Computer Networks 270 (2025) 111565

Contents lists available at ScienceDirect

Computer Networks

journal homepage: www.elsevier.com/locate/comnet

APCC: Active Precise Congestion Control for Campus Wireless Networks

Yongqi Yang [a], Qianyi Huang [b], Yixue Liu [a], Xiaojie Huang [a], Liang Zhang [c], Jiaxin Tian [c], Li Wang [a], Chen Tian [a,*], Wanchun Dou [a], Guihai Chen [a]

[a] State Key Laboratory for Novel Software Technology, Nanjing University, China
[b] Sun Yat-sen University, China
[c] Huawei, China

ARTICLE INFO

Keywords:
campus wireless networks
Congestion control
Modify AP only

ABSTRACT

Campus wireless networks are the most important last-mile networks, which are widely deployed for organizations like universities, large companies and factories. Emerging applications, such as video conferencing or immersive teaching, require the network to provide large bandwidth, low latency and reliable services. However, current campus wireless networks fall short of these requirements. Frequent fluctuations in wireless bandwidth cause a rate mismatch between the wireless and wired segments, resulting in long buffer queues that ultimately lead to packet loss and high latency. Therefore, a scheme for efficiently controlling the queue length in campus wireless networks is in need. However, this is challenging as it is infeasible to modify the protocol stacks of the end devices. In this paper, we propose APCC (Active Precise Congestion Control for Campus Wireless Networks), which performs congestion control at the access point (AP) APCC calculates the target rate for each flow and performs congestion control through modifying the receive window field in the TCP header of the ACK packet. In this way, the AP can actively and precisely control the flow rate before congestion occurs. Evaluation illustrates that APCC maintains a short queue length while making full use of the network bandwidth. When wireless link capacity fluctuates, APCC achieves an 6× shorter queue length and 4× lower packet latency than CUBIC for the 95th percentile. Compared to Zhuge (Meng et al., 2022) and P-Scheduler (Hoiland-Jorgensen et al., 2017), APCC reduces the 95th percentile of network latency by 52% and 22%, respectively, and decreases queue length by 71% and 51%, respectively. We believe that APCC presents a new paradigm for congestion control in campus wireless networks.

1. Introduction

Campus wireless networks are the most important last-mile network, providing Internet access to organizations such as universities, hospitals, and factories. According to a recent report [1], the market size of global campus wireless networks is increasing at a compound annual growth rate of 20%–30%. Emerging applications, such as immersive teaching, remote surgery, flexible manufacturing, require the campus wireless networks to provide large bandwidth, low latency and reliable services. However, most of today's widely deployed TCP congestion control algorithms, such as NewReno [2] and CUBIC [3], fall short of these requirements in campus wireless networks. In order to take full advantage of link capacity, these protocols gradually increase the congestion window. Although it can improve the network throughput, frequent fluctuations in wireless bandwidth cause a rate mismatch between the wireless and wired segments, leading to the formation of long buffer queues. This problem is further exacerbated by greedy TCP flows, as both factors contribute to packet accumulation, packet loss, and high latency. It tends to exhaust the buffer space, which results in packet loss and large latency, which are detrimental for many applications in campus wireless networks, such as production line automation and remote operations of machines.

Our goal is to achieve low packet loss rate and low latency while maintaining high throughput. It should satisfy two requirements. The first requirement is to enable congestion control without any modification to client devices and in-network switches. The second requirement is to optimize end-to-end latency without sacrificing bandwidth utilization. Explicit feedback schemes (e.g., [4–7]) require full upgrade of in-network switches, which is costly and not practical. Some methods (e.g., [8]) require modification on end-hosts, which is unpractical as the end-hosts (e.g., smartphones, laptop) are usually controlled by end users, which violate the first requirement in Table 1. Besides, AQM (Active Queue Management) schemes (e.g., [9–11]) avoid congestion by proactively drop packets before buffer overflow occurs. This decreases

* Corresponding author.
E-mail address: tianchen@nju.edu.cn (C. Tian).

https://doi.org/10.1016/j.comnet.2025.111565
Received 22 November 2024; Received in revised form 19 June 2025; Accepted 16 July 2025
Available online 25 July 2025

Table 1
Performance comparison of existing works and APCC.

Property \ Scheme	APCC	DRWA	P-Scheduler	Zhuge
Modified position	Modify AP only	Modify client devices only	Modify AP only	Modify AP only
Modified signal	rwnd	rwnd	Scheduling the packet directly	Delay ack
Queue	✓	–	✓	✓
RTT	✓	–	✓	×
Throughput	✓	–	×	✓
R1[a]	✓	×	✓	✓
R2[b]	✓	✓	×	×

[a] R1 (Requirement 1): Enable congestion control without any modification to client devices and in-network switches.
[b] R2 (Requirement 2): Optimize end-to-end latency without sacrificing bandwidth utilization.

network service reliability and often result in sub-optimal bandwidth utilization. These methods do not fulfill the second requirement in Table 1.

We find that frequent and large fluctuations in wireless bandwidth lead to rate mismatch between wireless and wired networks, and thus AP often become bottlenecks in campus wireless networks. By performing congestion control at the AP, the AP can control the data rate of each flow and thus the network can strike a balance between high throughput and short queue length. Compared with user devices, the network administrator has full control of the AP in the campus wireless networks and thus we can easily upgrade the APs to incorporate this idea.

Although the idea sounds straightforward, there are several challenges to be tackled. The first challenge is how to ensure that congestion control signals can bypass the wireless bottleneck. In traditional end-to-end congestion control, signals generated at the wireless segment may be significantly delayed due to queuing and retransmissions, causing the sender to react slowly to bandwidth changes. Zhuge [12] proposes to delay ACKs at the access point to signal congestion. However, this approach essentially increases the end-to-end latency intentionally, which is undesirable for latency-sensitive applications. The second challenge is how to determine an appropriate receive window size when using the AP to perform congestion control. Since delaying ACKs introduces additional latency and is not ideal, a preliminary alternative is for the AP to directly modify the receive window field in the TCP header of ACK packets. This method allows the AP to control the sender's rate without increasing end-to-end delay. However, the key issue lies in selecting a suitable window size: simply reducing the window size does slow down the sender, but it also hurts throughput and degrades bandwidth utilization. Therefore, it remains challenging to determine a window size that strikes a balance between maintaining good throughput and achieving effective congestion control. The third challenge is how to guarantee fairness among multiple flows while considering the dynamics of the traffic flows. If we evenly distribute the bandwidth among the flows, when a connection is newly established, the connection is in the slow start phase and thus it cannot fully utilize the allocated bandwidth. Uniform distribution may lead to throughput degradation.

In this paper, we propose APCC (Active Precise Congestion Control for Campus Wireless Network), which refers to Access Point Based Congestion Control. APCC regulates each data flow by modifying the receive window field in the ACK packet. In TCP, the sender uses the minimum of the cwnd (congestion window) and the rwnd (receive window) as its send window. To determine an appropriate window size, APCC calculates the target rate according to the link capacity and the current buffer size. Then, the AP determines how to allocate the target rate to each flow. APCC proposes a way to convert the traffic rate into the receive window size. Recognizing the diverse bandwidth requirements of different flows, APCC introduces a paradigm that empowers campus wireless networks administrators to allocate target rates based on their management strategy. APCC suggests utilizing flow-related indicators to guide bandwidth allocation (e.g., RSSI or data rate of the wireless link). In this way, APCC can actively and precisely control the traffic rate before congestion occurs. Table 1 summarizes the differences between APCC and existing works. We can see that APCC is the one that satisfies all the design requirements with any performance degradation.

We use NS-3 simulations to evaluate the performance of APCC. Evaluation results show that APCC maintains a shorter queue length while making full use of bandwidth. When facing significant fluctuations in wireless link capacity, APCC achieves an 6× shorter queue length and 4× lower packet latency than CUBIC for the 95th percentile. Moreover, APCC still has a large amount of buffer space available and thus can avoid packet loss in bursty scenarios. Compared to Zhuge, APCC reduces the 95th percentile of network latency by 18% and decreases queue length by 70%. Compared to P-Scheduler [13], APCC has no bandwidth under utilization phenomenon and faster throughput recovery when the wireless link capacity experiences fluctuations. We also consider the mobility of client devices. We emulate the random movement of mobile devices in a 10×10 m^2 rectangular area, centering around the AP. Compared to Zhuge [12], APCC reduces the 95th percentile of network latency by 52% and decreases queue length by 71%. Compared to P-Scheduler [13], APCC reduces the 95th percentile of network latency by 22% and decreases queue length by 51%.

The contributions of this paper can be summarized as follows:

1. We propose APCC, an AP-based TCP congestion control scheme that enables congestion control signals to bypass the wireless bottleneck. Compared to Zhuge [12], APCC achieves timely congestion feedback at a much lower cost.
2. APCC is an AP-based TCP congestion control scheme that only requires modifications at access points in campus wireless networks, avoiding changes to end devices or in-network switches/routers. This makes it practical for deployment in campus wireless networks.
3. APCC can perform per-flow rate control by modifying the receive window in ACK packets, helping to achieve a better balance among throughput, latency, and fairness.

2. Background and motivation

2.1. Packet loss in campus wireless networks

A typical campus network topology is illustrated in Fig. 1. The network includes the access layer, the convergence layer, and the core layer. It has become a trend for the last hop of campus networks to adopt wireless technologies (e.g., Wi-Fi).

In such a campus network architecture, the wireless hop between the AP and the client is usually the bottleneck. Due to multi-user access and mutual interference, wireless networks often experience significant fluctuations in available bandwidth. When the wireless link capacity decreases, if the network's traffic remains at a high level, it may lead to buffer overflow at the AP, resulting in packet loss. We simulate this phenomenon using NS-3 [14], configuring the TCP congestion control algorithm to NewReno. Further details on the simulation configuration can be found in Section 4.1. The results are presented in Fig. 2.

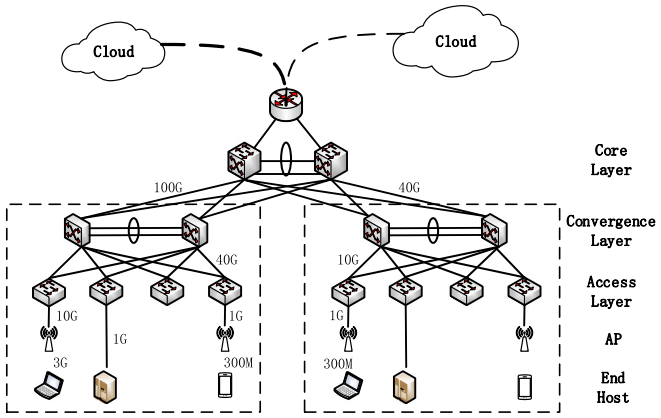


Fig. 1. A typical campus wireless network topology.

At the 5th second, there was a sudden drop in bandwidth. In such situations, queues rapidly form at the AP, resulting in significant end-to-end latency. Moreover, TCP is greedy in nature, leading to a gradual increase in traffic, and the packets will continue to queue in the AP buffer until they are eventually dropped. This process may repeat continuously, resulting in two negative consequences. Firstly, packet loss not only affects the reliability of network services but also wastes network bandwidth. Secondly, a significant number of packets are waiting in the buffer, causing queuing delays of tens or hundreds of milliseconds, leading to high average delays. Therefore, this is unacceptable for delay-sensitive applications such as remote surgery, necessitating a suitable solution to reduce average latency and control the queue length in the campus wireless networks.

2.2. Limitations of state-of-the-art transport protocols

2.2.1. End-to-end congestion control

Traditional end-to-end congestion control schemes, such as NewReno [2] and CUBIC [3], rely on packet loss events to determine whether the network is congested and reduce the sending rate accordingly. When the available bandwidth fluctuates on the wireless link, or when there is a mismatch between the wired and wireless segment rates, queues can quickly build up at the AP's relatively small buffer, resulting in bufferbloat. This can significantly increase the round-trip delay of data packets [13,15]. Among them, the BBR [16] congestion control algorithm proposed by Google is the most representative, which has been gradually deployed on Linux machines. In order to detect the available bandwidth of bottleneck links in the network, BBR makes the source send packets at a rate exceeding the bottleneck capacity, which will cause the packets to accumulate in the buffer. It makes BBR greatly improved compared to the algorithm based on packet loss, but still causes a certain queue accumulation. Although we configured a lower level of wireless bandwidth fluctuation in Fig. 3 compared to Fig. 2, BBR continues to exhibit persistent queue accumulation. The simulation results shown in Fig. 3(a) confirm this phenomenon.

It is important to clarify that the version of BBR evaluated in our experiments is BBRv1, which is not only widely deployed in practice but also the only version currently available in NS-3. Although BBR is designed to control queue length and minimize queuing delay—as discussed in Gomez et al. [17]—our simulation results clearly show that, in scenarios with wireless bandwidth fluctuations or mismatches between wired and wireless segment rates, BBRv1 still leads to noticeable queue buildup at the AP buffer. This observation is consistent with previous findings in the literature [18,19].

While BBRv1 offers significant improvements over traditional loss-based algorithms such as Cubic and NewReno by reducing queue buildup, our experiments show that it can only alleviate, but not fully eliminate, bufferbloat under dynamic wireless conditions. The queue length at the AP with BBRv1 is generally smaller than with Cubic and NewReno, but still remains non-negligible, leading to increased latency. BBRv2 introduces packet loss and ECN signals to improve coexistence and fairness; however, recent studies indicate that it still faces convergence and fairness issues, particularly in scenarios with multiple competing flows or in environments with bandwidth fluctuations and packet loss [17,18]. BBRv3 was developed to address these limitations and further optimize performance. Nevertheless, the literature shows that even BBRv3 can suffer from performance degradation or slow recovery under extreme network conditions, such as high loss rates or severe bandwidth fluctuations. Systematic evaluations of BBRv3 in complex scenarios are still ongoing, and some experiments have identified room for improvement in its bandwidth estimation accuracy and RTT fairness.

Although many studies have proposed improvements for BBR in variable environments, challenges remain due to its reliance on stable bottleneck assumptions. As a result, BBR may still experience queue accumulation during sudden bandwidth changes or rate mismatches. This is consistent with the broader literature, which shows that even the latest versions of BBR, while more robust, are not immune to the challenges posed by dynamic environments.

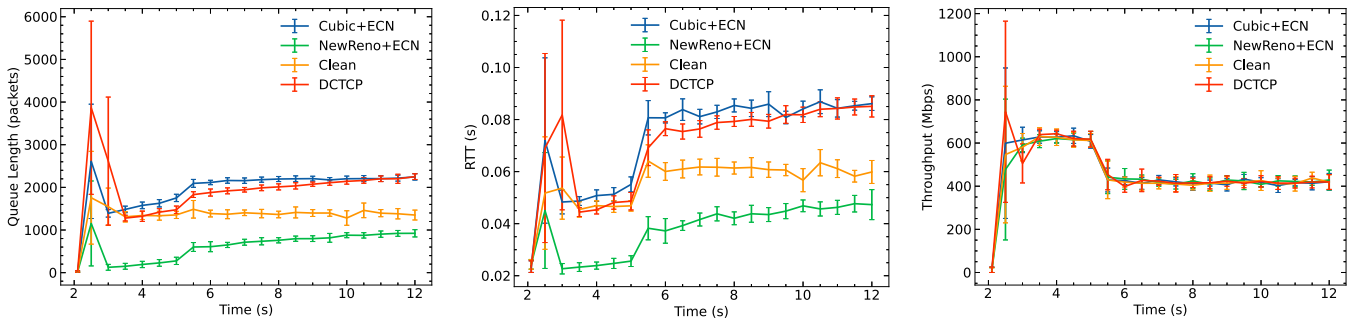
In summary, both our results and prior work indicate that while BBR reduces queue buildup compared to loss-based algorithms, it does not completely solve the problem, especially under variable wireless conditions.

2.2.2. AQM (Active Queue Management) schemes and ECN

AQM schemes such as RED [9], CoDel [10] and some other schemes [11] can drop some packets before the buffer is overflowed to notify the source of potential network congestion, so as to control the congestion in advance. However, packet loss will hurt the reliability of the network services. To overcome this drawback, these AQM solutions can enable ECN (Explicit Congestion Notification), using markers instead of packet loss events to notify the source of congestion. ECN has been widely used in many congestion control protocols, especially in data center networks, such as DCTCP [20], D2TCP [21], DCQCN [22], and some other strategies [23–25]. However, how to set the appropriate threshold of ECN is a widely discussed issue [26,27]. As an ECN-based end-to-end congestion control scheme, DCTCP is widely deployed in data centers. However, in the congestion avoidance phase, DCTCP increases the window blindly. This makes it difficult for DCTCP to quickly converge to an appropriate rate when the network condition fluctuates. Moreover, it can be found that DCTCP still accumulates a certain number of packets in the buffer. CLEAN [28], based on the ECN-proxy, employs the DCTCP algorithm to calculate the window size during congestion, and leverages the CUBIC algorithm for fast recovery. While CLEAN can promptly maintain a short queue on the AP, it exhibits suboptimal convergence and suffers from high tail delay. More notably, the ECN mechanism requires the joint support of end devices and intermediate equipments. It means that we cannot directly utilize the ECN mechanism in campus wireless networks, as end devices are not under the control of the network administrator.

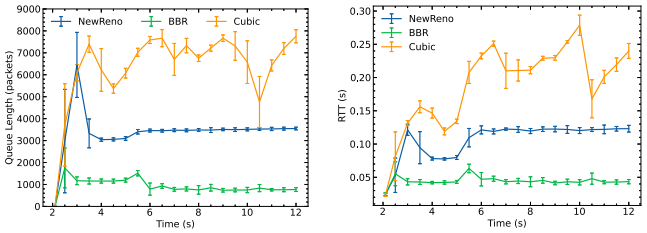
2.2.3. Explicit congestion control

Among the existing research work on congestion control, the explicit control schemes, such as XCP [4] and RCP [29], have drawn more attention. In these work, the switch uses each packet to provide multi-bit feedback to the sender based on direct knowledge of the link capacity. By telling the senders explicitly how to increase or decrease their rate, explicit congestion control schemes can lead to quick convergence of link capacity. However, XCP and RCP require significant modifications to packet formats, switches, and end devices. While these modifications are technically feasible, they introduce significant deployment challenges.



(a) Queue length (b) RTT (c) Throughput

Fig. 2. Performance variation for NewReno with ECN, CUBIC with ECN, Clean and DCTCP.



(a) Queue (b) RTT

Fig. 3. Queue Length and RTT Variation under Different Congestion Control Algorithms (CUBIC, NewReno, BBRv1).

2.2.4. Representative feedback and queue management schemes for wireless links

Zhuge [12] enables the AP to rapidly provide feedback on the dynamics of wireless links, so Zhuge has excellent performance for RTC (Real-Time Communication) applications. However, Zhuge does not perform satisfactorily in terms of average delay since Zhuge focuses on reducing tail delay and shortens the control loop by actively increasing the RTT (delay ack). Jiang [30] modifies terminals instead of APs and requires tcp timestamp to be enabled to estimate RTT, which is not accurate even when the DRS [31] algorithm is reused to estimate RTT. This causes some trouble for large scale deployments. Toke Høiland-Jørgensen's work [13] is the most advanced work on Bufferbloat. Most Wi-Fi access points based on the linux kernel already use this packet scheduler by default. Toke Høiland-Jørgensen et al. [13] integrated fq-codel into the linux kernel, which solves the bufferbloat very well. However, our simulations show that when the capacity of the wireless link experiences large fluctuations, flows may experience suboptimal throughput and the speed of throughput recovery is not satisfactory.

3. APCC design

3.1. APCC overview

Unlike other protocols that perform congestion control after the queue has been established, APCC actively controls the flow rate to match the link capacity, and thus avoiding queue accumulation.

Frequent and large fluctuations in wireless bandwidth often lead to rate mismatches between wireless and wired networks, so access points often become bottlenecks in campus wireless networks and a large portion of the congestion in the network occurs on the wireless links. It is worth noting that the AP can obtain the link throughput of the wireless links and observe the queue length at its buffer. Therefore, the AP can take advantage of this information to perform accurate target rate calculations. Furthermore, the AP can explicitly tell the senders how to control the send rate. Inspired by VCC [32,33] and DRWA [30], in APCC, the AP regulates the congestion control by modifying the rwnd (receive window) field in the header of ACK packets, and APCC designs a more efficient algorithm for modifying the window. After receiving the ACK packet, the source will set the send window size to be min(rwnd, cwnd). As long as rwnd is smaller than cwnd (congestion window), the send window size will be rwnd. In TCP, when the receiving buffer at the destination is insufficient, the destination will notify the sender via the receive window size. In APCC, if the original receive window size is smaller, the AP will not modify the rwnd field.

Specifically, the workflow of APCC is as follows (See Algorithm 3.1): First, APCC calculates a target rate of the AP according to the link capacity and the current buffer queue length. This enqueue rate can avoid queue accumulation when the AP forwards packets at full capacity. Then, the rate is distributed among multiple flows passing through the AP. The AP's strategy for distributing the rate can be designed with any indicators of the flow, since the APCC has already calculated the optimal target rate. This rate value is converted into a window size and filled in the receive window field of the ACK packet sent to the source. APCC maintains a flow table on the AP, recording the status of each flow, such as throughput, window size, RTT, etc. After receiving the returned ACK, the source sets its send window size to be the minimum one of the receive window size and the congestion window size.

3.2. Precise rate calculation

AP will calculate the target rate by the following formula:

$$targetRate(t) = \eta B - \max\left[\frac{Q(t) - K}{\delta}, 0\right], \tag{1}$$

where B is the link capacity, $Q(t)$ is the queue length at time t, K is the queue length threshold, η is a constant close to 1 (such as 0.95), and δ is a time unit constant. In IEEE 802.11 wireless networks, the AP can obtain information such as the Channel State Information (CSI) of the physical layer, thus being able to roughly estimate the link capacity. [34] also provides a method for estimating the wireless link capacity on the AP.

The first term in the right hand side of Eq. (1) refers to the dequeue rate when the AP forwards packets at full capacity. If the current queue length is low, that is, when $Q(t) < K$, APCC will set the target rate to ηB, where η is slightly less than 1. This is the experience from previous work [35–37] that by setting the target rate slightly lower than the link capacity, we can trade a small amount of bandwidth for a large reduction in latency. The second term is the capacity retained for draining the existing queues in the buffer, reducing the queue length from $Q(t)$ to K after δ time unit. When $Q(t) > K$, the second term becomes non-zero. Their difference is the available capacity for incoming flows, i.e, the target rate.

Algorithm 3.1: APCC Workflow

Input: Link capacity B, current buffer queue length $Q(t)$, set of active flows $\mathcal{F}$

Step 1: Calculate Target Rate
Compute the target enqueue rate $targetRate(t)$ for the AP based on link capacity B and current buffer queue length $Q(t)$;
$targetRate(t) \leftarrow \text{TargetRate}(B, Q(t))$;

Step 2: Distribute Rate Among Flows
foreach *flow* $f \in \mathcal{F}$ **do**
  Assign a portion r_f of $targetRate(t)$ to flow f according to the AP's scheduling strategy;

Step 3: Convert Rate to Window Size
foreach *flow* $f \in \mathcal{F}$ **do**
  Estimate RTT for flow f: RTT_f;
  $w_f \leftarrow \max\left(2, \left\lfloor \frac{r_f \times RTT_f}{MSS} \right\rfloor\right)$;
  Minimum window = 2, see Section 3.4. Fill w_f into the receive window field of the ACK packet sent to the source of flow f;

Step 4: Maintain Flow Table
Update the flow table on the AP to record the status of each flow;
foreach *flow* $f \in \mathcal{F}$ **do**
  Record throughput, window size, RTT, expiration time, and other relevant metrics for f;

Step 5: Source Adjusts Send Window
foreach *flow* $f \in \mathcal{F}$ **do**
  Upon receiving the ACK, source sets its send window size W_{send} as:;
  $W_{\text{send}} \leftarrow \min(w_f, cwnd_f)$;

The reason to set the queue length threshold K in APCC is that we need to ensure that the target rate does not overreact to small increases in queue length. The wireless link often schedules packets in batches [38]. The AP needs to aggregate data packets for batch scheduling, and thus it will cause the queues to accumulate to a certain extent before forwarding them. However, this increase in queue length does not mean network congestion. In addition, the phenomenon of micro-bursts in the network also occurs from time to time [39]. Micro-bursts will cause the queue length to increase instantaneously, and then be drained quickly, which also does not mean network congestion. Therefore, in order to prevent such phenomena from decreasing the target rate, it is necessary to set a queue length threshold. We propose to configure K to be larger than the queue growth caused by the AP's average batch scheduling time.

3.3. Per-flow rate control

After the AP determines the target rate, it needs to allocate this rate to each active flow. By modifying their receive window values, the AP can perform congestion control. This process is called active congestion control in this paper.

3.3.1. RTT estimation

Accurately estimating the delay for each data packet which have not yet been transmitted in a wireless network poses challenges. The shared nature of wireless networks leads to competition for wireless channel resources and frequent bandwidth fluctuations. Once the queue becomes long, the sender may not promptly receive congestion signals, preventing timely rate adjustments and exacerbating queue congestion. To consider the dynamic nature of the wireless link, we can refer to Zhuge's [12] scheme and decouples the latency into three parts, as shown in the following equation:

$$delay = qLong + qShort + tx.$$

$qLong$ [12] is defined as the delay from the time of packet arrival to the time when the packet goes to the head of the queue. $qShort$ [12] refers to the waiting delay when the packet is at the front of the queue, due to factors including MAC layer aggregation, "listening before talk" period, and random backoff when collision happens. tx represents the transmission delay, which is related to the wireless physical layer data rate. With the scheme Zhuge [12], we can accurately estimate the RTT even for dynamic wireless links.

3.3.2. Rate-to-window conversion

After calculating the target rate, APCC can assign rates to N active flows according to any indicators of flows. pHost [40], ExpressPass [41] suggests simply averaging the target rate for N active flows, or we can utilize flow-related indicators to guide bandwidth allocation (e.g., according to the estimated RTT as a metric).

$$targetRate_i = targetRate(t) \times \frac{RTT_i}{RTT_N}. \tag{2}$$

where RTT_i is the RTT value for flow i, $targetRate_i$ is the target rate that each flow should achieve, RTT_N is the total estimated RTT. Note that we set the minimum value of rwnd to be 2 packets and we will explain it later.

The above allocation scheme is presented as one possible approach; alternative metrics related to flows can also be used for rate allocation. In this work, we use pHost [40] as an example for evaluation. We evaluate the effects of their distribution scheme in the next section and discuss fairness.

3.4. Active flow statistics

Fairness among flows is a crucial consideration for every congestion control algorithm. Before APCC allocates bandwidth, the AP first checks the status of each flow to determine whether the send window of the flow can be taken over by APCC . In other words, it is necessary to judge whether the traffic rate of the flow can reach the share allocated to it. Since the AP regulates congestion control in APCC, a flow table needs to be maintained to store information about each flow, such as window size, RTT, etc. In order to identify whether the flow is in the idle period, APCC adds an entry, expiration time, to the flow table. If a flow has no data arriving at the AP within a certain time threshold, it is considered that the flow no longer exists or is in an idle state. Then the entry is deleted. The expiration time is set to be the RTT value of the flow. During the rate-to-window conversion process of APCC, the window size of the flow is guaranteed to be at least 2, that is, at least

2 packets can be sent in one RTT. Therefore, if a flow has no data to send within a RTT time, it is considered that the flow has been in an inactive state. After getting the number of active flows, APCC executes the algorithm set by the network administrator to allocate the target rate among the active flows.

4. Evaluation

4.1. Setup

In NS-3, we build a typical campus network topology according to Fig. 1. The topology is setup according to our experience in real-world campus network deployment and is supposed to emulate the real campus network architecture to the greatest extent. A large server is used to simulate the cloud server outside the campus and sends traffic into the campus network. The link bandwidth between the server and the campus network core switch is 100 Gbps; the link bandwidth between the core switch and the aggregation switch is 40 Gbps, and the link bandwidth between the aggregation switch and the access switch is 10 Gbps. There is an AP connected to the access switch with a 1 Gbps link. In the experiment, the wireless protocol standard is set to 802.11ax. All switches and APs have a 12 MB buffer. As we mentioned in 3.2, we propose to configure K to be larger than the queue growth caused by the AP's average batch scheduling time. We set the queue threshold K of APCC to 20 packets and the time parameter δ to 80 ms.

4.2. Performance

4.2.1. Queue length, throughput, and latency

The term "queue length" specifically refers to the length of the queue in the buffer of the access point. This metric is a critical indicator of the congestion level in the network, as it reflects the number of packets waiting to be transmitted through the access point. Monitoring and controlling the queue length is therefore essential for effective congestion control in the network. We compare APCC with a number of existing congestion control algorithms:

(a) NewReno [2], CUBIC [3], and BBR [16].
(b) NewReno+ECN, CUBIC+ECN: enabling the ECN mechanism on NewReno and CUBIC. We replace packet loss with packet marking, so that the sender can be notified of network congestion before packet loss occurs.
(c) DCTCP [20]: the most popular congestion control algorithm in data center, capable of maintaining short queues while achieving high throughput.
(d) Clean: Clean [28] is similar to APCC. Both perform congestion control by modifying the receive window of ACK packets. Clean enables ECN for the switch, and determines the window value by estimating the congestion level of the network according to the ECN flag.
(e) Zhuge: Zhuge [12], a state-of-the-art solution, introduces significant improvements in reducing tail delay in RTC. It achieves this by leveraging predictive techniques to estimate RTT and strategically delaying ACK messages.
(f) Packet-Scheduler [13]: Toke Høiland-Jørgensen's work is deployed on the ath9k driver for the linux kernel. Their work on bufferbloat is excellent. Most linux-based wifi APs already use this packet scheduler by default.

APCC does not need to be compared to DRWA [30], which modifies endpoints and requires TCP timestamp to be enabled, which creates challenges for large-scale deployments.

4.2.2. Simulation of bandwidth degradation

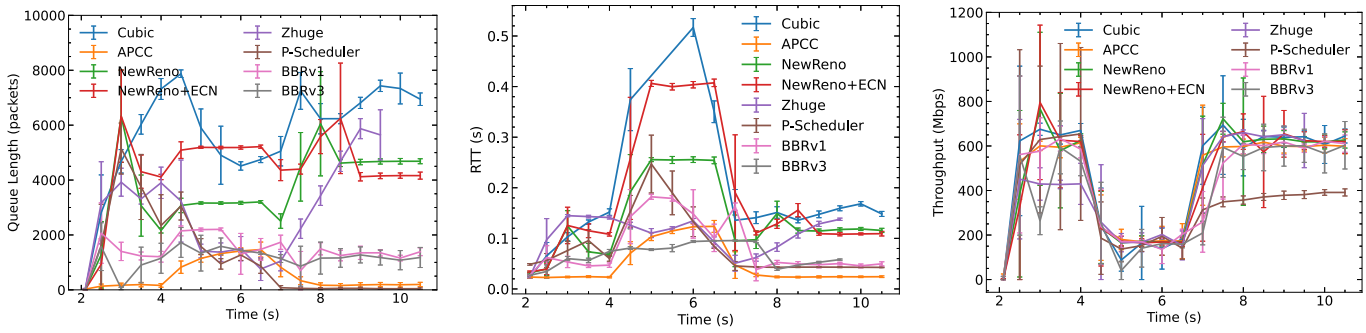
For NewReno and CUBIC with ECN support, the simulation results are shown in Fig. 2. Ecn-enabled algorithms have smaller average delay and queue length. When ECN support is added to them, although the delay is close to base RTT, it brings the problem of insufficient bandwidth utilization. The reason is that, to deal with network congestion, ECN-based solutions rely on ECN feedback to gradually and iteratively change the traffic rate, and finally converge to a stable state. How to set the ECN marking threshold has been recognized as a complex issue [27]. Clean maintains a low queue state while ensuring high throughput. However, the ECN mechanism requires the joint support of end devices and intermediate equipments. Although it is prone to packet loss, CUBIC achieves nearly full bandwidth utilization. According to [42], the algorithm based on packet loss has such a consequence: if the size of the buffer exceeds the BDP of the connection, periodic packet loss caused by the buffer overflow will not lead to a decrease in TCP throughput. Moreover, the queue length of Clean has a large variation, incurring spikes from time to time.

To demonstrate the performance of APCC, we designed a scenario with a drastic bandwidth drop. At the 4th second, the bandwidth decreased from 600 Mbps to 180 Mbps, and after a period of time, it increased from 180 Mbps to 600 Mbps again. The simulation results are shown in Fig. 4. Under such extreme packet loss and interference conditions, the Cubic algorithm exhibits a significantly increased RTT of 400 ms. Compared to CUBIC, APCC achieves a 73% reduction in the peak delay, and compared to NewReno, the peak delay is reduced by 58%. Moreover, APCC only requires upgrading the APs in campus wireless networks, thereby avoiding modifications to client devices or the need for whole-network device upgrades.

In comparison to CUBIC and NewReno, BBRv1 typically results in a shorter queue length at the access point; however, the queueing delay remains non-negligible, ultimately leading to increased latency. Overall, APCC achieves more favorable performance in these challenging scenarios. This is because BBR can only mitigate, but not completely eliminate, bufferbloat under dynamic wireless conditions. Recent studies [17–19] have shown that BBR still faces several limitations, particularly under extreme network conditions such as high packet loss rates or severe bandwidth fluctuations. Additionally, its convergence and fairness issues also remain significant concerns.

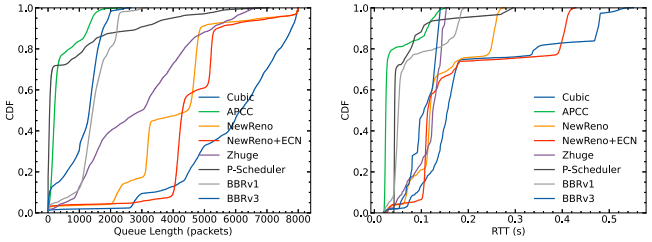
In Fig. 4, we further incorporate several representative state-of-the-art algorithms into our simulation. Specifically, we focus on Zhuge [12] and P-Scheduler [13], both of which do not require modifications to the end devices protocol stack or ECN support. For comparison, we also include traditional algorithms that demonstrate relatively strong performance, such as NewReno+ECN. The simulation results are presented in Figs. 4 and 5. Both Zhuge and P-Scheduler represent the state-of-the-art research in this area, and our analysis primarily focuses on their performance. When facing significant fluctuations in wireless link capacity, APCC exhibits a 18% reduction in the 95th percentile of network latency and a 70% reduction in queue length compared to Zhuge. Additionally, APCC shows an 12% reduction in the 95th percentile network latency and a 49% reduction in queue length compared to P-Scheduler [13]. The simulation results indicate that Zhuge has significantly higher average delay and maintains the queue length at a relatively higher level compared to APCC and P-Scheduler [13]. Although APCC and P-Scheduler have similar performance in reducing queue growth, APCC has no bandwidth under utilization phenomenon.

This is because Zhuge [12] delays sending ACKs to the source, resulting in an increase in RTT and preventing the direct reduction of the window size. Therefore, this is equivalent to sacrificing a small portion of the RTT to shorten the control loop. Even after the bandwidth reduction at 4 s, Zhuge continues to result in elevated RTT, and the RTT does not decrease even after the bandwidth recovers. Zhuge adopts window control based on the variations in RTT and maintains the window size unchanged when the RTT does not rise relative to its previous value. P-Scheduler [13] integrates Fq-Codel into



(a) Queue length (b) RTT (c) Throughput

Fig. 4. Performance comparison of APCC and other congestion control protocols when the bandwidth drops.



(a) Queue (b) RTT

Fig. 5. CDF analysis of RTT and Queue when wireless link capacity fluctuates significantly.

linux kernel, which is an excellent work to solve bufferbloat in WiFi. Our simulation results show that P-Scheduler schedules packets directly when the queue is piled up to a large size, and the stacked packets in the queue are emptied quickly. However, we can observe in Fig. 4 that when P-Scheduler starts working, it causes the traffic to appear under utilization and it will take some time to recover the throughput. In contrast to P-Scheduler, APCC does not suffer from under utilization and the queue length is maintained in a tolerable range.

4.2.3. Simulation of client device mobility

To comprehensively analyze APCC's performance on dynamic wireless links, we conduct simulations with mobile clients using NS3. We set up two mobile clients in a 10×10 m^2 area centering around the AP. The clients are configured to move randomly in this area and the wireless channel model is configured to be RandomPropagationLossModel. Under this setup, we compare the queue length, RTT, and throughput of APCC, Zhuge and P-Scheduler. They have similar bottleneck bandwidth in simulation experiments, as shown in Fig. 6(c), and they achieve nearly 100% bandwidth utilization.

However, as shown in Figs. 6 and 7, APCC shows a shorter queue length and delay. This is because APCC performs per-flow rate control by directly assigning a target rate for each flow. Compared to Zhuge [12], APCC reduces the 95th percentile of network latency by 52% and decreases queue length by 71%. As Zhuge delays ACK after detecting congestion events, it naturally increases the total delay. Rather than modifying the generic queueing layer, P-Scheduler [13] bypass it completely, and instead incorporate the smart queue management directly into the 802.11 protocol-specific subsystem. Compared to P-Scheduler [13], APCC reduces the 95th percentile of network latency by 22% and decreases queue length by 51%. The results are shown in Fig. 7. As Toke Høiland-Jørgensen discussed, the main drawback of doing this is a loss of flexibility. With this design, there is no longer a way to turn off the smart queue management completely; and it does add some overhead to the packet processing. This overhead may have some impact on the performance of P-Scheduler [13] when the wireless link capacity experiences fluctuations.

In the case that the client moves outside the serving range of the current AP and connects to another AP, APCC will be treated it as a newly joined flow at the new AP.

4.2.4. Parameter setting analysis

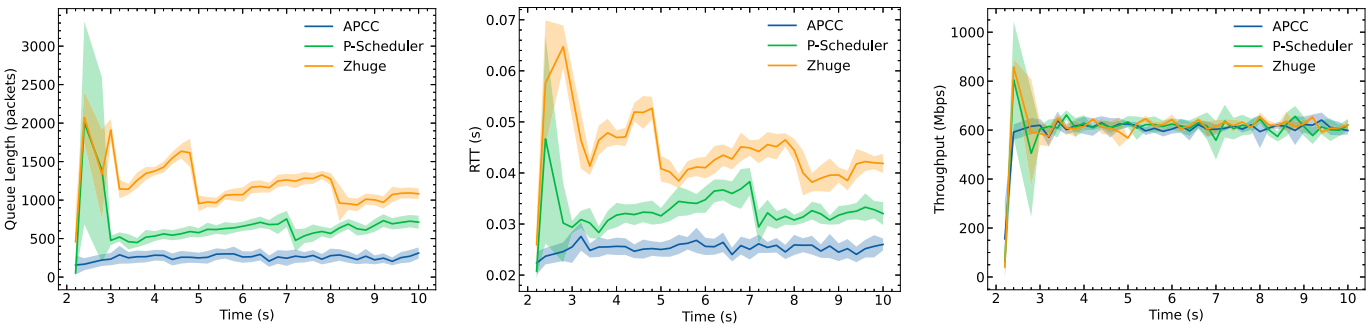
In Eq. (1), there are two parameters, the queue length threshold K and time constant δ. In this part, we test the throughput and latency performance of APCC with different values of K and δ. The results are shown in Fig. 8. The labels in the figure represent parameter pairs (K, δ). For Fig. 8, the closer the parameter pair is to the upper left corner, or it shows a trend towards the upper left corner, the better the performance of the algorithm.

We connect the results with same K with dotted lines. It shows that as δ increases, the delay first decreases and then increases rapidly, while the bandwidth utilization increases with δ to a certain extent and then decreases slightly. According to Eq. (1), when δ increases, the target rate increases, so is the bandwidth utilization. δ represents the time required to reduce the current queue length to the queue length threshold. If it is set too small, it means that APCC will greatly reduce the rate in order to drain the queue quickly, which may result in bandwidth under utilization. Moreover, this will destabilize the system, causing an increase in latency. If the setting is too large, it means that APCC reduces the flow rate by a small amount in order to drain the queue slowly. It is possible that the packet sending rate of the source end still exceeds the link capacity at this time, resulting in the continuous accumulation of queues and bringing about consequences of increased delay. As for the queue length threshold K, it can be observed that as K increases, the curve tends to move towards the right part of the figure. This is straightforward, as the increase of K means that the length of the queue in the stable state increases, and thus the average delay and the tail delay increase.

It can be inferred from the figure that when the parameter K is set to 20 packets and the parameter δ is set to 80 ms, the performance of APCC is optimal and resides in the upper left corner of the figure. In terms of parameter setting for APCC, it is suggested that a larger value for δ results in better stability of congestion control and higher bandwidth utilization, while a smaller value for δ results in faster response to congestion and lower delay. For a typical campus network environment, it is recommended to set δ to be 2~4 times of the RTT. As for the parameter K, it is recommended to set it as small as possible, provided that the queuing delay exceeds the time required for the AP to perform packet aggregation and batch forwarding.

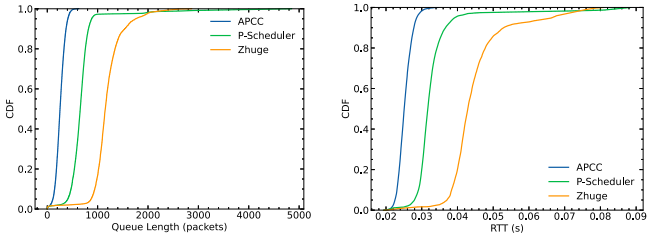
4.2.5. Fairness and convergence

To test whether APCC can quickly converge to the fair share, the experiment starts a single flow at the beginning and then adds a new flow every 5s. The result is shown in Fig. 9. It indicates that even when there are frequent AP switching events due to user movement, by



(a) Queue length (b) RTT (c) Throughput

Fig. 6. Performance comparison of APCC, P-Scheduler and Zhuge.



(a) Queue (b) RTT

Fig. 7. CDF analysis of RTT and Queue for APCC, Zhuge [12] and P-Scheduler [13].

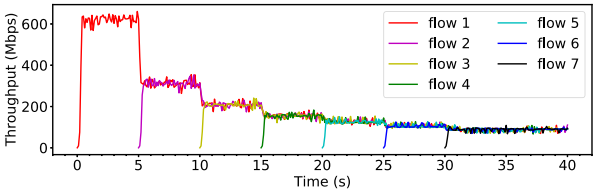


Fig. 9. Throughput for each flow over time.

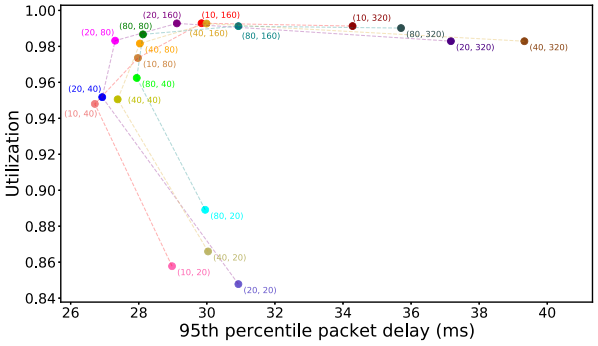


Fig. 8. The throughput and delay performance of APCC under different parameter pairs (K, δ).

treating each new connection as a new stream, APCC ensures that the congestion control algorithm is fair and efficient among all connections. This helps to maintain optimal network performance in scenarios with frequent AP switching.

5. Discussion

5.1. APCC overhead

In APCC, the AP needs to perform per-flow rate control. We may be concerned about the memory overhead and computational overhead at the AP. As for the memory overhead, for each flow, it takes 20 bytes to store per-flow information, including the flow ID, window size, RTT, and expiration time. According to our observation, there could be about tens of thousands concurrent flows. It requires 0.2–2 MB to store the per-flow state table. Given that the AP in the campus wireless networks usually has hundreds of MB memory, it has sufficient memory to store the state table. As for the computational overhead, the AP only needs to perform some simple calculations, the computational overhead is negligible. Therefore, the memory and computational overhead at the AP is affordable.

5.2. Does APCC violate the end-to-end principle?

While the end-to-end principle has long been a fundamental guideline in Internet architecture, it is not strictly followed in real-world network environments. Middleboxes such as NAT, QoS, and traffic shaping devices have been widely deployed to enhance network performance, security, and manageability. The use of such in-network mechanisms for traffic scheduling and optimization has become commonplace, and these practices can be seen as valuable supplements to the end-to-end principle rather than violations.

APCC is a pragmatic optimization scheme designed specifically for scenarios where end hosts are not easily controlled and protocol upgrades are difficult—such as in campus wireless networks. Its primary goal is to improve bandwidth utilization and fairness for the majority of users. In typical campus environments, the main challenges stem from wireless segment congestion and the lack of coordination between end hosts. APCC effectively addresses these issues, significantly enhancing the experience for most users. For special network zones like Science DMZs [43], which are dedicated to high-performance scientific data transfers, APCC can be flexibly bypassed or not deployed, thereby ensuring that extremely high bandwidth demands remain unaffected. As demonstrated by recent research published in SIGCOMM by Meng [12] et al. the academic community is actively exploring the feasibility and rationality of using in-network nodes to optimize end-to-end performance for emerging applications such as real-time communications (RTC). These developments indicate that the end-to-end principle is continuously evolving and being refined in response to modern network requirements.

In summary, the design philosophy of APCC aligns with the current academic trend of exploring in-network optimizations for end-to-end performance. As an optional network optimization mechanism, APCC

is compatible with existing bandwidth management approaches and provides network administrators with greater flexibility in optimizing network performance.

6. Related work

The congestion control protocol of the WAN is developing very rapidly, from the earlier Reno [44] to NewReno [2], HighSpeed TCP [45], BIC [46] and CUBIC [3]. These protocols work hard to improve TCP performance on paths with high BDP. However, since these congestion control protocols are based on packet loss, they do not care about the length of queues in the buffer. In campus wireless networks, the buffer size at the access point (AP) is limited. Therefore, packet loss-based congestion control mechanisms are not well suited for such environments. Some schemes based on explicit feedback are proposed, including ECN [8], XCP [4], VCP [5], and RCP [7]. These schemes require extensive changes to packet headers, switches, and client-side protocol stacks, which pose significant deployment challenges. This is more challenging for campus wireless networks, where client devices are not under the control of the network administrator.

In addition to traditional loss-based and explicit feedback-based congestion control protocols, BBR [16] and its subsequent versions (BBRv2 and BBRv3) have been proposed as modern congestion control algorithm. These protocols control the sending rate by estimating the bottleneck bandwidth and round-trip time, aiming to minimize queueing delay and bufferbloat. Although BBRv1 can significantly reduce queue buildup compared to loss-based algorithms such as Cubic and NewReno, it cannot fully eliminate bufferbloat, especially under dynamic wireless conditions or when there is a rate mismatch between wired and wireless segments. This observation is consistent with previous studies [17–19]. BBRv2 and BBRv3 incorporate packet loss and ECN signals to further improve fairness and performance. However, recent studies have shown that BBRv2 and BBRv3 still face challenges in convergence, fairness, and bandwidth estimation accuracy, particularly in complex or highly dynamic network environments [17–19]. Systematic evaluations of these newer versions are still ongoing.

To solve this problem, in this paper, we use the receive window modification mechanism to control the traffic rate. The earlier literature related to this area includes [47,48]. They modify the receive window in the header of TCP ACK returning to the sender. The calculation of the window size is crude and cannot maintain the switch queue at a stable length. They have also been found to be vulnerable to packet loss and has poor compatibility with some TCP transport protocols. The recent literature related to the use of the receive window is VCC [32,33], which decreases the receive window to force the guest to send fewer packets in the hypervisor. However, VCC is used in multi-tenant data centers, and it is not suitable for campus wireless networks. More importantly, how to determine the receive window size is not considered in VCC. DRWA [30] is to modify the window on the mobile ends, but it requires both the end and the access point to have the tcp timestamp option enabled in order to estimate the RTT. And its algorithm for calculating the window is not accurate enough to quickly keep up with changes in the capacity of the wireless link.

Pilosof [49] first presented a similar solution. However, Pilosof's scheme was not specifically tailored to the campus wireless networks. The crucial difference between APCC and Pilosof's solution lies in the control algorithm, where APCC uses AP's link information to accurately calculate the rate. As a result, APCC's window size calculation is more precise, and the design of APCC does not require any modifications to the end-devices in this process. P-Scheduler [13] is the most advanced work on Bufferbloat. Most Wi-Fi access points based on the linux kernel already use this packet scheduler by default. Toke Høiland-Jørgensen et al. [13] integrated fq-codel into the linux kernel, which solves the bufferbloat very well. However, our simulations show that when the capacity of the wireless link experiences large fluctuations, flows may experience suboptimal throughput and the speed of throughput recovery is not satisfactory.

Zhuge [12] is the state-of-the-art congestion control scheme that is designed for wireless networks. Zhuge enables the AP to rapidly provide feedback on the dynamics of wireless links, and thus Zhuge has excellent performance for RTC (Real-Time Communication) applications. However, Zhuge requires empirical experience to decide the congestion indicator (RTT threshold). When deployed in campus wireless networks, it requires the network administrator to manually configure the RTT threshold. Besides, Zhuge performs congestion control by delaying the acknowledgments, it naturally increases the end-to-end latency and thus has unsatisfactory performance on average latency. Compared to Zhuge, APCC uses the queue length as a threshold (it could simply be proportional to AP's buffer size), and thus APCC can effectively control the queue length and average latency.

7. Conclusion

In this paper, we propose APCC, a scheme for active and precise congestion control in campus wireless networks. Unlike other protocols that perform congestion control after the queue has been established, APCC actively calculates the target rate in the AP and converts it into the window size of each flow, fills it in the receive window field of the returned ACK. Evaluation results show that APCC maintains a short queue length while fully utilizing bandwidth, resulting in ultra-low latency. When facing significant fluctuations in wireless link capacity, APCC achieves an 6× shorter queue length and 4× lower packet latency than CUBIC for the 95th percentile. Compared to Zhuge [12], APCC reduces the 95th percentile of network latency by 18% and decreases queue length by 70%. Compared to P-Scheduler [13], APCC has no bandwidth underutilization phenomenon and faster throughput recovery when the radio link capacity experiences fluctuations. APCC also has fast convergence and almost complete fairness. We believe that APCC presents a new paradigm for congestion control in campus wireless networks.

CRediT authorship contribution statement

Yongqi Yang: Writing – review & editing, Writing – original draft, Visualization, Validation, Supervision, Software, Resources, Project administration, Methodology, Investigation, Formal analysis, Data curation, Conceptualization. **Qianyi Huang:** Writing – review & editing, Visualization, Supervision, Software, Resources, Project administration, Methodology, Investigation, Formal analysis, Data curation, Conceptualization. **Yixue Liu:** Visualization, Validation, Resources, Project administration, Methodology, Investigation, Formal analysis, Data curation, Conceptualization. **Xiaojie Huang:** Validation, Supervision, Software, Project administration, Methodology, Investigation, Formal analysis, Data curation, Conceptualization. **Liang Zhang:** Resources, Project administration, Investigation, Funding acquisition, Formal analysis, Conceptualization. **Jiaxin Tian:** Resources, Project administration, Investigation, Funding acquisition, Formal analysis, Data curation, Conceptualization. **Li Wang:** Visualization, Resources, Project administration, Funding acquisition, Formal analysis. **Chen Tian:** Validation, Supervision, Project administration, Methodology, Funding acquisition, Formal analysis, Data curation, Conceptualization. **Wanchun Dou:** Supervision, Project administration, Funding acquisition. **Guihai Chen:** Supervision, Resources, Project administration, Methodology, Funding acquisition.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

The authors would like to thank anonymous reviewers for their valuable comments. This research is supported by the Key Project of Jiangsu Province fundamental Research Program under Grant number BK20243053, the National Natural Science Foundation of China under Grant Numbers 62325205, 62072228, and 62172204, the Fundamental Research Funds for the Central Universities, China, the Collaborative Innovation Center of Novel Software Technology and Industrialization, and the Jiangsu Innovation and Entrepreneurship (Shuangchuang) Program.

Data availability

Data will be made available on request.

References

[1] Q. Research, Global Campus Network Market Size, Trends & Growth Opportunity, Tech. Rep., QualiKet Research, 2024, URL https://qualiketresearch.com/reports-details/Campus-Network-Market.
[2] S. Floyd, RFC 6582: The NewReno Modification to TCP's Fast Recovery Algorithm, IETF, 2012, URL https://www.rfc-editor.org/rfc/rfc6582.
[3] S. Ha, I. Rhee, L. Xu, CUBIC: A new TCP-friendly high-speed TCP variant, ACM SIGOPS Oper. Syst. Rev. 42 (5) (2008) 64–74.
[4] D. Katabi, M. Handley, C. Rohrs, Congestion control for high bandwidth-delay product networks, in: Proceedings of the 2002 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications, 2002, pp. 89–102.
[5] Y. Xia, L. Subramanian, I. Stoica, S. Kalyanaraman, One more bit is enough, in: Proceedings of the 2005 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications, 2005, pp. 37–48.
[6] P. Goyal, A. Agarwal, R. Netravali, M. Alizadeh, H. Balakrishnan, ABC: A simple explicit congestion controller for wireless networks, in: 17th USENIX Symposium on Networked Systems Design and Implementation, NSDI 20, 2020, pp. 353–372.
[7] N. Dukkipati, Rate Control Protocol (RCP): Congestion Control to Make Flows Complete Quickly, Citeseer, 2008.
[8] K. Ramakrishnan, S. Floyd, D. Black, RFC 3168: The Addition of Explicit Congestion Notification (ECN) To IP, IETF, 2001, URL https://www.rfc-editor.org/rfc/rfc3168.
[9] S. Floyd, V. Jacobson, Random early detection gateways for congestion avoidance, IEEE/ACM Trans. Netw. 1 (4) (1993) 397–413.
[10] K. Nichols, V. Jacobson, Controlling queue delay, Commun. ACM 55 (7) (2012) 42–50.
[11] S. Muhammad, T.J. Chaudhery, Y. Noh, Study on performance of AQM schemes over TCP variants in different network environments, IET Commun. 15 (1) (2021) 93–111.
[12] Z. Meng, Y. Guo, C. Sun, B. Wang, J. Sherry, H.H. Liu, M. Xu, Achieving consistent low latency for wireless real-time communications with the shortest control loop, in: Proceedings of the ACM SIGCOMM 2022 Conference, 2022, pp. 193–206.
[13] T. Hoiland-Jorgensen, M. Kazior, D. Taht, P. Hurtig, A. Brunstrom, Ending the anomaly: Achieving low latency and airtime fairness in WiFi, in: 2017 USENIX Annual Technical Conference, USENIX ATC 17, 2017, pp. 139–151.
[14] Network simulator 3, 2021, URL https://www.nsnam.org/.
[15] N. Khademi, AP buffer sizing in IEEE 802.11 WLANs, in: 2012 IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks, WoWMoM, 2012, pp. 1–3, http://dx.doi.org/10.1109/WoWMoM.2012.6263738.
[16] N. Cardwell, Y. Cheng, C.S. Gunn, S.H. Yeganeh, V. Jacobson, Bbr: Congestion-based congestion control: Measuring bottleneck bandwidth and round-trip propagation time, Queue 14 (5) (2016) 20–53.
[17] J. Gomez, E.F. Kfoury, J. Crichigno, G. Srivastava, Evaluating TCP BBRv3 performance in wired broadband networks, Comput. Commun. 222 (2024) 198–208.
[18] M. Ahsan, S.S. Muhammad, TCP BBR-n: Increased throughput for wireless-AC networks, PLoS One 18 (12) (2023) e0295576.
[19] S. Scherrer, M. Legner, A. Perrig, S. Schmid, Model-based insights on the performance, fairness, and stability of BBR, in: Proceedings of the 22nd ACM Internet Measurement Conference, 2022, pp. 519–537.
[20] M. Alizadeh, A. Greenberg, D.A. Maltz, J. Padhye, P. Patel, B. Prabhakar, S. Sengupta, M. Sridharan, Data center TCP (DCTCP), in: Proceedings of the ACM SIGCOMM 2010 Conference, 2010, pp. 63–74.
[21] B. Vamanan, J. Hasan, T. Vijaykumar, Deadline-aware datacenter TCP (D2TCP), ACM SIGCOMM Comput. Commun. Rev. 42 (4) (2012) 115–126.
[22] Y. Zhu, H. Eran, D. Firestone, C. Guo, M. Lipshteyn, Y. Liron, J. Padhye, S. Raindel, M.H. Yahia, M. Zhang, Congestion control for large-scale RDMA deployments, ACM SIGCOMM Comput. Commun. Rev. 45 (4) (2015) 523–536.
[23] A. Munir, I.A. Qazi, Z.A. Uzmi, A. Mushtaq, S.N. Ismail, M.S. Iqbal, B. Khan, Minimizing flow completion times in data centers, in: 2013 Proceedings IEEE INFOCOM, IEEE, 2013, pp. 2157–2165.
[24] D. Shan, W. Jiang, F. Ren, Absorbing micro-burst traffic by enhancing dynamic threshold policy of data center switches, in: 2015 IEEE Conference on Computer Communications, INFOCOM, IEEE, 2015, pp. 118–126.
[25] D. Shan, F. Ren, Improving ECN marking scheme with micro-burst traffic in data center networks, in: IEEE INFOCOM 2017-IEEE Conference on Computer Communications, IEEE, 2017, pp. 1–9.
[26] H. Wu, J. Ju, G. Lu, C. Guo, Y. Xiong, Y. Zhang, Tuning ECN for data center networks, in: Proceedings of the 8th International Conference on Emerging Networking Experiments and Technologies, 2012, pp. 25–36.
[27] Y. Pan, C. Tian, J. Zheng, G. Zhang, H. Susanto, B. Bai, G. Chen, Support ecn in multi-queue datacenter networks via per-port marking with selective blindness, in: 2018 IEEE 38th International Conference on Distributed Computing Systems, ICDCS, IEEE, 2018, pp. 33–42.
[28] X. Huang, J. Dong, W. Yang, C. Tian, J. Zhou, Y. Kai, M. Cai, N. Xia, W. Dou, G. Chen, CLEAN: Minimize switch queue length via transparent ECN-proxy in campus networks, in: 2021 IEEE/ACM 29th International Symposium on Quality of Service, IWQOS, IEEE, 2021, pp. 1–6.
[29] C.-H. Tai, J. Zhu, N. Dukkipati, Making large scale deployment of RCP practical for real networks, in: IEEE INFOCOM 2008-the 27th Conference on Computer Communications, IEEE, 2008, pp. 2180–2188.
[30] H. Jiang, Y. Wang, K. Lee, I. Rhee, Tackling bufferbloat in 3G/4G networks, in: Proceedings of the 2012 Internet Measurement Conference, 2012, pp. 329–342.
[31] E. Weigle, W.-c. Feng, Dynamic right-sizing: A simulation study, in: Proceedings Tenth International Conference on Computer Communications and Networks (Cat. No. 01EX495), IEEE, 2001, pp. 152–158.
[32] K. He, E. Rozner, K. Agarwal, Y.J. Gu, W. Felter, J. Carter, A. Akella, ACDC TCP: Virtual congestion control enforcement for datacenter networks, in: Proceedings of the 2016 ACM SIGCOMM Conference, 2016, pp. 244–257.
[33] B. Cronkite-Ratcliff, A. Bergman, S. Vargaftik, M. Ravi, N. McKeown, I. Abraham, I. Keslassy, Virtualized congestion control, in: Proceedings of the 2016 ACM SIGCOMM Conference, 2016, pp. 230–243.
[34] Y. Sarikaya, I.C. Atalay, O. Gurbuz, O. Ercetin, A. Ulusoy, Estimating the channel capacity of multi-hop IEEE 802.11 wireless networks, Ad Hoc Netw. 10 (6) (2012) 1058–1075.
[35] R. Jain, Congestion control and traffic management in ATM networks: Recent advances and a survey, Comput. Netw. ISDN Syst. 28 (13) (1996) 1723–1738.
[36] S. Kunniyur, R. Srikant, Analysis and design of an adaptive virtual queue (AVQ) algorithm for active queue management, ACM SIGCOMM Comput. Commun. Rev. 31 (4) (2001) 123–134.
[37] M. Alizadeh, A. Kabbani, T. Edsall, B. Prabhakar, A. Vahdat, M. Yasuda, Less is more: Trading a little bandwidth for ultra-low latency in the data center, in: 9th USENIX Symposium on Networked Systems Design and Implementation, NSDI 12, 2012, pp. 253–266.
[38] B. Ginzburg, A. Kesselman, Performance analysis of A-MPDU and A-MSDU aggregation in IEEE 802.11n, in: 2007 IEEE Sarnoff Symposium, IEEE, 2007, pp. 1–5.
[39] D. Shan, F. Ren, P. Cheng, R. Shu, C. Guo, Observing and mitigating micro-burst traffic in data center networks, IEEE/ACM Trans. Netw. 28 (1) (2019) 98–111.
[40] P.X. Gao, A. Narayan, G. Kumar, R. Agarwal, S. Ratnasamy, S. Shenker, Phost: Distributed near-optimal datacenter transport over commodity network fabric, in: Proceedings of the 11th ACM Conference on Emerging Networking Experiments and Technologies, 2015, pp. 1–12.
[41] I. Cho, D. Han, K. Jang, ExpressPass: End-to-end credit-based congestion control for datacenters, 2016, arXiv preprint arXiv:1610.04688.
[42] S. Varma, Internet Congestion Control, Morgan Kaufmann, 2015.
[43] J. Crichigno, E. Bou-Harb, N. Ghani, A comprehensive tutorial on science DMZ, IEEE Commun. Surv. Tutor. 21 (2) (2018) 2041–2078.
[44] V. Jacobson, Congestion avoidance and control, ACM SIGCOMM Comput. Commun. Rev. 18 (4) (1988) 314–329.
[45] S. Floyd, RFC 3649: HighSpeed TCP for Large Congestion Windows, IETF, 2003, URL https://www.rfc-editor.org/rfc/rfc3649.
[46] L. Xu, K. Harfoush, I. Rhee, Binary increase congestion control (BIC) for fast long-distance networks, in: IEEE INFOCOM 2004, Vol. 4, IEEE, 2004, pp. 2514–2524.
[47] S. Karandikar, S. Kalyanaraman, P. Bagal, B. Packer, TCP rate control, ACM SIGCOMM Comput. Commun. Rev. 30 (1) (2000) 45–58.
[48] H.-Y. Wei, S.-C. Tsao, Y.-D. Lin, Assessing and improving TCP rate shaping over edge gateways, IEEE Trans. Comput. 53 (3) (2004) 259–275.
[49] S. Pilosof, R. Ramjee, D. Raz, Y. Shavitt, P. Sinha, Understanding TCP fairness over wireless LAN, in: IEEE INFOCOM 2003. Twenty-Second Annual Joint Conference of the IEEE Computer and Communications Societies (IEEE Cat. No.03CH37428), Vol. 2, 2003, pp. 863–872, http://dx.doi.org/10.1109/INFCOM.2003.1208924, vol.2.