cost is to occupy the resources of small flows at the same priority level and increase their deadline miss rate. Similarly, a small portion of large flows with long latency requirements may also experience ultra-long FCTs, contributing to an increase in FCT_{all}^{L}. Therefore, latency-based PA methods exhibit limited adaptability to complex traffic.

C. Observations and the Notion of Demand Rate

We derive two key observations from the above analysis:

First, both bytes-based and latency-based PAs fail to accurately meet the diverse demands of flows. Bytes-based PA is desirable for ensuring fast completion of most flows, at the expense of the performance of large flows. However, due to the fact that flow size and flow importance are often misaligned, the priority inversion problem [16], [17] occurs frequently in bytes-based PA. Latency-based PA allows flows with short latency requirements to complete quickly with smaller FCT, yet this is unnecessary for meeting their deadlines and instead takes up network resources for other flows.

Second, bytes-based and latency-based PAs essentially only refer to limited flow states and cannot adapt well to complex and ever-changing network environments. Once the modeled network differs significantly from the real network, it leads to a surprising degree of mismatch between actual flow priorities and their supposed importance in the network. This makes it difficult to allocate limited resources to performance-critical flows based solely on application-provided priorities, resulting in serious performance losses based on existing PA methods.

Based on these observations, we introduce the concept of *demand rate* to systematically combine multi-dimensional flow states (e.g., number of bytes and latency requirement), thereby designing a fine-grained priority assignment strategy capable of reducing both FCT and deadline miss rate. Notably, our analysis of Eq. (11) and Eq. (12) suggests that deadline miss rate and FCT are related to the ratio of number of bytes to latency requirement, which is defined as *demand rate*. *Demand rate* is formally represented as:

$$D^R = B_r / \delta_r^o, \tag{15}$$

where B_r and δ_r^o are the number of bytes and the deadline of flow r, respectively. We further investigate the accurate relationship of *demand rate* and priority assignment in the next section.

IV. Fine-grained Priority Assignment Strategy

Based on our observations in Section III, we design a **fi**ne-g**ra**ined **p**riority **a**ssign**m**ent strategy, called **Firapam**, which references multi-dimensional flow states and network states to dynamically map the expected priority (e.g., application-defined priority) to actual priority through convex optimization. For Firapam, the priority of a flow is not fixed but can be dynamically adjusted. Firapam determines what traffic should get access to limited resources when demand exceeds network bandwidth by priority downgrades or upgrades. Through fine-grained and precise priority assignments, Firapam can adapt well to network dynamics and further improving the performance of priority-based proposals in DCNs characterized by high throughput, low latency, data bursts, and fluctuations.

A. Firapam Overview

A high-level system diagram of Firapam is depicted in Fig. 3. Firapam spans the application layer and the transport layer, and communicates with applications above it and network or transport stacks below it. Firapam does not interfere with underlying controls and does not require any modifications to existing hardware. Instead of deploying a central entity (CE) or central system (CS) to collect global information to determine priorities, Firapam uses a distributed algorithm implemented at sending hosts to assign the priority.

Firapam maintains p_{admit} on a per-(source host, destination host, priority) basis to implement a distributed admission control algorithm, and probabilistically admits the flow on

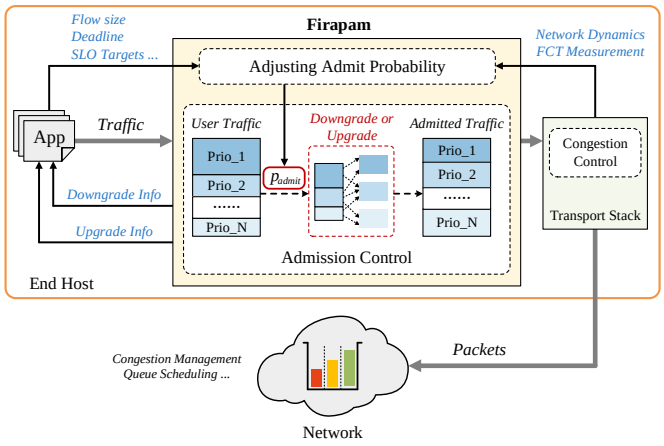


Fig. 3. Firapam system overview.

a given priority based on p_{admit}. Firapam controls p_{admit} and conducts admission control for flow priority according to Algorithm 1. The key idea behind this algorithm is simple: At each source host, for each priority class, Firapam collects the sending rate and packet loss rate of each flow to capture network status and transmission conditions. Then, it uses this information as well as application requirements as superior signals to adjust p_{admit}, thereby achieving more accurate and flexible responses.

The adjustment of p_{admit} relies on a framework based on convex optimization [34], with the goal of maximizing the network utility to ensure that multiple competing flows converge to a stable priority configuration that is fair and near-optimal. In the optimization process, Firapam employs a convex optimization algorithm based on gradient ascent to adjust priority configurations, which gradually approaches the optimal solution that maximizes the utility, thereby achieving swift reaction to changes, and fast and stable convergence.

B. Distributed Priority Admission Control of Firapam

Firapam utilizes a distributed algorithm implemented completely at sending hosts to map the expected priority to actual priority. Through regulating an admit probability p_{admit}, Firapam determines whether to admit a given flow on its requested priority (i.e., expected priority) or to downgrade/upgrade it, based on current network states and flow requirements. For unadmitted flows, Firapam issues them at a lower priority level; for flows in urgent need of network resources to meet their deadlines, Firapam issues them at their requested priority level or at a higher priority level.

1) Preparatory Work: Before implementing the admission control of Firapam, the priority of a flow is preliminarily given at the application level, called the expected priority, which can be calculated as $p_{exp} = p(\nu, \alpha)$, where threshold set α divides priority into several levels, ν is the indicator used to determine the priority. For example, if ν is the flow size, the application adopts a bytes-based PA method to set the expected priority.

However, due to the importance of a flow is determined by multiple factors and the relative importance of a flow changes from time to time, the priority set at the application level is not always accurate. Therefore, Firapam further conducts admission control for flow priority to adapt the existing PA methods to the diverse demands of traffic and the real-time state of the network. By dynamically mapping the expected priority to the actual priority, Firapam achieves the optimal priority assignment under current network states.

Algorithm 1: Admission Control of Firapam

Notation:
P_{num}: total number of priority levels;
$Downgrade\_Info$: downgrade information;
$Upgrade\_Info$: upgrade information.
Initialization:
begin
  for $i = 1, 2, \cdots, P_{num}$ **do**
    $p_{admit}[i] = 1$;
On the Issue of Flow (flow r, p_{exp})**:**
begin
  $i \leftarrow p_{exp}$;
  if $0 < p_{admit}[i] \leq 1$ **then**
    $pro\_temp \leftarrow$ random(); // Generate a random floating-point number between 0 and 1
    if $pro\_temp \leq p_{admit}[i]$ **then**
      $p_{act} = p_{exp}$;
    else
      $p_{act} = \max(p_{exp} - 1, 1)$;
      $Downgrade\_Info \leftarrow$ True;
  else if $1 < p_{admit}[i] < 2$ **then**
    $pro\_temp \leftarrow$ random();
    if $pro\_temp \leq 2 - p_{admit}[i]$ **then**
      $p_{act} = p_{exp}$;
    else
      $p_{act} = \min(p_{exp} + 1, N)$;
      $Upgrade\_Info \leftarrow$ True;
  Flow_Issue(flow r, p_{act}); // Issue this flow with the actual priority
On the Completion of Flow (B_r, δ_r^o, δ_r, l_r, p_{act})**:**
begin
  $i \leftarrow p_{act}$;
  Calculate the utility u according to Eq. (16);
  $D_{now}^R[i] \leftarrow B_r/\delta_r$; $u_{now}[i] \leftarrow u$;
  $g = (u_{now}[i] - u_{last}[i]) / (D_{now}^R[i] - D_{last}^R[i])$;
  $D_{last}^R[i] \leftarrow D_{now}^R[i]$; $u_{last}[i] \leftarrow u_{now}[i]$;
  // Update the admit probability corresponding to the current priority level
  $p_{admit}[i] = \max\{\min(p_{admit}[i] + a \cdot g, b_{up}), b_{low}\}$;

2) Firapam's Utility Framework: Firapam divides time into consecutive monitor intervals (MIs). At the end of each MI, the sending host calculates the numerical utility value of flow r according to the following utility function:

$$u(B_r, \delta_r^o, \delta_r, l_r) = \left(D^R\right)^t - bD^R(\delta_r^o - \delta_r) - cD^R l_r, \quad (16)$$

where parameters t, b, and c are constants, $0 < t < 1$, $b \geq 0$, $c > 0$. D^R is the *demand rate* of flow r, we have $D^R = B_r/\delta_r^o$, where B_r and δ_r^o represent the number of bytes and latency requirement of flow r, respectively; δ_r is the latency that flow r actually achieved; l_r is the observed packet loss rate. By default, we define B_r as the remaining flow size, δ_r^o as the deadline, and δ_r as the expected FCT or achieved FCT under current conditions (e.g., network states, priorities, and sending rate), which can be calculated as the ratio of flow size and actual sending rate of a given flow.

The utility function (Eq. (16)) is strictly convex when $t < 1$, which means that the local optimum is also the unique global optimum. Therefore, by continuously adjusting the priority configuration according to the gradient direction, the algorithm can gradually approach and ultimately achieve the goal of maximizing the network utility.

Overall, Firapam's utility function captures *demand rate* (D^R) to measure the absolute importance of a flow, as well as the actual performance (δ_r) and current network states (l_r) to measure the relative importance. Intuitively, Eq. (16) forms a reward when $\delta_r > \delta_r^o$, and is punished when l_r increases. If $\delta_r^o - \delta_r < 0$, it indicates that maintaining the current priority may cause flow r to miss its deadline. To prevent flow r from missing the deadline, it is necessary to raise the priority of flow r to increase its importance. On the contrary, if $\delta_r^o - \delta_r \geq 0$, flow r can meet its deadline by maintaining current priority and even suitably lower its priority to free up network resources for more important flows. When l_r increases, it indicates that there may be too many competitive flows in the current priority level, so issuing some of them at lower priorities is conducive to easing the conflict.

3) Translating Utility Gradients to Admit Probabilities: At the end of k-th MI, the sending host calculates the numerical utility value of MI $k-1$ and MI k according to Eq. (16), denoted by u_{k-1} and u_k, respectively. So the gradient of the utility function at MI k is:

$$g_k = \frac{u_k - u_{k-1}}{{D^R}_k - {D^R}_{k-1}}, \tag{17}$$

where ${D^R}_k$ and ${D^R}_{k-1}$ represent the *demand rate* of a flow at MI k and MI $k-1$, respectively.

Firapam utilizes the utility gradient g_k to deduce the direction and extent by which p_{admit} should be adjusted, and converts g_k into a change (i.e., step size) in p_{admit} (lines 25-33 of Algorithm 1). The admit probability in the next MI can be calculated as:

$$\begin{aligned} &p_{admit}(k+1) \\ &\quad = \max\left\{\min\left(p_{admit}(k) + a \cdot g_k, b_{up}\right), b_{low}\right\}, \end{aligned} \tag{18}$$

where $a > 0$, $0 < b_{low} < 1$, $1 < b_{up} < 2$; a denotes the conversion factor and is set to a conservative small value. When p_{admit} drops to 0, the requested priority level will not accept new flows. We set the lower bound of p_{admit}, denoted by b_{low}, to prevent starvation caused by small p_{admit}; b_{up} denotes the upper bound of p_{admit}, which prevents a large number of flows from being admitted on higher priority levels.

By converting the utility gradient into the admit probability, Firapam can equitably and effectively regulate the rate of flow issuance at different priority levels.

4) Determining the Actual Priority: For each flow, Firapam probabilistically maps its expected priority p_{exp} to the actual priority p_{act} based on the admit probability $p_{admit}[p_{exp}]$ (lines 7-24), where $p_{admit}[p_{exp}]$ represents the admit probability corresponding to priority p_{exp}. More specifically:

- If $0 < p_{admit}[p_{exp}] \leq 1$, Firapam issues this flow at its requested priority with a probability of p_{admit}; alternatively, it maps p_{exp} to the next lower priority level with a probability of $1 - p_{admit}$, i.e., $p_{act} = p_{exp} - 1$. If p_{exp} is already the lowest priority level, set $p_{act} = p_{exp}$.
- If $1 < p_{admit}[p_{exp}] < 2$, Firapam issues this flow at its requested priority with a probability of $2 - p_{admit}$; alternatively, it maps p_{exp} to the next higher priority level with a probability of $p_{admit} - 1$, i.e., $p_{act} = p_{exp} + 1$. If p_{exp} is already the highest priority level, set $p_{act} = p_{exp}$.

In the event of a flow being downgraded or upgraded, Firapam explicitly notifies the application of this information (line 16 and line 23), thus informing the application of network overload and congestion. Furthermore, the application can leverage the downgrade or upgrade information to discern critical flows and regulate the issuing of flows with higher priorities to prevent downgrades.

C. Handling Non-deadline Flows

Some applications in datacenters such as VM migration and data backup [16] generate flows devoid of strict latency requirements. These flows have no explicit deadline and can only provide limited byte-related information (e.g., flow size, remaining flow size, and bytes that have been sent), which impedes Firapam from precisely obtaining the exact *demand rate* of these flows.

To solve this challenge, we redefine certain variables in Eq. (16) for non-deadline flows. We count the maximum deadline d_{max} and its corresponding flow size B_{max} of other deadline flows within the current priority level over a period of time, and modify the utility function for non-deadline flow r as:

$$\begin{aligned} &u\left(B_r, d_{max}, \delta_r, l_r\right) \\ &\quad = \left(B_r\right)^t - bB_r\left(\frac{B_r}{\delta_r} - \frac{B_{max}}{d_{max}}\right) - cB_r l_r, \end{aligned} \tag{19}$$

where B_r is the remaining flow size, δ_r is the expected FCT or achieved FCT of non-deadline flow r.

To ensure that more deadline flows meet their deadlines, while impacting the other flows as little as possible, Eq. (19) uses $\frac{B_{max}}{d_{max}}$ to limit the sending rates of non-deadline flows. As the value of $B_r\left(\frac{B_r}{\delta_r} - \frac{B_{max}}{d_{max}}\right)$ increases, non-deadline flows tend to occupy a larger bandwidth share that potentially exceeds the maximum available bandwidth share of deadline flows, resulting in a reduction of the utility value of flow r. Additionally, aggressive control over admit probability is adopted beyond the maximum available share to quickly consume the bandwidth left over by deadline flows.

Except for the utility function, the adjustment of admit probability and determination of actual priority for non-deadline flows by Firapam are the same as for deadline flows (see Section IV-B).

D. Properties of Firapam

We demonstrate the main properties of Firapam to provide strong theoretical guarantees. We show that the assignment law of Firapam has a unique equilibrium point, it also ensures rapid convergence to equilibrium and achieves fairness in case of network perturbations or changes in traffic distribution.

Theorem 1 (Stability)**.** *Firapam's assignment law has a unique equilibrium point where the FCT of a flow converges to a fixed configuration δ^e such that $\delta^e \leq \delta^o$.*

Proof. For any flow r, let $\lambda_r = \frac{B_r}{\delta_r}$, and Eq. (16) can be rewritten as:

$$u = {\lambda_r}^t - b\left(\delta_r^o - \delta_r\right)\lambda_r - cl_r\lambda_r, \tag{20}$$

where λ_r is the actual sending rate of flow r. When $t \leq 1$, the family of utility functions in Eq. (20) falls into the category of "socially-concave" in game theory [34]. Divide both sides of the equation by $d\lambda_r$, we derive the utility dynamics as:

$$\dot{u}\left(\lambda_r\right) = \frac{du}{d\lambda_r} = t{\lambda_r}^{t-1} - b\left(\delta_r^o - \delta_r\right) - cl_r, \tag{21}$$

substitute $\lambda_r = \frac{B_r}{\delta_r}$ into (21), when $\dot{u}\left(\lambda_r\right) = 0$, we have

$$\frac{b{B_r}^{1-t}}{t}\delta_r + {\delta_r}^{1-t} = \frac{{B_r}^{1-t}}{t}\left(b\delta_r^o + cl_r\right). \tag{22}$$

Let C denote the capacity of the bottleneck link. According to the design in [35], Eq. (16) and Eq. (20) are *θ-loss-resilient* when $c = \frac{tC^{t-1}}{\theta}$, which means they do not decrease the sending rate under random packet loss rate of at most θ. So when $l_r \leq \theta$, we assume $c{B_r}^{1-t}l_r/t = 0$ to simplify analysis, and it is easy to observe that Eq. (22) has a unique equilibrium point δ_r^e such that $\dot{u}\left(\delta_r^e\right) = 0$ and $\delta_r^e \leq \delta_r^o$. □

Theorem 2 (Convergence)**.** *After a network perturbation or change in traffic distribution, Firapam's assignment law exponentially converges to equilibrium with a time constant $\frac{1}{\gamma_r} = \frac{\gamma \cdot T_{MI}}{a \cdot |g|}$, where T_{MI} is the duration of one MI.*

Proof. Let h_k be the distance of objective value δ_k at time k to optimality δ^e, and we have $h_k = h\left(\delta_k\right) = u\left(\delta_k\right) - u\left(\delta^e\right)$. η_k is the step size, γ is the condition number of utility function u such that $\gamma = \mathrm{t}$. Let $\lambda_k = \frac{B}{\delta_k}$, we have

$$\begin{aligned} h_{k+1} - h_k &= u\left(\delta_{k+1}\right) - u\left(\delta_k\right) \\ &= {\lambda_{k+1}}^t - {\lambda_k}^t - \left(b\delta^o + cl\right)\eta_k \leq -\frac{|\eta_k|}{t}h_k. \end{aligned} \tag{23}$$

According to Eq. (18), Firapam adopts a dynamic step size related to the utility gradient and converges with a period of T_{MI}, so we set $\eta_k = a \cdot g_k \cdot \gamma^*$ and obtain

$$h_{k+1} \leq h_k\left(1 - \frac{a \cdot |g_k| \cdot \gamma^*}{t}\right) \leq h_k \cdot e^{-\frac{a \cdot |g_k|}{\gamma \cdot T_{MI}}}. \tag{24}$$

Let $\gamma_r = \frac{a \cdot |g|}{\gamma \cdot T_{MI}}$ for $|g| = \frac{1}{k}\sum_{i=1}^{k}|g_k|$, we get $h_{k+1} \leq h_1 \cdot e^{-\gamma_r k}$. Therefore, Firapam's assignment law exponentially converges to equilibrium and optimality δ^e with a time constant $\frac{\gamma \cdot T_{MI}}{a \cdot |g|}$. The farther δ_k is from δ^e, the larger $|g_k|$ is, and the faster Firapam converges to an FCT close to δ^e. □

Definition 1. *The system achieves **relative fairness** if network resources are reasonably allocated to all flows to maximize their satisfaction. Therefore, instead of dividing network resources equally among all flows with diverse demands, a PA method aims to minimize the deadline miss rate: the more flows that meet their deadlines, the better the system's fairness. So the fairness of a PA method can be expressed as a minimization problem of deadline miss rate:*

$$\text{minimize } R_{DM}, \tag{25}$$

where R_{DM} is the overall deadline miss rate.

Theorem 3 (Fairness)**.** *Firapam achieves relative fairness. Its assignment law ensures the reliable allocation of network resources by minimizing the overall deadline miss rate.*

Proof. Suppose the expected FCT shifts from δ^e to δ^{new} after a network perturbation or change in traffic distribution, and takes $T = \frac{k_n}{\gamma_r}$ time to converge to a value less than the deadline, i.e., $\delta^e + \left(\delta^{new} - \delta^e\right) \cdot e^{-k_n\gamma_r} \leq \delta^o$. For flows with deadlines longer than T (account for $1 - O_k$), Firapam guarantees that they all complete within the deadline; for flows with deadlines shorter than T (account for O_k), they meet their deadlines with probability p_{admit} ($p_{admit} \leq 1$). So the overall deadline miss rate is $R_{DM} = \left(1 - p_{admit}\right) \cdot O_k$, where O_k is affected by convergence time T and latency requirement distribution $\mathbb{L}$. So the problem described in Eq. (25) can be transformed into a maximization problem of $k_n \cdot |g|$:

$$\max_{\delta^e \to \delta^{new}} k_n \cdot |g|. \tag{26}$$

Because Eq. (16) satisfies $\frac{d^2u}{d\lambda^2} < 0$ and has a maximum value $u\left(\delta^e\right)$, the maximum of $k_n \cdot |g|$ is the utility gradient g_{new} after the perturbation, i.e.,

$$\max_{\delta^e \to \delta^{new}} k_n \cdot |g| = g_{new} = \frac{u\left(\delta^{new}\right) - u\left(\delta^e\right)}{B/\delta^{new} - B/\delta^e}. \tag{27}$$

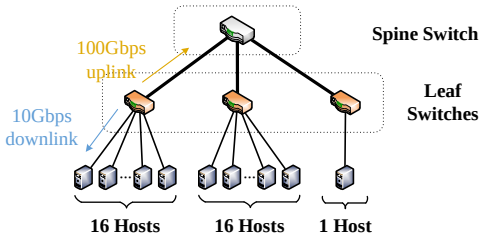
Therefore, by solving the utility gradient g_{new} and applying it to p_{admit}, Firapam minimizes the deadline miss rate after any perturbation or change, thereby achieving fairness among flows of different importance. □

On the whole, Firapam's distributed and fine-grained priority assignment algorithm can respond quickly to changes in both network traffic and requirement distribution. It prioritizes limited network resources to more important flows while avoiding starvation of less important flows to reduce the overall deadline miss rate, thus achieving relative fairness. As a result, Firapam can better support existing proposals that take advantage of the priority, which is conducive to further improving their performance. It is also more real-time and easier to deploy than existing methods that rely on a central entity (CE) to update priority assignment laws regularly.

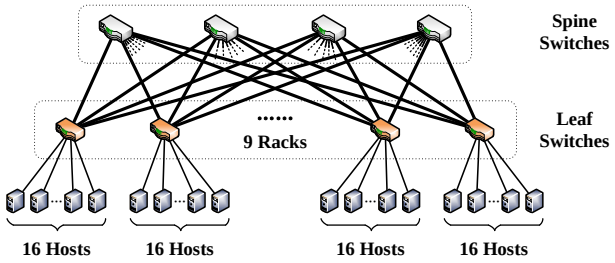
V. Evaluation

We implement Firapam in the simulation platform, which utilizes a packet-level simulator built atop YAPS [29] to realize event-driven simulations.

We conduct a comprehensive performance evaluation of Firapam and compare it with existing priority assignment



(a) Small-scale topology



(b) Large-scale topology

Fig. 4. Different scales of topologies used in the simulation.

(PA) methods. Our evaluation mainly focuses on four aspects: Firapam's gains on existing bytes-based and latency-based PAs, Firapam's adaptability to varying traffic distribution and requirement distribution, Firapam's ability to reduce the overall deadline miss rate, and Firapam's performance under high workload and bursty traffic patterns.

A. Setup

Our evaluation is based on a packet simulator YAPS [29]. We evaluate Firapam extensively in packet-level simulations.

Topology: We employ the leaf-spine topology proposed in pFabric [24], and use two different scales of topologies. The small-scale topology interconnects 33 hosts through 3 leaf switches connected to 1 spine switch in a full mesh, as depicted in Fig. 4(a). The large-scale topology consists of more nodes, including 144 hosts, 9 leaf switches, and 4 spine switches, as depicted in Fig. 4(b). Each leaf (or top-of-rack) switch has 16 10Gbps downlinks (to the hosts) and 1 100Gbps uplink (to the spine).

Traffic pattern: We adopt two traffic patterns: *all-to-all* traffic pattern where each host sends flows to other hosts with Poisson arrivals, and *incast* traffic pattern where multiple hosts simultaneously send flows to one host.

Priority class: To clearly measure the effectiveness of different PA laws, we use three priority classes in the experiment, namely priority_H, priority_M, and priority_L. Class priority_H has the highest priority and is usually associated with real-time interactive applications or key control traffic. Class priority_M is less stringent on the latency relative to class priority_H. priority_H and priority_M flows are assigned with deadlines by which they must be completely transmitted or will lose value. Class priority_L has the lowest priority and is associated with unimportant flows, which have loose latency targets or require completion only. Hence, the deadline miss rate of priority_L flows is given less emphasis.

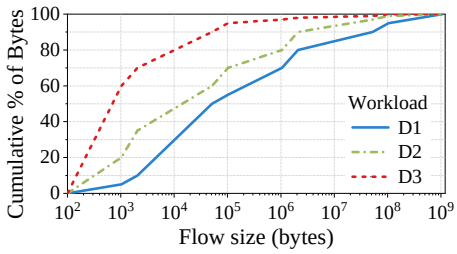


Fig. 5. Three cumulative distributions of flow size used to evaluate Firapam.

Configuration: The granularity of monitor interval (MI) depends on the type of protocol used by the scheme: for most priority-based proposals, MI is set to the round-trip time (RTT); for schemes designed based on remote procedure call (RPC) (e.g., Aequitas [12]), MI is set to the end-to-end RPC latency, which is the time between the first RPC packet arriving at the transport layer and the last RPC packet being acknowledged at the transport layer. Unless specified otherwise, we default to the settings recommended in [35]: $t = 0.9$, $b = 900$, $c = 11$ in Eq. (16) and $a = 25$ in Eq. (18).

Workload: We design three traffic distribution scenarios based on realistic DCN workloads: D1 for the workload of accessing a collection of servers at Facebook [30], D2 for the workload of Google search application [31], and D3 for the aggregated workload from all applications running in a Google datacenter [31]. Fig. 5 depicts the cumulative distributions of flow size in three scenarios. The average flow sizes for scenarios D1, D2, and D3 are 61.5MB, 17.1MB, and 1.6MB, respectively. Additionally, we set short latency requirements to account for 50%, middle latency requirements to account for 40%, and long latency requirements to account for 10% of all flows in each of the three scenarios.

B. Supporting Existing Priority Assignment Methods

We first evaluate the performance of Firapam within a small-scale topology illustrated in Fig. 4(a). We choose the PA methods of Homa [14] and PASE [11] as representatives of bytes-based and latency-based PA methods, respectively.

TABLE II
THE PRIORITY DISTRIBUTION OF DIFFERENT CASES IN THREE SCENARIOS

	Bytes-based PA			Latency-based PA		
PD(%)	**D1**	**D2**	**D3**	**D1**	**D2**	**D3**
Case1	30,50,20	55,25,20	70,20,10	50,40,10	50,40,10	50,40,10
Case2	33,34,33	33,34,33	33,34,33	33,34,33	33,34,33	33,34,33
Case3	30,20,50	20,25,55	10,20,70	10,30,60	10,30,60	10,30,60

When adopting bytes-based PA methods, smaller flows are assigned with higher priorities, so the proportion of priority_H, priority_M, and priority_L flows in three scenarios mainly depends on the flow size distribution. In contrast, when adopting latency-based PA methods, the ideal priority distribution for each scenario is consistent with its respective latency requirement distribution. However, as discussed in Section III, fluctuations in traffic distribution or requirement distribution can lead to a heavy degree of mismatch between

TABLE III
THE DEADLINE MISS RATE (DMR) OF BYTES-BASED PAS UNDER CHANGING TRAFFIC DISTRIBUTION

	Deadline miss rate (%)	D1		D2		D3	
		DMR_H	DMR_M	DMR_H	DMR_M	DMR_H	DMR_M
Case1	**PA+Firapam**	**14.32**	**19.12**	**21.8**	**12.86**	**20.78**	**5.1**
	PA+Aequitas	16.56	22.62	26.54	14.26	36.6	6.74
	PA only	23.42	38.8	47.92	19.86	48.72	7.46
Case2	**PA+Firapam**	**15.54**	**15.4**	**16.36**	**16.5**	**9.52**	**11.54**
	PA+Aequitas	18.26	17.68	19.24	18.98	14.12	16.62
	PA only	26.2	26.08	27.86	28.14	16.46	19.4
Case3	**PA+Firapam**	**14.3**	**10.34**	**10.84**	**13.42**	**2.26**	**4.82**
	PA+Aequitas	16.7	11.76	22.76	19.62	2.6	6.46
	PA only	23.22	14.62	15.64	20.86	2.76	7.14

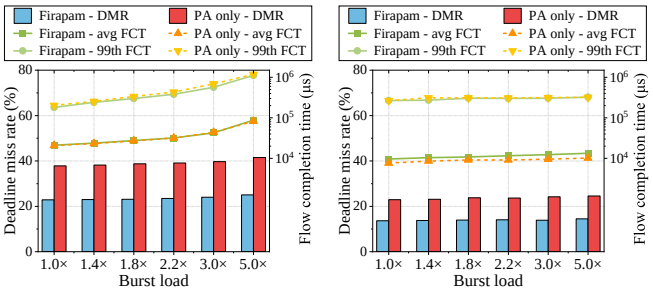
TABLE IV
THE DEADLINE MISS RATE (DMR) OF LATENCY-BASED PAS UNDER CHANGING TRAFFIC DISTRIBUTION

	Deadline miss rate (%)	D1		D2		D3	
		DMR_H	DMR_M	DMR_H	DMR_M	DMR_H	DMR_M
Case1	**PA+Firapam**	**19.84**	**16.7**	**20.94**	**17.74**	**16.62**	**12.58**
	PA+Aequitas	24.4	19.78	25.54	21.24	27.98	18.82
	PA only	39.8	31.02	43.9	33.44	33.76	21.5
Case2	**PA+Firapam**	**15.54**	**15.4**	**16.36**	**16.5**	**9.52**	**11.54**
	PA+Aequitas	18.26	17.68	19.24	18.98	14.12	16.62
	PA only	26.2	26.08	27.86	28.14	16.46	19.4
Case3	**PA+Firapam**	**6.12**	**14.16**	**5.62**	**15.1**	**2.32**	**7.92**
	PA+Aequitas	6.54	16.1	6.08	18.0	2.66	11.8
	PA only	7.16	22.88	7.14	24.88	2.76	13.46

the actual priorities of flows and their supposed importance in the network, resulting in the priority inversion problem [12], [16], [17] that undermines the benefits brought by priorities and significantly degrades network performance.

Hence, we change the priority distribution (PD) in different scenarios to reflect different mismatch levels, the proportion of $\mathrm{priority}_H$, $\mathrm{priority}_M$, and $\mathrm{priority}_L$ flows changes as shown in Table II. With an *all-to-all* traffic pattern, average $0.8\times$ workload, and dynamic $1.4\times$ burst load, the deadline miss rate (DMR) of bytes-based and latency-based PA methods, as well as their performance after the deployment of Firapam for different cases are listed in Table III and Table IV, respectively. We also choose a priority admission control method based on additive increase multiplicative decrease (AIMD) (see Aequitas [12]) for comparison, and calculate the performance of different PA methods after deploying Aequitas.

As shown in Table III and Table IV, when priority downgrade or upgrade is disabled in the PA, changes in traffic distribution or requirement distribution significantly affect the deadline miss rate of both bytes-based and latency-based PAs. After adopting the AIMD-based priority downgrading scheme, for those performance-critical flows, Aequitas reduces the DMR of $\mathrm{priority}_H$ flows (DMR_H) by 3.6%∼44.6%. Firapam's dynamic allocation strategy based on convex optimization brings further performance improvements to existing PAs. Firapam can promote efficient resource allocation and better support performance-critical flows by identifying flow importance more rapidly and precisely. As a result, Firapam successfully minimizes the deadline miss rate. It reduces DMR_H by 14.5%∼52.3% and the DMR of $\mathrm{priority}_M$ flows (DMR_M) by 38.1%∼46.9% compared



(a) Scenario D1 (b) Scenario D3

Fig. 6. The deadline miss rate and flow completion time with flow size distributions of D1 and D3 under different burst loads.

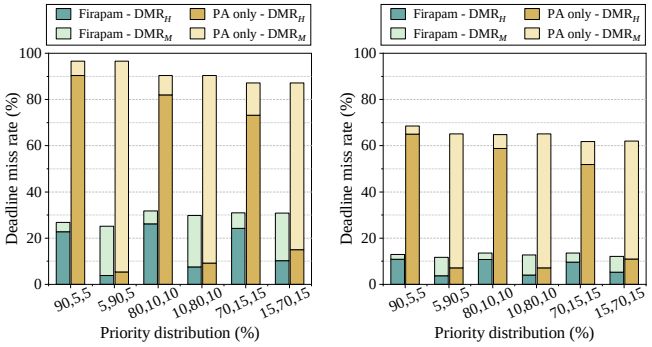
with PA with no downgrade or upgrade (*PA only*), and further reduces DMR_H by 6.4%∼40.6% and DMR_M by 12%∼33.2% compared with Aequitas. These results also indicate that a straightforward allocation of high priorities to most flows is not helpful in improving the overall DMR but increases the likelihood of flows with strict latency requirements missing their deadlines.

C. Dealing with Bursty Traffic Patterns and High Workloads

We then evaluate the performance of Firapam within a large-scale topology shown in Fig. 4(b). Under high workloads, low-priority flows may suffer from long-term starvation. To verify whether Firapam can timely capture the delays caused by network overload, congestion, and queuing, we gradually increase the burst load of the network and calculate the DMR and FCT of the strategy.

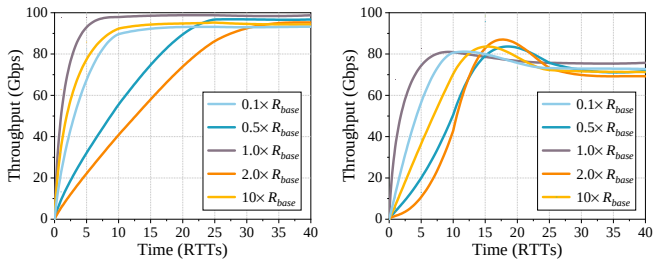
With an average $0.8\times$ workload and a latency-based PA, we adopt the flow size distributions of scenario D1 and D3, respectively, and increase the burst load from $1\times$ to $5\times$. The DMR, average FCT, and $99th$ percentile flow completion tail for different flow size distributions are illustrated in Fig. 6. As the burst load increases, the DMR of PA with and without Firapam both increase. However, it is clear that the DMR after Firapam's deployment is better than *PA only* in all cases. The traffic includes some flows that cannot be completed within the deadlines even if sent at line rate, which forms a fixed DMR that increases with the burst load. Firapam can minimize the deadline miss rate of flows other than these, so its DMR only increases linearly at a slow rate as the burst load increases. Meanwhile, Firapam reduces the DMR at the cost of slightly increasing the total FCT. Although *PA only* achieves a lower overall FCT, it only allows a subset of high-priority flows to complete quickly, and more flows will miss their deadlines due to the lack of network resources.

Subsequently, we increase the average network workload to $0.99\times$ and utilize an *incast* traffic pattern. We simulate a large number of flows with the same priority erupt simultaneously, and compare the performance of different PAs before and after Firapam's deployment. The results are presented in Fig. 7. In Fig. 7(a), we set all flows to a uniform size of 61.5MB and assign the same priority to most flows. When $\mathrm{priority}_H$ flows account for 90%, the DMR_H of PA without Firapam reaches up to 90.4%, which means that $\mathrm{priority}_H$ flows compete



(a) Bytes-based PA (b) Latency-based PA

Fig. 7. The deadline miss rate (DMR) of different PAs when incast occurs.



(a) Firapam adjustment (b) AIMD adjustment

Fig. 8. The convergence of effective throughputs under different requirements.

fiercely for resources but still fail to complete in time. After Firapam's deployment, only important flows are retained in priority_H, while less urgent flows are downgraded to lower priority levels to prevent them from seizing network resources from performance-critical flows, thereby reducing the DMR by 72.3%∼73.9%. At the same time, flows that are downgraded to priority_M or upgraded to priority_H can also achieve a lower deadline miss rate than *PA only*. In Fig. 7(b), we adopt the traffic distribution of D1 and set the same latency requirement to most flows. Similarly, Firapam reduces the DMR of *PA only* by 81.2%∼82%. Overall, when Firapam is disabled, traditional PAs struggle to effectively determine what traffic should get access to limited resources, resulting in higher deadline miss rate when a large number of same-priority flows emerge. This problem is particularly severe under incast traffic patterns.

D. Convergency

Firapam not only reacts quickly to reduce the deadline miss rate, but also features good convergency. In this section, we focus on the effective throughput at application level, which is the throughput of flows that meet their deadline requirements.

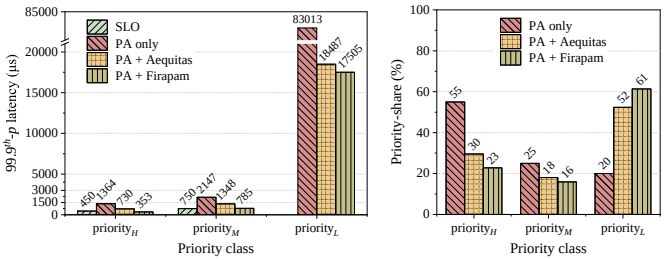
The effective throughput will converge to the neighborhood of 100Gbps when the aggregate requirement of flows fits the network capacity. We take the set of deadlines corresponding to this aggregate requirement as the baseline, represented as R_{base}. Then, we change the deadlines of priority_H and priority_M flows to simulate different degrees to which aggregate requirement exceeds network capacity, where different sets of deadlines can be expressed as different multiples of the baseline (e.g., $1.0 \times R_{base}$). With average $0.99\times$ workload and dynamic $3\times$ burst load, when priority assignment is performed in a Firapam manner and an AIMD manner (i.e., Aequitas), respectively, the convergence of effective throughputs is shown in Fig. 8.

Fig. 8(b) indicates that the convergence speed of AIMD adjustment is independent of the matching degree between network capacity and traffic requirement. And when the aggregate requirement does not fit the network capacity, its effective throughput requires about 25 RTTs to converge. Moreover, AIMD adjustment considers less on the urgency of performance-critical flows and results in more flows missing deadlines, thus only achieving the effective throughput of less than 75Gbps. On the contrary, as shown in Fig. 8(a), Firapam converges its effective throughput to more than 90Gbps, a 38.85% improvement over AIMD adjustment, and achieves a dynamic convergence speed depending on the matching degree between network capacity and traffic requirement. The greater the gap between aggregate requirement and baseline R_{base} is, the faster Firapam's effective throughput converges to equilibrium, which accords with the conclusion in Theorem 2: the convergence speed of Firapam is related to $|g|$, that is, to the gap between requirement and actual capability $|\delta^o - \delta|$.

E. Large-scale Evaluation with Production Workloads

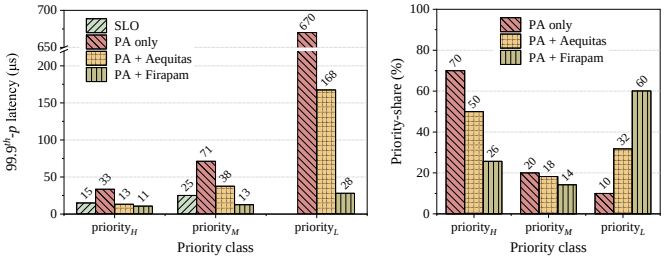
To further investigate the application potential of Firapam in real-world environments, we evaluate the performance of Firapam in the large-scale topology with several production workloads. Specifically, we adopt two production DCN workloads, W1 [32] and W2 [33], which are the Web search workload for DCTCP [32] analysis and the workload on the Hadoop cluster at Facebook, respectively. W1 is dominated by large flows, with an average flow size of 12.2MB and flows larger than 10MB account for 30%. While small flows dominate W2, of which flows with sizes ranging from 100B to 1KB account for 61%, and the average flow size of W2 is 0.41MB. We also increase the burst load (i.e., the maximum instantaneous load) on the link to $15\times$ its capacity to simulate the situation of extreme overload that can occur in real environments. To facilitate the analysis, we set the same latency requirements for flows in each priority class, represented as service level objective (SLO). Firapam's performance under workloads W1 and W2 is illustrated in Fig. 9 and Fig. 10, respectively.

Under workloads W1, Firapam reduces the tail latency in priority classes priority_H and priority_M by 74.12% and 63.44% respectively compared with *PA only*, and by 51.64% and 41.77% compared with Aequitas. Under workloads W2, Firapam reduces the tail latency in priority_H and priority_M by 66.67% and 81.69% compared with *PA only*, and by 15.38% and 65.79% compared with Aequitas. Overall, Firapam achieves a tail latency lower than the SLO or very close to the SLO, which means it can effectively meet the performance requirements of applications to achieve higher stability. Even under extreme overloads that can occur in production, Firapam still meets SLOs well. In addition, according to Fig. 9(b) and Fig. 10(b), we find that Firapam



(a) 99.9th percentile latency (b) Share of priority classes

Fig. 9. Firapam's performance in the large-scale topology with production workload W1.



(a) 99.9th percentile latency (b) Share of priority classes

Fig. 10. Firapam's performance in the large-scale topology with production workload W2.

downgrades most of the coarse-grained high priorities set at the application level to lower priority classes (e.g., priority_L) through admission control, making the actual priorities of flows better match their supposed importance in the network. As a result, although Firapam downgrades 41%∼50% of flows with higher priorities to priority_L, it still achieves the lowest deadline miss rate. Firapam reduces the overall DMR by 53.11%∼59.17% compared with *PA only*, and by 22.11%∼47.07% compared with Aequitas.

In conclusion, Firapam is effective in handling bursty traffic and incast traffic under high workloads. By providing flexible and precise admission control for flow priority and quickly converging to the fair share in the current network state, Firapam notably reduces the deadline miss rate and achieves favorable fairness compared to existing PAs.

VI. Conclusion

Up to this point, DCNs lack effective methods other than centralized control to appropriately assign priorities to flows with diverse demands in a shared environment, resulting in increasing challenges in providing precise differentiated services. In this paper, we presented Firapam, a novel fine-grained priority assignment strategy, to push the performance limits of priority-based proposals in DCNs. Firapam stands out by reacting to both current network states and flow characteristics to achieve dynamic priority assignment, thus facilitating efficient allocation of network resources and meeting diverse traffic requirements to a greater extent. We demonstrated that Firapam possesses several desirable properties, including stability, fast convergence, and fairness. Compared with existing priority assignment methods, Firapam notably reduced the deadline miss rate and displayed swift responsiveness to fluctuations in network states, making it especially well-suited for dynamic DCN environments.

Acknowledgment

This work is supported in part by the National Natural Science Foundation of China under Grants No. 62302472 and No. 62372425, the Fundamental Research Funds for the Central Universities under grant No. WK2100000039, and Youth Innovation Promotion Association of the Chinese Academy of Sciences (CAS) under Grant No. Y202093.

References

[1] V. Addanki, O. Michel, and S. Schmid, "PowerTCP: Pushing the performance limits of datacenter networks," in *Proceedings of the 19th Symposium on Networked Systems Design and Implementation (NSDI)*, pp. 51–70, USENIX, 2022.

[2] W. Wang, M. Moshref, Y. Li, G. Kumar, T. S. E. Ng, *et al.*, "Poseidon: Efficient, robust, and practical datacenter CC via deployable INT," in *Proceedings of the 20th Symposium on Networked Systems Design and Implementation (NSDI)*, pp. 255–274, USENIX, 2023.

[3] C. Wilson, H. Ballani, T. Karagiannis, and A. I. T. Rowstron, "Better never than late: Meeting deadlines in datacenter networks," in *Proceedings of the 2011 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM)*, pp. 50–61, ACM, 2011.

[4] B. Vamanan, J. Hasan, and T. N. Vijaykumar, "Deadline-aware datacenter TCP (D2TCP)," in *Proceedings of the 2012 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM)*, pp. 115–126, ACM, 2012.

[5] M. Alizadeh, A. Kabbani, T. Edsall, B. Prabhakar, A. Vahdat, and M. Yasuda, "Less is more: Trading a little bandwidth for ultra-low latency in the data center," in *Proceedings of the 9th Symposium on Networked Systems Design and Implementation (NSDI)*, pp. 253–266, USENIX, 2012.

[6] P. X. Gao, A. Narayan, S. Karandikar, J. Carreira, S. Han, *et al.*, "Network requirements for resource disaggregation," in *Proceedings of the 12th Symposium on Operating Systems Design and Implementation (OSDI)*, pp. 249–264, USENIX, 2016.

[7] G. Kumar, N. Dukkipati, K. Jang, H. M. G. Wassel, X. Wu, *et al.*, "Swift: Delay is simple and effective for congestion control in the datacenter," in *Proceedings of the 2020 Conference on Data Communication, Technologies, Architectures, and Protocols for Computer Communication (SIGCOMM)*, pp. 514–528, ACM, 2020.

[8] R. Zhou, D. Dong, S. Huang, and Y. Bai, "FastTune: Timely and precise congestion control in data center network," in *Proceedings of the 2021 Intl Conf on Parallel & Distributed Processing with Applications, Big Data & Cloud Computing, Sustainable Computing & Communications, Social Computing & Networking (ISPA/BDCloud/SocialCom/SustainCom)*, pp. 238–245, IEEE, 2021.

[9] C. Delimitrou and C. Kozyrakis, "Paragon: QoS-aware scheduling for heterogeneous datacenters," in *Proceedings of the 18th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, pp. 77–88, ACM, 2013.

[10] M. A. Qureshi, J. Yan, Y. Cheng, S. H. Yeganeh, Y. Seung, *et al.*, "Fathom: Understanding datacenter application network performance," in *Proceedings of the 2023 Conference of the ACM Special Interest Group on Data Communication (SIGCOMM)*, pp. 394–405, ACM, 2023.

[11] A. Munir, G. Baig, S. M. Irteza, I. A. Qazi, A. X. Liu, and F. R. Dogar, "Friends, not foes: Synthesizing existing transport strategies for data center networks," in *Proceedings of the 2014 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication (SIGCOMM)*, pp. 491–502, ACM, 2014.

[12] Y. Zhang, G. Kumar, N. Dukkipati, X. Wu, P. Jha, *et al.*, "Aequitas: Admission control for performance-critical RPCs in datacenters," in *Proceedings of the 2022 Conference of the ACM Special Interest Group on Data Communication (SIGCOMM)*, pp. 1–18, ACM, 2022.

[13] X. Wu, Z. Wang, W. Wang, and T. S. E. Ng, "Augmented queue: A scalable in-network abstraction for data center network sharing," in *Proceedings of the 2023 Conference of the ACM Special Interest Group on Data Communication (SIGCOMM)*, pp. 305–318, ACM, 2023.

[14] B. Montazeri, Y. Li, M. Alizadeh, and J. K. Ousterhout, "Homa: A receiver-driven low-latency transport protocol using network priorities," in *Proceedings of the 2018 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication (SIGCOMM)*, pp. 221–235, ACM, 2018.

[15] H. Rezaei and B. Vamanan, "ResQueue: A smarter datacenter flow scheduler," in *Proceedings of the 2020 Web Conference (WWW)*, pp. 2599–2605, ACM, 2020.

[16] L. Chen, K. Chen, W. Bai, and M. Alizadeh, "Scheduling mix-flows in commodity datacenters with Karuna," in *Proceedings of the 2016 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication (SIGCOMM)*, pp. 174–187, ACM, 2016.

[17] C. Hong, M. Caesar, and B. Godfrey, "Finishing flows quickly with preemptive scheduling," in *Proceedings of the 2012 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication (SIGCOMM)*, pp. 127–138, ACM, 2012.

[18] M. P. Grosvenor, M. Schwarzkopf, I. Gog, R. N. M. Watson, A. W. Moore, *et al.*, "Queues don't matter when you can JUMP them!," in *Proceedings of the 12th Symposium on Networked Systems Design and Implementation (NSDI)*, pp. 1–14, USENIX, 2015.

[19] J. Liu, J. Huang, Z. Li, Y. Li, J. Wang, and T. He, "Achieving per-flow fairness and high utilization with limited priority queues in data center," *IEEE/ACM Transactions on Networking*, vol. 30, no. 5, pp. 2374–2387, 2022.

[20] H. Chen, F. Wang, N. Helian, and G. Akanmu, "User-priority guided min-min scheduling algorithm for load balancing in cloud computing," in *Proceedings of the 2013 National Conference on Parallel Computing Technologies (PARCOMPTECH)*, pp. 1–8, IEEE, 2013.

[21] H. Zhang, J. Zhang, W. Bai, K. Chen, and M. Chowdhury, "Resilient datacenter load balancing in the wild," in *Proceedings of the 2017 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication (SIGCOMM)*, pp. 253–266, ACM, 2017.

[22] G. Shen, Q. Li, W. Shi, F. Han, Y. Jiang, and L. Gu, "Poche: A priority-based flow-aware in-network caching scheme in data center networks," *IEEE Transactions on Network and Service Management*, vol. 19, no. 4, pp. 4491–4504, 2022.

[23] S. Hu, Y. Zhu, P. Cheng, C. Guo, K. Tan, *et al.*, "Tagger: Practical PFC deadlock prevention in data center networks," *IEEE/ACM Transactions on Networking*, vol. 27, no. 2, pp. 889–902, 2019.

[24] M. Alizadeh, S. Yang, M. Sharif, S. Katti, N. McKeown, *et al.*, "pFabric: Minimal near-optimal datacenter transport," in *Proceedings of the 2013 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication (SIGCOMM)*, pp. 435–446, ACM, 2013.

[25] W. Bai, L. Chen, K. Chen, D. Han, C. Tian, and H. Wang, "PIAS: Practical information-agnostic flow scheduling for commodity data centers," *IEEE/ACM Transactions on Networking*, vol. 25, no. 4, pp. 1954–1967, 2017.

[26] M. Chowdhury and I. Stoica, "Efficient coflow scheduling without prior knowledge," in *Proceedings of the 2015 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication (SIGCOMM)*, pp. 393–406, ACM, 2015.

[27] L. Chen, J. Lingys, K. Chen, and F. Liu, "AuTO: Scaling deep reinforcement learning for datacenter-scale automatic traffic optimization," in *Proceedings of the 2018 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication (SIGCOMM)*, pp. 191–205, ACM, 2018.

[28] D. Zats, T. Das, P. Mohan, D. Borthakur, and R. H. Katz, "DeTail: Reducing the flow completion time tail in datacenter networks," in *Proceedings of the 2012 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication (SIGCOMM)*, pp. 139–150, ACM, 2012.

[29] G. Kumar, A. Narayan, and P. Gao, "YAPS: Yet another packet simulator," *https://github.com/NetSys/simulator.*, 2016. Accessed on Jun., 2024.

[30] B. Atikoglu, Y. Xu, E. Frachtenberg, S. Jiang, and M. Paleczny, "Workload analysis of a large-scale key-value store," in *Proceedings of the 12th ACM SIGMETRICS/PERFORMANCE Joint International Conference on Measurement and Modeling of Computer Systems (SIGMETRICS)*, pp. 53–64, ACM, 2012.

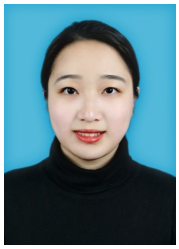
[31] R. Sivaram, "Some measured google flow sizes," 2008. Google internal memo, available on request.

[32] M. Alizadeh, A. G. Greenberg, D. A. Maltz, J. Padhye, P. Patel, *et al.*, "Data center TCP (DCTCP)," in *Proceedings of the 24th ACM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM)*, pp. 63–74, ACM, 2010.

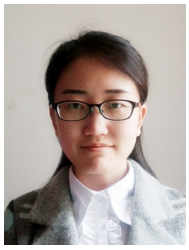
[33] A. Roy, H. Zeng, J. Bagga, G. Porter, and A. C. Snoeren, "Inside the social network's (datacenter) network," in *Proceedings of the 29th ACM Conference on Special Interest Group on Data Communication (SIGCOMM)*, pp. 123–137, ACM, 2015.

[34] E. Even-dar, Y. Mansour, and U. Nadav, "On the convergence of regret minimization dynamics in concave games," in *Proceedings of the 41st Annual ACM Symposium on Theory of Computing (STOC)*, pp. 523–532, ACM, 2009.

[35] M. Dong, T. Meng, D. Zarchy, E. Arslan, Y. Gilad, *et al.*, "PCC Vivace: Online-learning congestion control," in *Proceedings of the 15th Symposium on Networked Systems Design and Implementation (NSDI)*, pp. 343–356, USENIX, 2018.



Rui Zhuang received her B.S. degree from the Department of Information Engineering, Hefei University of Technology (HFUT), in July, 2019. She is currently working toward the Ph.D degree in the School of Cyber Science and Technology, University of Science and Technology of China (USTC). Her research interests include future Internet architecture design, transmission optimization and datacenter network.



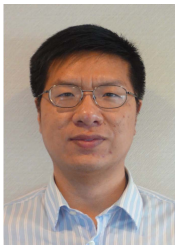
Jiangping Han (M'22) received her bachelor's degree and the Ph.D. degree both from the Department of Electronic Engineering and Information Science (EEIS), USTC, in 2016 and 2021, respectively. She is currently a Post-Doctoral fellow with the School of Cyber Science and Technology, USTC. From Nov. 2019 to Oct. 2021, She was a visiting scholar with the School of Computing, Informatics, and Decision Systems Engineering, Arizona State University. She was a Post-Doctoral researcher with the School of Cyber Science and Technology, USTC, in 2021-2023. She is currently an associate researcher with the School of Cyber Science and Technology, USTC. Her research interests include future Internet architecture design and transmission optimization.

Kaiping Xue (M'09-SM'15) received his bachelor's degree from the Department of Information Security, University of Science and Technology of China (USTC), in 2003 and received his doctor's degree from the Department of Electronic Engineering and Information Science (EEIS), USTC, in 2007. From May 2012 to May 2013, he was a postdoctoral researcher with the Department of Electrical and Computer Engineering, University of Florida. Currently, he is a Professor in the School of Cyber Science and Technology, USTC. He is also the director of Network and Information Center, USTC. His research interests include next-generation Internet architecture design, transmission optimization and network security. His work won best paper awards in IEEE MSN 2017 and IEEE HotICN 2019, the Best Paper Honorable Mention in ACM CCS 2022, the Best Paper Runner-Up Award in IEEE MASS 2018, and the best track paper in MSN 2020. He serves on the Editorial Board of several journals, including the IEEE Transactions on Information Forensics & Security (TIFS), the IEEE Transactions on Dependable and Secure Computing (TDSC), the IEEE Transactions on Wireless Communications (TWC), and the IEEE Transactions on Network and Service Management (TNSM). He has also served as a (Lead) Guest Editor for many reputed journals/magazines, including IEEE Journal on Selected Areas in Communications (JSAC), IEEE Communications Magazine, and IEEE Network. He is an IET Fellow and an IEEE Senior Member.

This article has been accepted for publication in IEEE Transactions on Network Science and Engineering. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TNSE.2024.3429402

Jian Li (M'20-SM'23) received his bachelor's degree from the Department of Electronics and Information Engineering, Anhui University, in 2015, and received doctor's degree from the Department of Electronic Engineering and Information Science (EEIS), University of Science and Technology of China (USTC), in 2020. From Nov. 2019 to Nov. 2020, he was a visiting scholar with the Department of Electronic and Computer Engineering, University of Florida. From Dec. 2020 to Dec. 2022, he was a Post-Doctoral researcher with the School of Cyber Science and Technology, USTC. He is currently an associate researcher with the School of Cyber Science and Technology, USTC. He serves as an Editor of China Communications. His research interests include quantum networking, wireless networks, and next-generation Internet architecture.

David S.L. Wei (SM'07) received his Ph.D. degree in Computer and Information Science from the University of Pennsylvania in 1991. From May 1993 to August 1997 he was on the Faculty of Computer Science and Engineering at the University of Aizu, Japan (as an Associate Professor and then a Professor). He has authored and co-authored more than 140 technical papers in various archival journals and conference proceedings. He is currently a Professor with the Computer and Information Science Department at Fordham University. He was a lead guest editor or a guest editor for several special issues in the IEEE Journal on Selected Areas in Communications, the IEEE Transactions on Cloud Computing, and the IEEE Transactions on Big Data. He also served as an Associate Editor of IEEE Transactions on Cloud Computing, 2014-2018, an editor of IEEE J-SAC for the Series on Network Softwarization & Enablers, 2018 – 2020, and an Associate Editor of Journal of Circuits, Systems and Computers, 2013-2018. Dr. Wei is the recipient of IEEE Region 1 Technological Innovation Award (Academic), 2020, for contributions to information security in wireless and satellite communications and cyber-physical systems. He is a member of ACM and AAAS, and is a life senior member of IEEE, IEEE Computer Society, and IEEE Communications Society. Currently, Dr. Wei focuses his research efforts on cloud and edge computing, cybersecurity, and quantum computing and communications.



Ruidong Li (SM'07) received a bachelor in engineering from Zhejiang University, China, in 2001, and received a doctorate of engineering from the University of Tsukuba in 2008. He is an associate professor in College of Science and Engineering, Kanazawa University, Japan. Before joining Kanazawa University, he was an a senior researcher with the Network System Research Institute, National Institute of Information and Communications Technology (NICT). He is the founder and chair for the IEEE SIG on big data intelligent networking and IEEE SIG on intelligent Internet edge, and the secretary of IEEE Internet Technical Committee. He also serves as the chairs for conferences and workshops, such as IWQoS 2021, MSN 2020, BRAINS 2020, ICC 2021 NMIC symposium, ICCN 2019/2020, NMIC 2019/2020 and organized the special issues for the leading magazines and journals, such as IEEE Communications Magazine, IEEE Network, IEEE IEEE Transactions on Network Science and Engineering (TNSE), and etc.. His current research interests include future networks, big data networking, blockchain, information-centric network, internet of things, network security, wireless networks, and quantum Internet. He is a senior member of the IEEE and a member of IEICE.

Qibin Sun (F'11) received the Ph.D. degree from the Department of Electronic Engineering and Information Science (EEIS), University of Science and Technology of China (USTC), in 1997. He is currently a professor in the School of Cyber Science and Technology, USTC. His research interests include multimedia security, network intelligence and security, and so on. He has published more than 120 papers in international journals and conferences. He is a fellow of IEEE.

Jun Lu received his bachelor's degree from southeast university in 1985 and his master's degree from the Department of Electronic Engineering and Information Science (EEIS), University of Science and Technology of China (USTC), in 1988. Currently, he is a professor in the School of Cyber Science and Technology and the Department of EEIS, USTC. He is also the president of Jiaxing University. His research interests include theoretical research and system development in the field of integrated electronic information systems, network and information security. He is an Academician of the Chinese Academy of Engineering (CAE).