Bidirectional Bandwidth Coordination under Half-Duplex Bottlenecks for Video Streaming

Jing Chen*, Bo Wang*†, Zhiyuan Xu*, Yan Zhang*, Minhu Wang*, Mingwei Xu*†, Zili Meng‡
*Tsinghua University †Zhongguancun Laboratory ‡Hong Kong University of Science and Technology
j-chen16@tsinghua.org.cn, wangbo2019@tsinghua.edu.cn,{xuzy24,zhangyan24,wangmh19}@mails.tsinghua.edu.cn, xumw@tsinghua.edu.cn, zilim@ust.hk

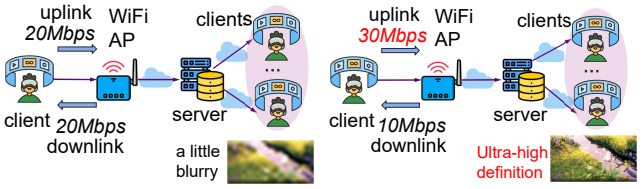
***Abstract*—Many video streaming applications will simultaneously transfer data in both directions, from the user to the Internet (uplink) and from the Internet to users (downlink). However, for wireless local area networks (WLANs), the dominant scenarios, the uplink and downlink flows share the same half-duplex physical channel and compete for bandwidth resources. Their bandwidths would be fairly apportioned under the existing link layer access method, but a fair share might be suboptimal for applications. For better application performance, we propose Plum, to coordinate the bitrate of uplink and downlink flows, and allocate the bandwidth in both directions to cater to the application's demands. To make the deployment of Plum practical, we aim at not modifying the link layer but optimizing the transport layers and above. We evaluate our mechanisms with simulations based on real-world traces and testbed experiments, and results show that Plum could improve the video bitrate of streaming applications by up to 48-59%.**

***Index Terms*—Video streaming, WLAN.**

I. Introduction

Wireless video streaming applications with high interactivity, such as augmented reality (AR) / virtual reality (VR) chatting or remote rendering [1]–[7] and videoconferencing [8]–[10], are getting more and more popular. For these applications, wireless networks are the most convenient way to access the Internet, which is expected to take up more than half of traffic in the foreseeable future [11]. A noticeable difference is that applications like VR require high-volume data transmission in *both directions* – both uplink (from the user to the Internet) and downlink (from the Internet to the user) flows are volumetric. This is different from traditional live streaming [12]–[14], where the data is mainly unidirectional from the Internet to the user. For example, while receiving video contents, VR users would upload several tens of Mbps stream containing their avatars, graphics, videos and many other VR world components to a remote server or other users [15]. (§II-A1)

An outstanding feature of WLAN is that their channels are *half-duplex* – The access point (AP) transmits downlink flows and the client transmits uplink flows into the same physical channel, competing for limited network resources (Fig.2). Meanwhile, the uplink and downlink flows tend to *fairly*[1] share the network capacity based on the IEEE 802.11 medium access control (MAC) layer protocol [16] under the most widely deployed channel access method [17]. (§II-A2)



(a) Blurry streams for many users without bandwidth coordination. (b) UHD streams for many users with bandwidth coordination.

Figure 1: Video streaming might prefer unfair bidirectional bandwidth allocation

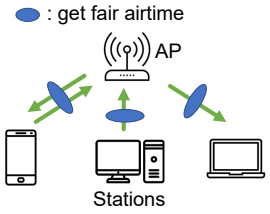


Figure 2: Uplink and downlink flow compete based on WiFi per-station fairness

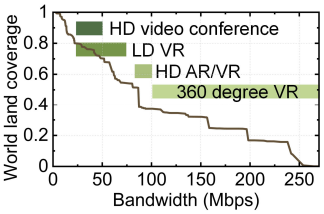


Figure 3: Video bandwidth requirement vs. World support for WiFi speed

However, from the application's perspective, a fair share between uplink and downlink might not be optimal for the Quality-of-Experience (QoE). Instead, an *unfair bandwidth allocation between the uplink and downlink* can maximize the QoEs. For example, in a multi-user video streaming application, a user's uploading video quality decides the upper bound of all the other viewers' video quality[2]. As exemplified in Fig.1(a), if the uplink and downlink flow fairly split the WLAN capacity (e.g. 40 Mbps), the host only uses 20 Mbps to transmit a degraded quality VR video stream. In a VR virtual world where a player is presenting important videos to attendees, the degradation of uplink quality will affect the QoE of all users [18], [19] – at the same time, the quality of the video/graphics from other attendees to this video player does not matter as much. In this case, if we could unfairly "allocate" more bandwidth to the uplink and less to the downlink, many viewers would enjoy a satisfactory experience (Fig.1(b)).

The most straightforward way is to modify the MAC layer protocol to be adaptive to the application requirements. How-

[1]The word "fair" means getting equal chances to transmit data. The bandwidth would be statistically close to equal, not precisely equal (See §V).

[2]Even if the relay server can adopt super-resolution to upscale the video quality [18], [19], the authentic outputs are both computational-expensive and unfaithful for real-time video streaming.

979-8-3503-5171-2/24/$31.00 ©2024 IEEE

ever, this would incur large cross-layer coordination overhead. It is also impractical to ask video streaming users to upgrade their wireless hardware to enjoy the performance benefits. To make it more lightweight, we seek to solve the problem with rate control (from the transport layer and above). This is feasible because, when the downlink sending rate is decreased and the uplink increased, there are fewer downlink packets to be sent and the vacated WiFi channel airtime could be used by the uplink flow. The bandwidth is then "yielded" from one direction to the other.

Unfortunately, existing rate control mechanisms at endpoints are *unidirectional*, that they only control the outgoing flow sent by the user and lack coordination between the (uplink) flow sent by the user and the (downlink) flow received by the user. Both the adaptive bitrate (ABR) in the application layer [20]–[26] and the congestion control in the transport layer [27]–[32] are trying to match the sending rate with the *one-way available bandwidth* observed by the sender, and fail to realize bidirectional flow coordination (§II-B)

To achieve a better QoE for video streaming applications, we propose Plum, an application-layer rate control method that coordinates bidirectional (i.e. uplink and downlink) bandwidths under half-duplex bottlenecks. Our key insight is to *actively adjust the application data rates to "reallocate" the bandwidth between the uplink and downlink* rather than passively matching the competition outcome from the MAC layer. We implement this in the application layer as it is more convenient to get application information. (§II-C)

However, we still face two main challenges during design: at the policy level, deciding how much bandwidth to yield between both directions to match the application's need is challenging. The QoE of the streaming application is nontrivial to define and optimize with multiple users. The uplink and downlink bitrates in total are limited by the capacity and trading off with each other. A higher uplink bitrate would increase other users' QoE while a higher downlink bitrate would benefit the user's own QoE. With the constraints and trade-off, it is nontrivial to decide the bidirectional bitrates. In response, we set up an optimization model based on Hoβfeld's utility function [33] (§III-B), provide interpretable mathematical analysis results of the optimal rate allocation under various conditions, and use a nonlinear programming algorithm to solve the optimal rate allocation and maximize QoE in multiuser streaming scenarios. (§III-C)

From the system level, it is also challenging to practically enforce bidirectional bandwidth coordination. Since the bidirectional flows have distributed senders (i.e., clients and a relay server), it requires careful system designs to make bandwidth decisions in one place and integrally notify all the senders without inconsistency. Meanwhile, it is also important to work well with existing rate control mechanisms like congestion control. In response, we design a server-dominant, easily deployable system that uses entangled state machines to consistently control the bitrate of uplink and downlink flows without interrupting the congestion control logic. (§III-D)

We conduct our experiments with real-world trace-driven simulations and testbed experiments under the scenario of multiuser video streaming applications. Our simulation results show that Plum could improve the average user bitrate of video streaming clients by up to 48-59% compared to the best of the baselines, while being fair to the worst-case users. The testbed experiments also show $\sim$16% improvement in average user bitrate, further validating the real-world effectiveness of Plum. (§IV)

The main contributions of this paper are:

- We identify the lack of bandwidth coordination between uplink and downlink traffic under WiFi bottlenecks, which impairs the performance of streaming applications. (§II)
- We design Plum, a bandwidth coordination method with both a policy and a system, that tailors the uplink and downlink bitrate to acquire more reasonable bandwidth shares under half-duplex bottlenecks. (§III)
- We carry out NS-3 simulations and testbed experiments to evaluate Plum's performance, user fairness, and effectiveness of Plum's design. Our code is available at https://github.com/PlumRateControl/Plum. (§IV)

Ethical statement: Only Fig.6 involves human subjects. Our tracker application only measures bandwidth usage and gets user consent from the volunteers. User identity is masked, and no private information (e.g., application names) is stored.

II. Background & Motivation

In this section, we introduce the background of bidirectionally volumetric streaming applications (§II-A1) and WLAN half-duplex bottlenecks (§II-A2), then motivate Plum (§II-B).

A. Background

1) Bidirectionally Volumetric Streaming: In recent years, high-definition interactive streaming applications are becoming increasingly popular, including three-dimensional Augmented Reality / Virtual Reality (3D AR/VR) for socialization, gaming or metaverse [6], [15], [34], [35], and 4K video conferencing [8]–[10], [36], where clients push heavy-loaded uplink flow (e.g., from their device camera or local files) and watch a downlink video from peer client(s). These applications involve volumetric streaming upload and download simultaneously, which require tens to hundreds Mbps bandwidths to support high streaming quality (e.g., green bands in Fig.3, data sourced from [15], [35], [36]). Meanwhile, these applications often have many attendees [35].

2) WLAN: Today's video streaming users largely use WLANs to access the Internet, as WLANs are ubiquitously deployed and supported by user devices. Fig.3 plots the median download speeds of WiFi users based on Speedtest public dataset [37], which is often higher than the actual real-time bandwidths. With the growth of bandwidth requirements of the applications (e.g. HD VR), more than half of the world's land could not support such demands. With similar signal strengths and data rates, the transmission from the AP to the station and the station to the AP will share comparable bandwidths. This is based on the most widely deployed access method, Distributed Coordination Function (DCF). (See discussions on

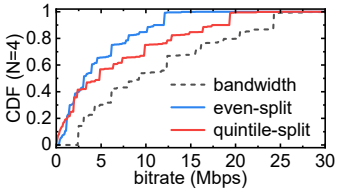


Figure 4: Theoretical improvement with uneven bidirectional bandwidth

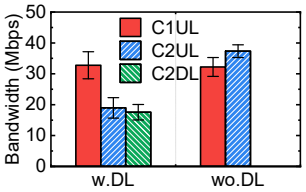
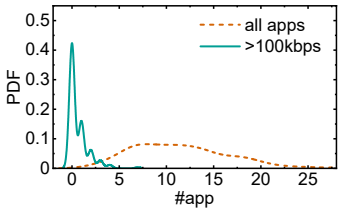
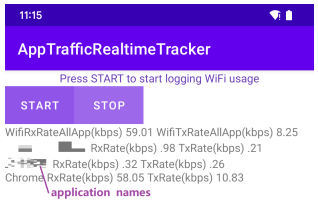


Figure 5: Bandwidths w./w.o. downlink flow to Client2 (C2DL)



(a) Probability Distribution Function (PDF) of application parallelism



(b) Real-time traffic tracker GUI (blank region cropped)

Figure 6: Real-time smartphone application parallelism.[3]

other access methods in §V.) In other words, without any special control, the uplink and downlink flows of one user would get fair bandwidth shares.

B. Motivation

Imbalanced uplink and downlink bandwidth allocation might result in higher QoE. For interactive applications with multiple participants, the uplink bitrate is the upper bound of the downlink bitrate for other viewers [18], [19]. When other users have abundant bandwidth, the uplink bandwidth is crucial. As a motivating example, we estimate the expected benefit of higher uplink bandwidth based on real-world WiFi traces (*Tr-Game* in §IV-A), where the bandwidth distribution is the dotted line (labeled as *bandwidth*) in Fig.4. Assume that one user suffers from this WiFi bottleneck and that the other users (accessing the Internet via wired links or different WiFi APs) have abundant bandwidth, and the uplink flow of each user is transcoded and relayed to other users. If the WiFi user receives a downlink flow enough for its communication, moderately increasing the uplink bandwidth is supposed to improve the average user bitrate of all users. We compare the distribution of the user bitrate when the WiFi bottleneck bandwidth is evenly split (*even-split*), and when the bandwidth is unevenly split between the uplink (0.8) and downlink (0.2) (*quintile-split*). The median of the average bitrate is improved by 58.1% with *quintile-split*. At other times, we might also want to allocate more bandwidth for the WiFi user's downlink when other peers are too congested to obtain higher bitrate even when the WiFi user uploads videos with higher quality. The bandwidth shares ought to adapt to various situations (number of users, QoE metrics, network bandwidth limits) to maximize the overall QoE.

Changing the MAC layer access method would be heavyweight and lack deployability. Intuitively, the bandwidth share between the uplink and downlink is the direct consequence of the MAC layer access method. However, modifying the implementation of the MAC layer access method (e.g. the backoff contention window) might have three drawbacks: (1) Changing the contention mechanism might break the fairness principle and affect the bandwidth of other users; (2) Adapting the MAC layer mechanisms to application layer instructions would be a large-span cross-layer design, which breaks the simplicity of the layered network architecture and might bring up unintended interactions and spaghetti designs [38], [39]. It incurs cross-layer signaling overhead for the MAC layer to get the application requirements in real time and to frequently change its channel access mechanism. (3) Changing the WiFi MAC-layer channel access method is hard to deploy as it may require modifications to the firmware of commercial WiFi routers. Therefore, we should consider transport layer rate control (i.e., congestion control) or application layer rate control to realize bidirectional bandwidth coordination.

Bidirectional rate coordination is feasible under the existing MAC access method. The downlink bandwidth of one user could be yielded to its uplink (and vice versa), instead of being preempted by other competing users. This is because the current MAC layer access method follows per-station fairness, i.e., each station (user) in the WLAN gets equal airtime to transmit. (See the evolution of WiFi fairness in [40]) Our experiment based on a simple testbed also observes the above WLAN bandwidth acquisition behaviors. We launch Iperf [41] bulk flows between each client and a server. The two clients (Client1 is a laptop and Client2 is a desktop) are connected to the same WiFi AP (Tenda AX1803), and the AP and the server are in the same sub-network. Each Iperf experiment lasts for one minute and is repeated 5 times. The results show that the bidirectional bandwidth reallocation of Client2 barely impacts Client1's bandwidth (Fig.5). The uplink bandwidth of Client2 (C2UL) (37.3Mbps) without downlink to itself (C2DL) is approximately the sum of C2UL and C2DL (19.0+17.6=36.6Mbps) when there are flows from both directions. Therefore, we could seek to reapportion the uplink and downlink rates under the existing MAC access method.

The existing rate control mechanisms are unidirectional. As we summarized in §I, existing rate control mechanisms control the rate of each stream independently, which is well-functioning when there is only one stream or loosely related streams (e.g., parallel, homogeneous streams in the same direction simply to enhance throughput) inside an application. However, when the streams are from different directions and have complex cascaded influences on the application QoE, indulging them to grab bandwidth on their own would not end up with a satisfactory QoE. In addition to traditional congestion control and ABR that work for unidirectional flow, a few other rate control mechanisms approach the edge of our issue. However, they are also optimizations for unidirectional flows and cannot achieve bidirectional bandwidth coordination. Scavenger congestion control [42], [43] prioritizes network bandwidth for certain applications but lacks

[3]Tested with a self-developed Android program by invoking the *NetworkStatsManager* API. The data is collected every 10 minutes by 5 volunteers when they normally use their smartphones across a week.

prioritization for bidirectional flows in the same application. Meanwhile, based on our measurement, in many cases there is only one application with volumetric traffic on a smartphone (>100kbps, the solid green line in Fig.6(a)), even though many applications run simultaneously on smartphones (dotted line). Therefore, there may not be a scavenger flow to yield the bandwidth. While also considering the half-duplexity of WiFi channels, TACK [44] optimizes the frequency of acknowledgments (ACKs), which is also inside each unidirectional flow. For streaming applications with volumetric reverse traffic, the contention overhead of small ACKs becomes marginal.

C. Design Choice

It is feasible and more convenient to coordinate bidirectional bandwidths in the application layer compared to the transport layer. Compared to the transport layer rate control, we choose application layer rate control since (1) it is easier to get the application instructions and (2) it is more deployable without any kernel-space modifications. We can realize our intended functionality by actively shrinking the bitrate of the video from one direction, and letting the stream from the other direction grab the vacated bandwidth (Fig.7). In the shrunk direction, the congestion control enters the "application-limited" state. In the other direction, the requirement for the congestion control algorithm is that it is a well-functioning algorithm that could quickly grab the vacated bandwidth. At a high level, the existence of bidirectional flows in one application creates a factual gap between "application" and "flow". Existing rate control regards the sender and receiver of a flow (Layer 4) as the "endpoint" in the end-to-end argument. Instead, we should regard the application (Layer 5) as the ultimate endpoint and coordinate bidirectional flows under the application to eventually achieve a higher QoE.

Summary: We should coordinate the bandwidth for bidirectional streams under half-duplex bottlenecks, by *controlling application data rate to reallocate bidirectional bandwidths*.

III. Design

Realizing this idea of bidirectional bandwidth coordination still faces many challenges (§III-A). In this section, we solve these challenges and elaborate our design (§III-B,§III-C,§III-D).

A. Design Challenges & Overview

Policy Challenge: How much bandwidth should be yielded from one direction to the other? Both uplink and downlink flow constitute application traffic and impact the user-perceived QoE. How much bandwidth should be yielded between the uplink and downlink, to achieve the best QoE for the applications? If the flow from one direction (e.g., the downlink) is yielded overly, the user would entirely sacrifice her/his own watching experience. If the yielding is insufficient, we may not be able to observe a significant improvement in overall performance. The network capacity limits and video content dependencies should also be considered, making the target bitrate non-trivial to decide. In response, we set up utility function as metrics (§III-B), formulate an optimization problem, and mathematically analyze the optimal bandwidth allocation in a multiuser video streaming scenarios (§III-C). Broader application scenarios are discussed in §VI.

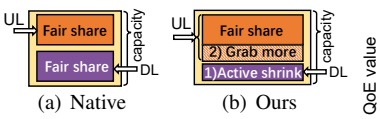


Figure 7: Reallocating the bandwidth by tuning bidirectional bitrates. UL and DL are for uplink and downlink flows.

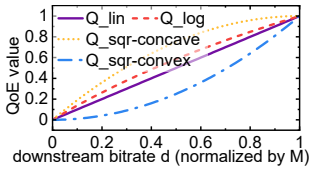


Figure 8: The trends of the QoE functions

Name	Function ($Q(d)$)	Coefficient
Q_{lin}	d/M	–
Q_{log}	$\log(d+1)/\log(M+1)$	–
Q_{sqrcc}	$\alpha d^2 + (M^{-1} - M\alpha)d$	$-M^{-2} \leq \alpha < 0$
Q_{sqrcv}	$\alpha d^2 + (M^{-1} - M\alpha)d$	$0 < \alpha < (M^2 - 2M)^{-1}$

Table I: Single-user QoE functions used in our utility function. M is the maximum application bitrate. The QoE function gets a value of 1 if d exceeds the maximum application bitrate M.

System challenge: How to practically and integrally enforce bidirectional bandwidth coordination? After knowing the optimal bandwidths, integrally enforcing them requires collaborative control at both the clients and the server by in-band decision notification. How to ensure consistency of rate decisions at all involved end-hosts? Under varying wireless networks, our policy needs to adapt its target bitrates to varying network conditions from time to time. As a rate control mechanism, Plum also needs to seamlessly integrate with existing rate control modules such as congestion control. To cope with these system challenges, we use entangled state machines to enforce bidirectionally consistent rate control states. We periodically detect changes in network capacity and control the video encoding and forwarding units without intercepting the original logic of the congestion control. (§III-D)

Design Overview. The rest of this section first introduces how to decide the bitrate of the media traffic in both directions. To this end, we first set up the metric to evaluate application performance in §III-B, which comprehensively reflects overall performance and user fairness in multiuser applications. With the metric as our objective, we formulate and solve the target rate allocation in §III-C. In this part, we not only show some interpretable results in some simplified cases but also adopt a nonlinear programming method to produce optimal rate allocation results for general cases. Finally, we elaborate our system design to complete the function of bidirectional rate coordination in §III-D.

B. QoE Metrics Setup

An overarching problem is how to measure overall performance. When the application involves multiple users, it becomes difficult to tell whether the overall performance is improved or impaired when users experience uneven performance. For example, if five users get an enhanced experience while three other users get a slightly degraded experience, may we call this an improvement and how do we evaluate the level of improvement or degradation?

To handle this issue, we select a utility function as the maximization objective for our optimization, based on Hoβfeld's QoE metrics for multiuser applications [33].

$$\mathbb{U} = (1-\rho)\cdot\overline{Q(\mathbf{d})} + \rho\cdot F(Q(\mathbf{d})), \quad (1)$$

$$\text{where} \quad F(\mathrm{Q}(\mathbf{d})) = 1 - 2\sigma(\mathrm{Q}(\mathbf{d})) \quad (2)$$

In this formula, $\mathbf{d}$ is a vector that contains the downlink bitrate for all users, $Q(\cdot)$ is a normalized QoE function of downlink bitrates for a single user, scaling in the range of [0,1], and $\sigma(Q(\mathbf{d}))$ is the standard deviation of the QoE of all users. The utility is a function of downlink bitrates as they directly influence the users' watching experience. Video streaming applications may benefit from a higher upload speed, but it should finally reflect in someone's downlink bitrate. The utility is composed of an average performance item (i.e., the average QoE of each user $\overline{Q(\mathbf{d})}$) and a fairness item (i.e., the fairness among all users $F(Q(\mathbf{d}))$). As the QoE of each user scales in [0,1], the average performance item scales in [0,1]. The fairness item gets a value of 1 if all users get the same QoE and zero standard deviation, and 0 if half users have the worst QoE and the other half have the best QoE. Hence, the fairness item also scales in [0,1]. ρ is a parameter between 0 and 1, to tune the weight between the average performance and users' fairness. A greater ρ represents a stronger emphasis on the user's fairness. We use the standard deviation in Hoβfeld metric to evaluate fairness instead of the max-min fairness in Minerva [45], as it is less sensitive to the inherent differences in network condition among users. For example, if the available bandwidth of one user is much lower than the others, it might be deterministic to the max-min fairness and make the actual fairness among other users indifferent in the metric.

The advantage of this utility definition is that it is independent of the underlying QoE function. As long as the (single-user) QoE function is a normalized function, the above utility function could compute a summarized QoE for multiple users who participate in the application. Therefore, we do not need a single-user QoE metric with accurate parameters but only its trend, because the $Q(\cdot)$ needs to be normalized and rescaled in [0,1]. Meanwhile, intuitively, the single-user QoE would be positively related to its downlink bandwidth, so the $Q(\cdot)$ function should be monotone non-decreasing. Combining the above conditions, we use four types of $Q(\cdot)$ function as shown in Tab.I: linear Q_{lin}, logarithmic Q_{log}, concave square Q_{sqrcc}, and convex square Q_{sqrcv} functions. All of them map [0, Maximum application bitrate] to [0,1]. Their trends are visualized in Fig.8, and we could also get other QoE functions lying in the region between the short dotted and dash-dotted lines by tuning the coefficient of Q_{sqrcc} and Q_{sqrcv}.

C. Target Allocation Analysis

We set up an optimization model to analyze the target uplink and downlink bitrates. The model has a few assumptions:

- The streaming application involves video interactions among multiple clients, and the clients have bidirectional volumetric video streams.
- The uplink flow of each client is multicasted to other clients by a server (e.g., the Selective Forwarding Unit (SFU) server in videoconferencing or the relay servers of other streaming platforms, all "server" hereinafter refers to this).
- The streams could only be transcoded to lower bitrate during forwarding (i.e., the downlink bitrate is bounded by the sum of other clients' uplink). (see §II-B)
- The sum of the WLAN uplink and downlink bandwidths is regarded as the available bandwidth. (see §II-B)

The formulation of the problem is as follows:

Variables: $\mathbf{d}$, a vector that contains the downlink bitrate d_i for each client i (application participants).

Constraints:

- (*Positive bitrate selection*) $\forall i, d_i \geq 0$, i.e., the selected downlink bitrate must be non-negative.
- (*Available bandwidth bound*) $\forall i, B_i - d_i \geq 0$, i.e., the selected downlink bitrate must be smaller than the capacity of the half-duplex bottleneck B_i, so that the client could send some uplink traffic without incurring network congestion.
- (*Content source bound*) $\forall i, d_i \leq \sum_{j\neq i}(\min(B_j - d_j, M/V))$, i.e., the downlink bitrate of one user could not exceed the streaming contents provided by other users. M is the maximum application bitrate. V is the degree of compressibility of one client's video when it is played back to another client (i.e., the original video streaming might be downward transcoded). If we relax the consideration for the maximum application bitrate, this constraint is equivalent to

$$\textstyle\sum_i d_i \leq S = \sum_{i\neq k, k=argmax B_k} B_i \quad (3)$$

Objective: maximize the utility function $\mathbb{U}(\mathbf{d})$ in Eq.1.

Before analysis, we introduce several other necessary denotations: n represents the number of users in total, u_i represents the uplink bitrate sent by client i, and $\max^{2nd}(\cdot)$ denotes the second largest value among a certain range.

Results for a special case ($\rho = 0$ and the QoE function takes Q_{lin}). In this case, optimizing the utility function is simplified into maximizing $\mathbb{U}' = \Sigma_i \min(d_i, \Sigma_{j\neq i} u_j)$, the average receiving bitrate of all clients. The receiving bitrate could roughly reflect the video quality and the watching experience of the client. We first take the simplified version of the content source bound constraint (Eq.3). Under this special case, it is feasible for us to directly solve the optimization problem. Since the solution varies depending on whether the users are network-limited or application-limited, we discuss the two cases separately and draw the following two conclusions (proved in [40]).

Theorem 1 (Network-limited)**.** *If all clients have a half-duplex bottleneck, the bandwidth allocations to achieve the maximum average receiving bitrate of all clients (utility under the special case) are* $\max^{2nd}(B_i) \leq \sum_i u_i \leq \max(B_i)$ *and* $d_i = B_i - u_i$.

The intuitive explanation for this conclusion is that in these n users, there could only be one user with a non-saturated bottleneck. If more than one user has nonsaturated bottlenecks,

the traffic between them could be increased to enhance the overall receiving bitrate.

Theorem 2 (Application-limited)**.** *If there exists at least one client with abundant network bandwidth (more than the sum of all flows under the maximum bitrate), then the uplink bandwidth of each client should be as large as possible to improve the average receiving bitrate (utility under this special case) of all clients.*

Intermediate results (but with a certain degree of interpretability) for general cases. For more general cases, we use Lagrange multiplier approach [46] to solve the extreme points of the problem. In this part, we take the simplified version of the content source bound constraint (Eq.3).

We use the following denotations as the Lagrange multipliers: two vectors λ and μ that each contain n elements, and another variable η. These variables are equal to or greater than zero. Then, we construct a Lagrange function (The S in the following equations is defined in Eq.3):

$$\mathcal{L}(\mathbf{d}, \lambda, \mu, \eta) = \mathbb{U}(\mathbf{d}) + \sum_i(\lambda_i d_i + \mu_i(B_i - d_i)) + \eta(S - \sum_i d_i)$$

with inequality constraints:

$$\forall i, d_i \geq 0; \quad \forall i, B_i - d_i \geq 0; \quad S - \sum_i d_i \geq 0$$

Based on the Karush-Kuhn-Tucker (KKT) conditions for the Lagrange multiplier approach with inequality constraints [47], we could get:

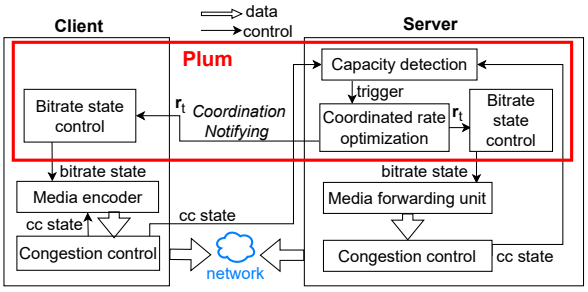
$$\forall i, \nabla_{d_i} = \nabla \mathbb{U}_{d_i}(\mathbf{d}) + \lambda_i - \mu_i - \eta = 0;$$
$$\forall i, \lambda_i d_i = 0; \quad \forall i, \mu_i(B_i - d_i) = 0; \quad \eta(S - \sum_i d_i) = 0$$

From solving the KKT conditions, we could know that the downlink bitrate d_i for each user i in the extreme points could be one of the following five potential values:

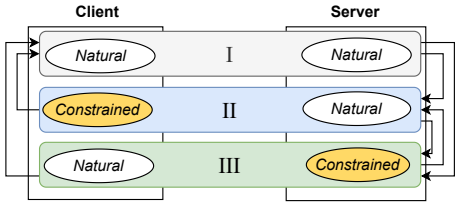
$$①d_i = 0;\ ②d_i = B_i;\ ③d_i = S - \sum_{j \neq i} d_i;\ ④\nabla_{d_i} Q(d_i) = 0;$$
$$⑤Q(d_i) = \overline{Q(\mathbf{d})} + \frac{1-\rho}{2\rho} \cdot \frac{n}{n-1} \cdot \sigma(Q(\mathbf{d}))$$

In these five values, ①② are trivially drawn from the problem constraints, ③ is equivalent to the special case result in Thm. 1. ④ is also trivial since the $Q(\cdot)$ function is usually monotonically increasing, as we discussed in §III-B. ⑤ is the one that comprehensively reflects the components in the utility function (Eq.1). An intuitive understanding for ⑤ is that if none of the users is network-limited, d_i for all users should satisfy ⑤ and would result in the same d as the maximum application bitrate for all users. In practice, when the common situation is that some users are network-limited, selecting the same bitrate for all users would make the bitrate very low. In this case, users must balance between maintaining their own receiving bitrate and providing quality uplink video for others by satisfying either ③ or ⑤.

However, there are too many extreme points (5 for each client as above, 5^n in total) that could be solved from the KKT conditions. It is hard to enumerate all these possible solutions and comparatively validate them. Thus, we still need a scalable method to get the optimal solutions for our problem.



(a) Plum's system design



(b) State machines in Plum bitrate state control module

Figure 9: Plum's design

Optimization method for general cases. Finally, to get the optimum solutions, we adopt the Sequential Least SQuare Programming (SLSQP) algorithm [48] to get our optimal bitrates. The SLSQP algorithm can handle nonlinear object functions (with quadratic problems with second-order approximation) and various constraints (with linearized constraints approximation), which applies to our problem. The algorithm is then able to converge to the optimal point.

D. System Design

To make it feasible in real-world deployment, we now propose the system to realize our bidirectional bandwidth coordination policy. We outline the system design in Fig.9(a). Originally, the media would be first encoded on the client side and then sent out based on the speed under congestion control. The media traverse the network and are received by the server. The media forwarding unit would then process the received media and forward them to other clients under server-side congestion control. However, this would not achieve the desired application performance as we have analyzed. To achieve higher overall performance, Plum adds several controlling modules atop these existing mechanisms.

Plum has three main logical modules: the *capacity detection* module, *coordinated rate optimization* module and *bitrate state control* module, as shown in the red box in Fig.9(a). Among them, the capacity detection and coordinated rate optimization modules are located on the server, while the bitrate state control module is located on both the client and the server. The workflow performs as follows: The capacity detection module regularly detects the capacity by getting bandwidth estimations from the congestion control module, and triggers the coordinated rate optimization module to recompute the target (bidirectional) bitrates if obvious capacity changes are detected to take place. Then, the coordinated rate optimization module analyzes with the new capacities and provides the updated target bitrates. The server notifies the

client with the target bitrate $\mathbf{r}_t$. Based on these new bitrate decisions, the clients and the server would update their states by the bitrate state control module in an entangled way. On the client side, the bitrate state control would control the encoding bitrate in the media encoder. On the server side, the bitrate state control would guide the media forwarding unit to decide the transcoding ratio of the media to be forwarded. With more than one user participating in the application, all modules run their logic independently for each client except for the coordinated rate optimization module. There is only one centralized coordinated rate optimization module that decides the bitrates for all clients with an overarching global view. In the following, we introduce the design and implementation of the three main logical modules of Plum:

Capacity detection. The capacity detection module periodically checks the network capacity by getting the bandwidth estimation from the congestion control (e.g., rate-based congestion control algorithms like BBR [28] would explicitly make a rough estimation of the available network bandwidths). The server estimates the downlink bandwidth while the client estimates the uplink bandwidth and notifies the server by application layer messages (or piggybacked by the media traffic). This module then takes the sum of the uplink and downlink bandwidth estimation as the capacity of the WiFi bottleneck. Then, we compare the real-time capacities with previous capacities and report a change if the capacity of *any client* reveals an obvious change. In our implementation, our detection period is set as 5 seconds. We define "obvious change" as changes in capacity compared to the previous capacity exceeding 20%.

Coordinated rate optimization. We set up a Python solver that calls the `optimize.minimize` API [49] in SciPy to perform the optimization. We use Linux IPC (Inter-Process Communication) sockets to communicate with the Python solver to feed the inputs and get the optimized bitrates. Aside from the convenience of getting a global view of all the clients, putting this optimization module on the server also enables us to utilize stronger computation resources on the servers.

Bitrate state control. After deciding the target bitrates, we further need state machines to control the media encoding or processing and selective forwarding at both the client and the server in an entangled manner. The transitions of the state machine are only triggered on the server and are notified to the clients. We first define *Natural* as a host encoding and sending media based on rates decided by the host's own congestion control algorithm, and *Constrained* as a host encoding and sending media on a settled rate (which is computed by Plum's optimization and is lower than its own congestion control rate). As shown in Fig.9(b), there are three states in total: (I) both the client and the server are in the *Natural* state, (II) the client is *Constrained* and the server is *Natural*, and (III) the server is *Constrained* and the client is *Natural*. When capacity detection determines an obvious change in capacities, the coordinated rate optimization module provides a set of new target bitrates. We first check whether the results are valid (i.e., satisfying all the constraints in §III-C). If the results are valid, we go to state II if the bitrate of a client should be reduced based on the target bitrates, and go to state III if the forwarding bitrate of the server should be reduced (the state transitions on the right side of Fig.9(b)). If the results are invalid, we go back to the natural state I (the transitions on the left side of Fig.9(b)). To accelerate the rate convergence, when a host transits from *Natural* state to the *Constrained* state, we store the previous bitrate, and immediately restore to this historical rate when the host is switched back to the *Natural* state. In practice, we first complete the state transitions at the server, and send $\mathbf{r}_t$ to the client if the client's new state is *Constrained* and send 0 if *Natural*. The client would then transit its state accordingly, thus ensuring the consistency between server and client. The overhead of coordination notification is one float field in the application protocol header.

IV. Evaluation

In this section, we present our evaluation of Plum from different aspects including performance (§IV-A,§IV-B,§IV-C), user fairness and sensitivity (§IV-D) with NS-3 simulations. We finally present testbed experiments (§IV-E).

A. Performance Gain

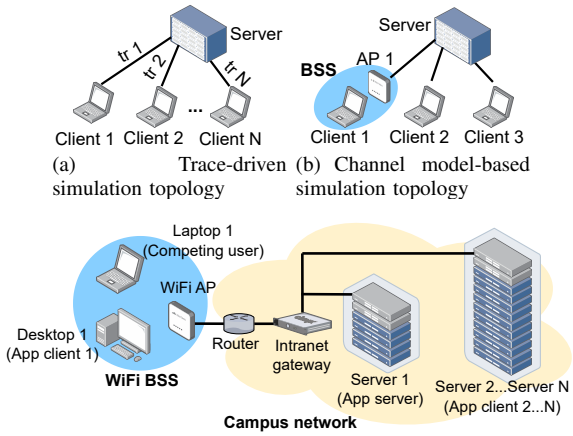
First, we evaluate the performance gains achieved by Plum for a topology where all clients are connected to a central server following the SFU WebRTC architecture, where the access network of each client is independent (see in Fig.10(a)) using NS-3 simulations.

Experiment Setup. We evaluate Plum for two different real-world WiFi traces: Tr-Game [50] (from WiFi gaming users) and Tr-Restaurant [51] (from a WiFi user in a crowded restaurant). In our experiments, each client randomly selects a trace from the trace set and uses it to set the link bandwidth. In the transport layer, we establish TCP connections between clients and the server, and use BBR [28] as the congestion control algorithm, as it is a widely adopted low-latency congestion control algorithm in the real world [52]. The frame rate is set to 20 fps. We set ρ in Eq.1 to 0.5. We use the Q_{lin} QoE function as default. We vary the outbound bandwidth of the server from 100 Mbps to more than 1000 Mbps.

We compare Plum to the following baseline algorithms:

- `TACK` [44]: a flow optimization method with adaptive ACK frequencies. The maximum delayed ACK count is set to 16, and other parameters follow the original paper.
- `LastN` [53]: an application optimization method, where the server only forwards the streams of the latest speakers and pauses the stream from an inactive client to spare bandwidth. The paused user resumes uploading content with a probability of 1/3.
- `Vanilla`: a baseline without special optimization.

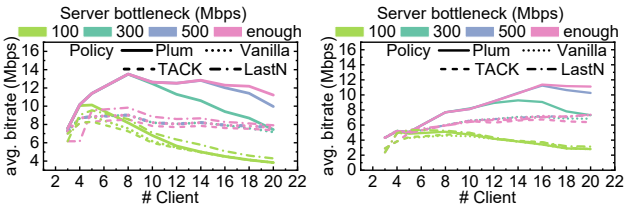
In Fig.11(a), we present the average downlink bitrate for the users for the Tr-Game trace. We see that Plum can improve the average downlink bitrate by up to 48.0% for Tr-Game traces (and 58.5% for Tr-Restaurant in Fig.11(b)), even compared to



(a) Trace-driven simulation topology (b) Channel model-based simulation topology

(c) Testbed experiment topology

Figure 10: Experiment topologies



(a) Tr-Game traces (b) Tr-Restaurant traces

Figure 11: Plum's bitrate performance in trace-driven experiments

the best among the baselines. In fact, as the number of clients increases, the improvement increases, since more clients would amplify the benefit of unfair bidirectional bandwidths.

Because the bandwidth of the server is limited, the average downlink bitrate would eventually stop increasing and gradually drop as the server becomes the bottleneck. We see similar trends for the Tr-Restuarant trace in Fig.11(b). With a larger server bandwidth, the bottleneck at the server will occur after a large number of clients. The highest downlink bitrate is reached at 8 for the Tr-Game trace and at 16 under Tr-Restaurant trace.

To avoid clutter, we only plot the results for 100 Mbps server bandwidth and enough server bandwidth for baseline TACK and LastN in Fig.11. Their performance when the server bandwidth is 300 Mbps and 500 Mbps is bounded by their performance with enough server bandwidth. When the server bandwidth is 100 Mbps, all the policies are restricted by the bottleneck at the server and have similar performance. When there is enough server bandwidth, the performance for TACK and LastN are similar to Vanilla. TACK works well under unidirectionally volumetric scenarios where reducing the ACK frequency reduces the medium acquisition from the other direction. However, when there is constant data flow from the opposite direction, it performs slightly worse than Vanilla because slower ACKs reduce the send rate. A purely application-layer optimization like LastN pauses inactive clients to save bandwidth can achieve some improvement, but is still less effective than bandwidth coordination that is aware of both the network and the application requirements.

To see the real-time bitrate distribution, we present the Cumulative Distribution Function (CDF) of the highest received

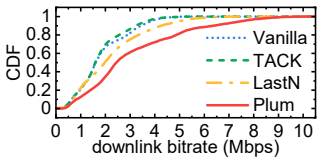


Figure 12: Distribution of real-time bitrate

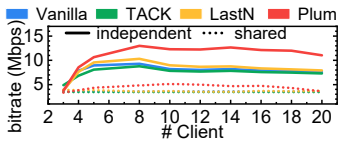


Figure 13: Bitrate including 2 clients sharing the same BSS

Settings	Simulation	Testbed
IEEE 802.11 standard	802.11ac	802.11ac
Physical model	YANS [54]	real WiFi channel
Frequency band	5GHz ISM band [55]	
Access method	DCF [17]	
WiFi fairness	per-station fairness	
MAC rate adaptation	Ideal[4]	Minstrel [56][5]
Mobility model	2D random walk	static
Mobility range	30 meter square	–

Table II: Wireless and mobility settings in our experiments

bitrates among all sources (i.e., each client is supposed to receive flows from all the other clients) in Fig.12. There are 8 clients in total, with enough server bandwidth and Tr-Game traces (the best operating point). Plum improves 90%ile (i.e. 90^{th} percentile) bitrate by more than 49.8%; the proportion exceeding 5 Mbps is improved by 2.6×.

B. Performance with Shared Channel

In practice, we do not expect all clients to have direct and independent access to the server. For example, it is also possible for more than one client to access the server through the same AP (and hence share the same wireless channel). To consider these cases, we use the topology in Fig.10(a) where 2 of the Clients are in the same WiFi Basic Service Set (BSS). We present the bitrates of these two clients sharing the same BSS (`shared`) and the bitrates of other clients (`independent`) separately in Fig.13. While the improvement gain achieved by Plum is smaller compared to that for the clients connected to different APs, but we can still achieve up to 33.7% (for the sharing BSS clients) and 26.1% (for other clients) bitrate improvement compared to the state-of-the-art algorithms.

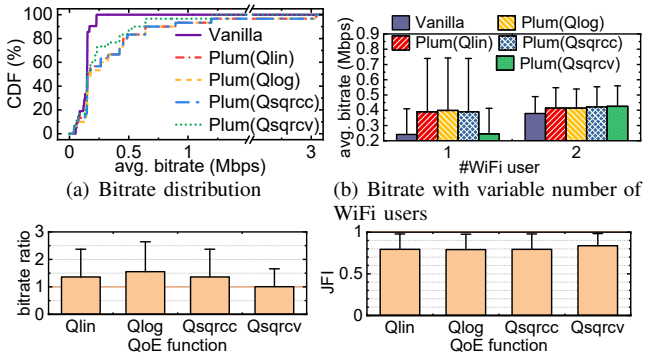
C. Performance under Mobility

As our trace-driven experiments above do not incorporate WiFi physical and MAC layer implementations (e.g., modulation and coding scheme selection), we ran further experiments incorporating the NS-3 WiFi module to evaluate Plum under user mobility. We use the topology in Fig.10(b) with three wireless / wired users, with a variable number of WiFi clients. When there is more than one WiFi client, they are put in different BSSs. The bandwidth of all wired links is set to 50 Mbps. The other wireless settings are shown in Tab.II. Client 1 (a WiFi client) moves by the mobility model in Tab.II and Client 2,3 are stationary. We run each session for 20 minutes and use 10 seeds to eliminate randomness.

In Fig.14(a), we can see that Plum improves the 90%ile user bitrate by 3.8×, and improves the 10%ile user bitrate

[4]The ideal rate adaptation selects the best modulation mode with out-of-band information.

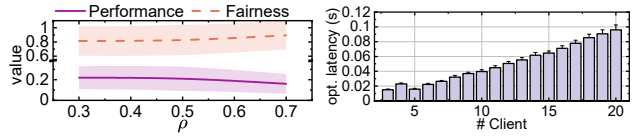
[5]Minstrel in the default rate adaptation algorithm in Linux kernel.



(a) Bitrate distribution (b) Bitrate with variable number of WiFi users

(c) Bitrate ratio of worst-performing user (d) The Jain's Fairness Index (JFI) of Plum

Figure 14: Plum's performance and user fairness in channel model-based experiments



(a) Sensitivity of ρ (b) Optimization latency

Figure 15: Microbenchmarks. The shaded area and the error bar represent the standard deviation.

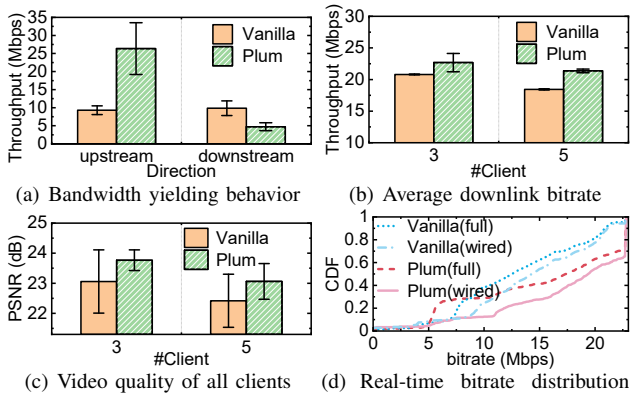
by 25% compared to Vanilla when there is one WiFi user. Fig.14(b) shows that Plum increases the average bitrate by up to 65.1% when there is one WiFi user and 12.3% when there are two. The increase is more significant when there is one WiFi user because wired clients have abundant bandwidth to fully exploit the benefit of the increased uplink bitrate of Client 1, while Client 2 might also be network-limited when it is also a WiFi user. It is not surprising that our performance gains are reduced because the bandwidth varies much more compared to the static scenarios due to user mobility.

D. Fairness and Sensitivity Analysis

Fairness. We present Plum's impact on the worst-performing user in Fig.14(c), i.e., the ratio of the receiving bitrate of the worst-performing user under Plum against without Plum. We see that all Plum variants improve the performance of the worst-performing user (above the orange line). This is due to the cases when two other users increase their upload bitrates to improve the worst user's bitrate significantly. We also calculate the Jain's Fairness Index (JFI) [57] for all users. As shown in Fig.14(d), the JFIs of all Plum variants exceed 0.79.

Sensitivity of QoE functions. Among the four types of QoE functions, Plum with Q_{sqrcv} has the highest JFI (0.84). This is because Q_{sqrcv} is the only convex function in the four QoE functions in Tab.I, making the QoE value for a user grows relatively slowly at the start with the increase of bitrate. Therefore, Plum would prefer to first enhance the fairness item in Eq.1 before devoting itself to improving the average performance for every user. On the contrary, Q_{log} would prefer to improve performance first, which also explains why the performance with Q_{log} is often the best in Fig.14.

Parameter sensitivity of ρ. We also test the sensitivity of the parameter ρ in the utility function (Eq.1). As shown in



(a) Bandwidth yielding behavior (b) Average downlink bitrate

(c) Video quality of all clients (d) Real-time bitrate distribution (full: all users, wired: wired users)

Figure 16: Bandwidth yielding behavior and Plum's performance in testbed experiments

Fig.15(a), with the increase of ρ, the normalized performance term in Eq.1 would gradually decrease and the fairness term would gradually increase. This is in line with our expectations. The range of change is around 0.06 and 0.08 in absolute value for the performance term and fairness term correspondingly.

E. Testbed Experiments

Finally, we evaluate Plum in a real testbed environment with the following settings. Our testbed topology (Fig.10(c)) has a desktop and several Ubuntu servers. The desktop (Desktop 1) accesses the network via WiFi and is one of the clients. The router behind the WiFi AP is in the same campus network as the Ubuntu servers. The servers are in the same intranet and have gigabit NICs. Server 1 acts as the application server, while other servers (Server 2...N) act as the clients in the application and are located in a different rack. Laptop 1 represents another user that shares the same wireless network with our client. Laptop 1 sends a constant 10 Mbps Iperf TCP flow to another server (not shown in Fig.10(c)). Desktop 1 is equipped with a TP-LINK TL-WDN5200H wireless card. The WiFi router model is Tenda AX1803. Server 1 is a Dell PowerEdge R730 server with 16 CPU cores. The desktop accesses the Internet through the WiFi router. Our servers are Dell PowerEdge R740 with 80 CPU cores. We implement Plum in Salsify [58], [59], a state-of-the-art video streaming framework that implements video codec in a functional style and makes it easy to change the coding bitrate.

We first test the bandwidth-yielding behavior in a pair of uplink and downlink flows between the desktop and a Ubuntu server. We halve the downlink bitrate and observe a 2.8× increase in uplink throughput (Fig.16(a)). We then let all clients send streams to each other. Plum could improve the average goodput by 15.9% under 5 clients (Fig.16(b)).

We also evaluate the video quality of Plum in terms of the Peak Signal-to-Noise Ratio (PSNR). As shown in Fig.16(c), Plum increases the average PSNR of all users. The benefit exceeds the decrease of downlink bandwidth of the WiFi user.

We present the distribution of the real-time downlink bitrate in Fig.16(d). The results show that Plum improves the median bitrate by 37.3% among all users (`full`-labeled curves). For all wired users, the portion of bitrate above 10 Mbps increases

Factors	Mechanisms
Airtime	WiFi fairness control
BER & Coding rate	MAC-layer rate adaptation
MAC frame size	Layer 4/5 rate control

Table III: Key influential factors to WiFi bandwidths, and corresponding mechanisms that control them.

by 15%. The bitrate of the self-sacrificing (WiFi) user is also always $>$ 6 Mbps (the left end of the `Plum(full)` curve), not impacting the basic watching of the worst-case user.

We further evaluate the optimization latency of Plum. We repeat 50 times to eliminate randomness. As shown in Fig.15(b), the optimization latency would not exceed 100 ms for 20 participants in the application. Optimization latency would increase linearly with the number of participants, which means that the latency would not exceed 1 second even with 200 participants. This is acceptable since the adjustment to the encoding bitrate would not be very frequent (second level).

V. Related Work

Key influential factors to the throughput of a flow in WLAN. The IEEE 802.11 Medium Access Control (MAC) layer uses the Carrier-Sense Multiple Access with Collision Avoidance (CSMA/CA) access method to handle transmission collision problems in WLAN [16]. It uses carrier sense function (e.g., Network Allocation Vector (NAV) in Distributed Coordination Function (DCF) access method) to indicate the duration of the MAC frame a station needs to transmit, and uses the backoff mechanism to disalign the transmission timing of different users to avoid collisions. Under these mechanisms, a station or an AP gets certain time periods (airtime) to send their data [60]. The airtime apportioned to each user is determined by the WiFi fairness control. During the airtime, the bitrate error rate (BER) of the channel and the modulation and coding rates would also affect throughput. Modulation and Coding Scheme (MCS) with higher rates might suffer a higher BER, so MAC-layer rate adaptations [56], [61]–[67] tune the MCS to seek a balance and maximally exploit the channel capacity. MAC layer rate adaptation is orthogonal to our scope since selecting a sub-optimal MCS only reduces the bandwidth during the sender's airtime, but cannot yield the bandwidth to the other direction. Finally, the MAC frame size would also impact the real bandwidth, as a larger frame size would raise the temporal proportion of data transmission relative to accessing overhead (e.g., the inter-frame spaces). The higher the data rates from the transport layer and above (Layer 4/5), the larger frames could be packed at the MAC layer. (See summary in Tab.III)

Flow prioritization in the MAC layer. IEEE 802.11e standard introduces the Enhanced Distributed Channel Access (EDCA) method [68], [69], which offers prioritized Quality of Service (QoS) support for downlink flows in different Access Categories (AC). EDCA achieves this by configuring different contention windows (CW) and retry factors for each AC in the WiFi AP. However, EDCA only offers QoS support for downlink flows, lacking QoS support between uplink and downlink flows. Another optimization on WiFi AP is the bidirectional DCF, which offers higher bandwidth to downlink traffic by piggybacking data packets after ACKs on AP [70] or reducing backoff counter or minimum CWs for APs and stations [71], [72]. The bidirectional DCF is designed for old-fashioned applications with mice uplink traffic. It could only fixedly give more bandwidths to the downlink flows and could not flexibly coordinate uplink and downlink bandwidths.

VI. Discussion

Bidirectional bandwidth coordination for other application scenarios. In this paper, we establish our design on the multiuser video streaming scenarios. Single-user interactive streaming applications like remote desktops might also want unfair bidirectional bandwidths. For example, remote desktop users need to deliver user instructions (e.g. program commands) and dependent files, before the remote desktop server can compute and generate the reactive screen display contents. If the uplink flow is delivered at low rates, the expected response would be further postponed. Bidirectional bandwidth coordination could prioritize the fast delivery of the uplink flows and make the interactions more seamless. We leave further study on these scenarios as our future work.

Cooperation with other congestion control algorithms. In our experiments, we integrate Plum with BBR. For other rate-based congestion control algorithms, we could also take its delivery rate as a bandwidth estimation. For congestion window (CWND)-based congestion control (e.g., CUBIC [27]), the delivery rate could be approximated by dispensing the CWND over an RTT. As we outlined in §II-C, the congestion control algorithm is supposed to work well with Plum as long as it can match the rate to the available bandwidth and timely grab increased bandwidth.

VII. Conclusion

We present Plum, a bandwidth coordination mechanism that can flexibly change the bandwidth share between the uplink and downlink streams under WLAN bottlenecks. Plum can significantly improve the performance of bidirectionally volumetric video streaming applications while maintaining user fairness. Plum is easily deployable and could be generalized to various streaming applications. With the evolution of network applications, we believe that it is important to revisit the hidden assumptions of existing network mechanisms to address the challenges arising from next-generation applications. Our paper represents a first step in this effort, by considering the half-duplexity of the WiFi bottleneck to improve the QoE for bidirectionally volumetric video streaming applications.

Acknowledgment

We sincerely thank our shepherd Ben Leong and the anonymous reviewers for their valuable feedback. This work is sponsored by the National Natural Science Foundation of China (No. 62372261 and 62221003). Bo Wang and Mingwei Xu are the corresponding authors.

REFERENCES

[1] R. T. Azuma, "A survey of augmented reality," *Presence: teleoperators & virtual environments*, vol. 6, no. 4, pp. 355–385, 1997.

[2] J. Carmigniani, B. Furht, M. Anisetti, P. Ceravolo, E. Damiani, and M. Ivkovic, "Augmented reality technologies, systems and applications," *Multimedia tools and applications*, vol. 51, pp. 341–377, 2011.

[3] K. Lee, "Augmented reality in education and training," *TechTrends*, vol. 56, pp. 13–21, 2012.

[4] C. Anthes, R. J. García-Hernández, M. Wiedemann, and D. Kranzlmüller, "State of the art of virtual reality technology," in *2016 IEEE aerospace conference*. IEEE, 2016, pp. 1–19.

[5] "Worldwide virtual reality revenue," https://vrscout.com/news/global-vr-hardware-revenue-forecaset/, 2017.

[6] R. Cheng, N. Wu, M. Varvello, S. Chen, and B. Han, "Are we ready for metaverse? a measurement study of social virtual reality platforms," in *Proceedings of the 22nd ACM Internet Measurement Conference*, 2022, pp. 504–518.

[7] S. Shi and C.-H. Hsu, "A survey of interactive remote rendering systems," *ACM Computing Surveys (CSUR)*, vol. 47, no. 4, pp. 1–29, 2015.

[8] "Zoom official website," https://explore.zoom.us/en/products/meetings/, 2023.

[9] "Google meet official website," https://meet.google.com/, 2023.

[10] "Video conferencing, meeting, calling — microsoft teams," https://www.microsoft.com/en-us/microsoft-teams/group-chat-software, 2023.

[11] S. P. Bonacci and R. Wood, "Global Economic Value of Wi-Fi_2021-2025," https://www.wi-fi.org/download.php?file=/sites/default/files/private/Global_Economic_Value_of_Wi-Fi_2021-2025_202109.pdf, 2021.

[12] "Live streaming statistics about twitch and facebook," https://www.theverge.com/2020/1/9/21058907/twitch-youtube-mixer-facebook-live-streaming-numbers-growth-q4, 2020.

[13] "Youtube live streaming official website," https://www.youtube.com/live, 2023.

[14] "Twitch official website," https://www.twitch.tv, 2023.

[15] M.-H. Chen, K.-W. Hu, I.-H. Chung, and C.-F. Chou, "Towards vr/ar multimedia content multicast over wireless lan," in *2019 16th IEEE Annual Consumer Communications & Networking Conference (CCNC)*. IEEE, 2019, pp. 1–6.

[16] "Ieee standard for information technology–telecommunications and information exchange between systems - local and metropolitan area networks–specific requirements - part 11: Wireless lan medium access control (mac) and physical layer (phy) specifications," *IEEE Std 802.11-2020 (Revision of IEEE Std 802.11-2016)*, pp. 1–4379, 2021.

[17] H. Wu, S. Cheng, Y. Peng, K. Long, and J. Ma, "Ieee 802.11 distributed coordination function (dcf): analysis and enhancement," in *2002 IEEE International Conference on Communications. Conference Proceedings. ICC 2002 (Cat. No. 02CH37333)*, vol. 1. IEEE, 2002, pp. 605–609.

[18] J. Kim, Y. Jung, H. Yeo, J. Ye, and D. Han, "Neural-enhanced live streaming: Improving live video ingest via online learning," in *Proceedings of the Annual conference of the ACM Special Interest Group on Data Communication on the applications, technologies, architectures, and protocols for computer communication*, 2020, pp. 107–125.

[19] H. Yeo, H. Lim, J. Kim, Y. Jung, J. Ye, and D. Han, "Neuroscaler: neural video enhancement at scale," in *Proceedings of the conference of the ACM special interest group on data communication*, 2022, pp. 795–811.

[20] T.-Y. Huang, R. Johari, N. McKeown, M. Trunnell, and M. Watson, "A buffer-based approach to rate adaptation: Evidence from a large video streaming service," in *Proceedings of the conference of the ACM special interest group on data communication*, 2014, pp. 187–198.

[21] J. Jiang, V. Sekar, and H. Zhang, "Improving fairness, efficiency, and stability in http-based adaptive video streaming with festive," in *Proceedings of the 8th international conference on Emerging networking experiments and technologies*, 2012, pp. 97–108.

[22] X. Yin, A. Jindal, V. Sekar, and B. Sinopoli, "A control-theoretic approach for dynamic adaptive video streaming over http," in *Proceedings of the conference of the ACM special interest group on data communication*, 2015.

[23] K. Spiteri, R. Urgaonkar, and R. K. Sitaraman, "Bola: Near-optimal bitrate adaptation for online videos," *IEEE/ACM transactions on networking*, vol. 28, no. 4, pp. 1698–1711, 2020.

[24] H. Mao, R. Netravali, and M. Alizadeh, "Neural adaptive video streaming with pensieve," in *Proceedings of the conference of the ACM special interest group on data communication*, 2017, pp. 197–210.

[25] Y. Sun, X. Yin, J. Jiang, V. Sekar, F. Lin, N. Wang, T. Liu, and B. Sinopoli, "Cs2p: Improving video bitrate selection and adaptation with data-driven throughput prediction," in *Proceedings of the conference of the ACM special interest group on data communication*, 2016.

[26] S. Sengupta, N. Ganguly, S. Chakraborty, and P. De, "Hotdash: Hotspot aware adaptive video streaming using deep reinforcement learning," in *2018 IEEE 26th International Conference on Network Protocols (ICNP)*, 2018, pp. 165–175.

[27] S. Ha, I. Rhee, and L. Xu, "Cubic: a new tcp-friendly high-speed tcp variant," *ACM SIGOPS operating systems review*, vol. 42, no. 5, pp. 64–74, 2008.

[28] N. Cardwell, Y. Cheng, C. S. Gunn, S. H. Yeganeh, and V. Jacobson, "Bbr: Congestion-based congestion control: Measuring bottleneck bandwidth and round-trip propagation time," *Queue*, vol. 14, no. 5, pp. 20–53, 2016.

[29] V. Arun and H. Balakrishnan, "Copa: Practical delay-based congestion control for the internet," in *15th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 18)*, 2018, pp. 329–342.

[30] K. Winstein, A. Sivaraman, H. Balakrishnan *et al.*, "Stochastic forecasts achieve high throughput and low delay over cellular networks." in *10th USENIX Symposium on Networked Systems Design and Implementation (NSDI 13)*, vol. 1, no. 1, 2013, pp. 2–3.

[31] M. Dong, Q. Li, D. Zarchy, P. B. Godfrey, and M. Schapira, "{PCC}: Re-architecting congestion control for consistent high performance," in *12th USENIX Symposium on Networked Systems Design and Implementation (NSDI 15)*, 2015, pp. 395–408.

[32] M. Dong, T. Meng, D. Zarchy, E. Arslan, Y. Gilad, B. Godfrey, and M. Schapira, "{PCC} vivace:{Online-Learning} congestion control," in *15th USENIX Symposium on Networked Systems Design and Implementation (NSDI 18)*, 2018, pp. 343–356.

[33] T. Hoßfeld, P. E. Heegaard, L. Skorin-Kapov, and M. Varela, "No silver bullet: Qoe metrics, qoe fairness, and user diversity in the context of qoe management," in *2017 Ninth International Conference on Quality of Multimedia Experience (QoMEX)*, 2017, pp. 1–6.

[34] D. Zhang, B. Han, P. Pathak, and H. Wang, "Innovating multi-user volumetric video streaming through cross-layer design," in *Proceedings of the Twentieth ACM Workshop on Hot Topics in Networks*, 2021, pp. 16–22.

[35] "Vrchat live player count & statistics 2024," https://playercounter.com/vrchat/, 2024.

[36] castr, "Unleash the power of 4k streaming bandwidth," https://castr.com/blog/4k-streaming-bandwidth/#:~:text=To%20stream%204K%20video%20seamlessly%20without%20buffering%20issues%2C,speeds%20of%2050%20Mbps%20or%20higher%20are%20suggested., 2023.

[37] Speedtest, "Speedtest global index," https://www.speedtest.net/global-index, 2024.

[38] V. Kawadia and P. R. Kumar, "A cautionary perspective on cross-layer design," *IEEE Wireless communications*, vol. 12, no. 1, pp. 3–11, 2005.

[39] M. Chiang, S. H. Low, A. R. Calderbank, and J. C. Doyle, "Layering as optimization decomposition: A mathematical theory of network architectures," *Proceedings of the IEEE*, vol. 95, no. 1, pp. 255–312, 2007.

[40] "Proofs and the evolution of wifi fairness notions," https://github.com/PlumRateControl/supplementary-materials, 2024.

[41] "Iperf - the ultimate speed test tool for tcp, udp and sctp," https://iperf.fr/iperf-doc.php, 2023.

[42] S. Shalunov, G. Hazel, J. Iyengar, and M. Kühlewind, "Low Extra Delay Background Transport (LEDBAT)," RFC 6817, Dec. 2012. [Online]. Available: https://www.rfc-editor.org/info/rfc6817

[43] T. Meng, N. R. Schiff, P. B. Godfrey, and M. Schapira, "Pcc proteus: Scavenger transport and beyond," in *Proceedings of the Annual conference of the ACM Special Interest Group on Data Communication on the applications, technologies, architectures, and protocols for computer communication*, 2020, pp. 615–631.

[44] T. Li, K. Zheng, K. Xu, R. A. Jadhav, T. Xiong, K. Winstein, and K. Tan, "Tack: Improving wireless transport performance by taming acknowledgments," in *Proceedings of the Annual conference of the ACM Special Interest Group on Data Communication on the applications, technologies, architectures, and protocols for computer communication*, 2020, pp. 15–30.

[45] V. Nathan, V. Sivaraman, R. Addanki, M. Khani, P. Goyal, and M. Alizadeh, "End-to-end transport for video qoe fairness," in *Proceedings of the ACM Special Interest Group on Data Communication*, 2019, pp. 408–423.

[46] D. P. Bertsekas, *Constrained optimization and Lagrange multiplier methods*. Academic press, 2014.

[47] M. French, *General Conditions for Solving Optimization Problems: Karush-Kuhn-Tucker Conditions*. Cham: Springer International Publishing, 2018, pp. 143–157. [Online]. Available: https://doi.org/10.1007/978-3-319-76192-3_6

[48] D. Kraft, "A software package for sequential quadratic programming," *Forschungsbericht- Deutsche Forschungs- und Versuchsanstalt fur Luft- und Raumfahrt*, 1988.

[49] "scipy.optimize.minimize — scipy v1.11.3 manual," https://docs.scipy.org/doc/scipy/reference/optimize.minimize-slsqp.html, 2023.

[50] Z. Meng, X. Kong, J. Chen, B. Wang, M. Xu, R. Han, H. Liu, V. Arun, H. Hu, and X. Wei, "Hairpin: Rethinking packet loss recovery in edge-based interactive video streaming," in *21st {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 24)*, 2024.

[51] Z. Meng, Y. Guo, C. Sun, B. Wang, J. Sherry, H. H. Liu, and M. Xu, "Achieving consistent low latency for wireless real-time communications with the shortest control loop," in *Proceedings of the conference of the ACM special interest group on data communication*, 2022, pp. 193–206.

[52] A. Cohen, "Netflix is 'killing' my other streaming services," https://www.compiralabs.com/post/netflix-is-killing-my-other-streaming-services, 2020.

[53] B. Grozev, L. Marinov, V. Singh, and E. Ivov, "Last n: relevance-based selectivity for forwarding video in multimedia conferences," in *Proceedings of the 25th ACM Workshop on Network and Operating Systems Support for Digital Audio and Video*, 2015, pp. 19–24.

[54] M. Lacage and T. R. Henderson, "Yet another network simulator," in *Proceedings of the 2006 Workshop on ns-3*, 2006, pp. 12–es.

[55] ITU, "Itu-r sm.1896-1: Frequency ranges for global or regional harmonization," https://www.itu.int/dms_pubrec/itu-r/rec/sm/R-REC-SM.1896-1-201809-I!!PDF-E.pdf, 2018.

[56] "Minstrel linux wireless," https://wireless.wiki.kernel.org/en/developers/documentation/mac80211/ratecontrol/minstrel, 2016.

[57] R. K. Jain, D.-M. W. Chiu, W. R. Hawe *et al.*, "A quantitative measure of fairness and discrimination," *Eastern Research Laboratory, Digital Equipment Corporation, Hudson, MA*, vol. 21, 1984.

[58] S. Fouladi, J. Emmons, E. Orbay, C. Wu, R. S. Wahby, and K. Winstein, "Salsify: Low-latency network video through tighter integration between a video codec and a transport protocol," in *15th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 18)*, 2018, pp. 267–282.

[59] "Alfalfa — a vp8 encoder and decoder," https://github.com/excamera/alfalfa, 2021.

[60] T. Høiland-Jørgensen, M. Kazior, D. Täht, P. Hurtig, and A. Brunström, "Ending the anomaly: Achieving low latency and airtime fairness in wifi." in *USENIX Annual Technical Conference*, 2017, pp. 139–151.

[61] G. Holland, N. Vaidya, and P. Bahl, "A rate-adaptive mac protocol for multi-hop wireless networks," in *Proceedings of the 7th annual international conference on Mobile computing and networking*, 2001, pp. 236–251.

[62] B. Sadeghi, V. Kanodia, A. Sabharwal, and E. Knightly, "Opportunistic media access for multirate ad hoc networks," in *Proceedings of the 8th annual international conference on Mobile computing and networking*, 2002, pp. 24–35.

[63] D. Qiao, S. Choi, and K. G. Shin, "Goodput analysis and link adaptation for ieee 802.11 a wireless lans," *IEEE transactions on Mobile Computing*, vol. 1, no. 4, pp. 278–292, 2002.

[64] I. Haratcherev, K. Langendoen, R. Lagendijk, and H. Sips, "Hybrid rate control for ieee 802.11," in *Proceedings of the second international workshop on Mobility management & wireless access protocols*, 2004, pp. 10–18.

[65] M. Lacage, M. H. Manshaei, and T. Turletti, "Ieee 802.11 rate adaptation: a practical approach," in *Proceedings of the 7th ACM international symposium on Modeling, analysis and simulation of wireless and mobile systems*, 2004, pp. 126–134.

[66] J. Kim, S. Kim, S. Choi, and D. Qiao, "Cara: Collision-aware rate adaptation for ieee 802.11 wlans." in *Infocom*, vol. 6, 2006, pp. 1–11.

[67] S. H. Wong, H. Yang, S. Lu, and V. Bharghavan, "Robust rate adaptation for 802.11 wireless networks," in *Proceedings of the 12th annual international conference on Mobile computing and networking*, 2006, pp. 146–157.

[68] "Ieee standard for information technology–local and metropolitan area networks–specific requirements–part 11: Wireless lan medium access control (mac) and physical layer (phy) specifications - amendment 8: Medium access control (mac) quality of service enhancements," *IEEE Std 802.11e-2005 (Amendment to IEEE Std 802.11, 1999 Edition (Reaff 2003)*, pp. 1–212, 2005.

[69] H. Wu, X. Wang, Q. Zhang, and X. Shen, "Ieee 802.11e enhanced distributed channel access (edca) throughput analysis," in *2006 IEEE International Conference on Communications*, vol. 1, 2006, pp. 223–228.

[70] N. S. Nandiraju, H. Gossain, D. Cavalcanti, K. R. Chowdhury, and D. P. Agrawal, "Achieving fairness in wireless lans by enhanced ieee 802.11 dcf," in *2006 IEEE International Conference on Wireless and Mobile Computing, Networking and Communications*. IEEE, 2006, pp. 132–139.

[71] J. Jeong, S. Choi, and C.-k. Kim, "Achieving weighted fairness between uplink and downlink in ieee 802.11 dcf-based wlans," in *Second International Conference on Quality of Service in Heterogeneous Wired/Wireless Networks (QSHINE'05)*. IEEE, 2005, pp. 10–pp.

[72] M. Bottigliengo, C. Casetti, C.-F. Chiasserini, and M. Meo, "Smart traffic scheduling in 802.11 wlans with access point," in *2003 IEEE 58th Vehicular Technology Conference. VTC 2003-Fall (IEEE Cat. No. 03CH37484)*, vol. 4. IEEE, 2003, pp. 2227–2231.