CONVERGE: QoE-driven Multipath Video Conferencing over WebRTC

Sandesh Dhawaskar Sathyanarayana*, Kyunghan Lee§, Dirk Grunwald*, Sangtae Ha*
University of Colorado Boulder*, Seoul National University§

ABSTRACT

Video conferencing has become a daily necessity, but protocols to support video conferencing have yet to keep pace despite the innovation in next-generation networks. As video resolutions increase and mobile applications using multiple cameras for photos and videos become popular, the need to meet the Quality of Experience (QoE) requirements is growing. Multipath protocols could be a possible solution.

In this paper, we show that a straightforward extension of WebRTC for supporting multipath can perform worse than legacy WebRTC and propose CONVERGE, a WebRTC-compliant multipath video conferencing platform. CONVERGE improves QoE through three main components: a video-aware scheduler, video QoE feedback, and video-aware and path-specific packet protection. The video-aware scheduler uses the real-time video structure to schedule packets, and the video QoE feedback from the receiver helps the scheduler adjust the number of packets on each path. Additionally, the video-aware and path-specific packet protection mechanism improves on the existing FEC mechanism in WebRTC by considering the trade-off between FEC and QoE. CONVERGE is built as part of the Chromium browser, making it compatible with any device or network path. CONVERGE improves overall media throughput by 1.2×, reduces end-to-end latency by 20%, and enhances image quality by 55% compared to WebRTC.

CCS CONCEPTS

• **Networks** → **Network protocols**; **Cross-layer protocols**;

KEYWORDS

Multipath, Video Conferencing, QoE, WebRTC

ACM Reference Format:
Sandesh Dhawaskar Sathyanarayana*, Kyunghan Lee§, Dirk Grunwald*, Sangtae Ha* . 2023. CONVERGE: QoE-driven Multipath Video Conferencing over WebRTC. In *ACM SIGCOMM 2023 Conference (ACM SIGCOMM '23), September 10–14, 2023, New York, NY, USA.* ACM, New York, NY, USA, 17 pages. https://doi.org/10.1145/3603269.3604822

1 INTRODUCTION

Real-time video conferencing has been a popular internet application since the 1990s [2, 4, 5, 18, 19, 22, 49, 73] and continues to grow rapidly. Advances in mobile devices with multiple high-resolution cameras [42] and the rise of remote work, teaching, and socializing have contributed to this growth. According to Forbes, 25% of professional jobs in North America are expected to be remote, and this trend is likely to continue through 2023 [68].

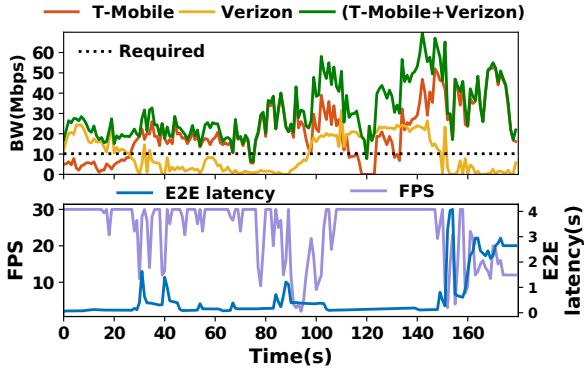


Figure 1: WebRTC performance degrades (bottom) due to variations in cellular bandwidth (BW) with mobility (top)

New applications, such as Dualgram [38], Delfieart [25], and Duovision [23], which use multiple cameras for a more immersive experience, are also growing in popularity. However, these bandwidth-intensive applications may stress current wireless networks and can result in lower-quality video conferencing. While next-generation networks like 5G/6G and WiFi6/7 offer more network capacity [1, 50, 57, 82, 85, 86], video conferencing applications still experience poor performance [39, 56, 70, 74], including higher latency and frame drops [29, 30, 47, 51].

We conducted two video calls with WebRTC, the standard platform for web-based real-time streaming applications [35], using four devices simultaneously. One call was made over T-Mobile and the other over Verizon, with Full-HD resolution and a maximum encoding rate of 10 Mbps. As we see in Figure 1, variations in frames per second (FPS) and per-frame end-to-end latency (E2E) caused interruptions in the call, leading to a poor QoE for users.

The majority of modern mobile devices have multiple connections, such as dual-SIM [10, 11, 46, 59, 64] or WiFi with cellular [61], and aggregating these connections is a potential solution to provide better QoE and meet the growing bandwidth demands of video conferencing applications. As seen in Figure 1, when one network cannot provide the minimum required bandwidth, the other network can compensate.

In this paper, we propose a new conferencing system called CONVERGE that addresses the challenges of multipath video conferencing and improves the user QoE. CONVERGE utilizes a video-aware scheduler, video QoE feedback, and video-aware and path-specific packet protection to achieve this improvement. CONVERGE

ACM SIGCOMM '23, September 10–14, 2023, New York, NY, USA

ACM ISBN 979-8-4007-0236-5/23/09.
https://doi.org/10.1145/3603269.3604822

addresses the specific needs of video conferencing applications, unlike existing solutions such as MPTCP [8, 65, 79], MPRTP [71], and MPQUIC [12, 13], which are not tailored for this purpose.

Video-aware scheduler. The conventional packet scheduler lacks the knowledge of the intricacies involved in encoding and decoding video frames, which leads to the disruption of decoding sequences and, ultimately, negatively affects the user experience during video conferencing with frequent pauses and frame drops. The video-aware scheduler in Converge addresses this issue by utilizing the real-time video information from the encoder to schedule packets. By understanding which packet belongs to which video frame and its significance and dependencies, the scheduler assigns packets over the path that optimizes the QoE at the receiver. The scheduler also continuously adjusts the number of packets sent over each path based on the video QoE feedback received from the receiver, ensuring that the assignment of packets and rate control is based on the video construction at the receiver application, not just the network conditions.

Video QoE feedback. Converge learns from the buffering and video frame construction process at the receiver to provide crucial video QoE feedback to the sender. The receiver monitors the video frame construction to detect any deterioration in the video QoE caused by path asymmetries. Since path asymmetry is inevitable in multipath scenarios, it is essential to identify when it begins to affect the video QoE at the receiver. The video QoE feedback from the receiver enables the sender to understand the impact of multipath on the video QoE.

Video-aware and path-specific packet protection. Protecting against packet loss is critical for achieving better video QoE, particularly in dynamic network conditions. However, relying solely on Forward Error Correction (FEC) to decode frames can increase E2E latency [54], which negatively impacts QoE. In mobile networks, WebRTC can send approximately 25% excess FEC packets compared to media packets, but our findings indicate that only around 10-25% of those FEC packets are actually utilized. Converge tackles this challenge by employing video-aware and path-specific FEC mechanisms based on real-time video QoE feedback from the receiver. This approach takes into account the impact of path characteristics on video frame decoding, outperforming WebRTC's static, table-based FEC selection.

We developed Converge as a user-space conferencing system to ensure compatibility with a wide range of devices and standard protocols for real-time streaming, such as Real-time Transport Protocol (RTP) and Real-time Transport Control Protocol (RTCP). In particular, we built Converge on top of WebRTC, a de-facto standard for web-based real-time video conferencing systems, to reduce deployment costs. Converge supports multipath and multiple cameras, and it seamlessly falls back to the standard WebRTC protocols if either endpoint does not support multipath, ensuring backward compatibility with many existing WebRTC-based video conferencing applications.

We evaluated Converge in both emulated and real-world settings for conferencing scenarios involving up to three camera streams. Our results demonstrate that Converge delivers a high-quality video conferencing experience with improved QoE and uninterrupted calls, even in challenging network conditions.

In summary, our paper makes the following contributions:

- We report that the existing multipath scheduling mechanisms in MPTCP, MPRTP, and MPQUIC are unsuitable for video conferencing, and their performance is far from satisfactory. We identify several key challenges associated with real-time video conferencing applications over multipath. To the best of our knowledge, this is the first comparative study to support multipath for video conferencing applications (§2.3).
- We then design (§3) and build (§4) Converge by creating a closed loop between the video QoE feedback, which indicates deteriorations in QoE due to multipath at the receiver, and a video-aware scheduler that assigns and reassigns packets on each path. Converge also incorporates loss protection based on the receiver's feedback on each packet's importance and dependencies in video frames.
- We implement Converge as an integral part of the WebRTC code base, which consists of 16GB of code with over 300K files, by adding or modifying more than 250 files. This makes Converge a practical video conferencing system that seamlessly supports multipath and multiple cameras. Furthermore, Converge is backward compatible with any device and legacy application using WebRTC in the browser or native applications (§5).
- Compared to WebRTC, Converge improves overall media throughput by 1.2×, reduces E2E latency by 20%, enhances image quantization by 55%, and reduces video freezes by 52%. In real-world testing, Converge ensures a minimum FPS of 24, resulting in superior video QoE (§6).

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

2 BACKGROUND

This section starts by presenting an overview of WebRTC as a video conferencing system (§2.1). We then delve into the discussion on the three existing multipath protocol schedulers that can be used for multipath video conferencing (§2.2). Lastly, we demonstrate how these schedulers can negatively impact performance and identify the challenges associated with achieving better performance based on our observations (§2.3).

2.1 Overview of WebRTC

WebRTC has three components: the sender, network controller, and receiver, as shown in Figure 2. The sender takes the inferred rate from the network controller and encodes video frames captured by the camera at that rate. These encoded video frames are packetized into RTP packets and transmitted. The receiver receives the RTP packets and utilizes two buffers to generate video frames. Additionally, the receiver also sends loss and delay reports to the sender using RTCP packets. The network controller then analyzes these reports and adjusts the encoding rate for the sender accordingly, and this process repeats.

Network controller. WebRTC utilizes the Google Congestion Control (GCC) [6, 7, 87] algorithm to estimate the network path dynamics, such as available bandwidth. The sender's GCC uses loss and delay reports from the receiver to calculate two rates. The sender then uses the lower of the two rates for encoding to minimize congestion. GCC calculates these rates whenever feedback is

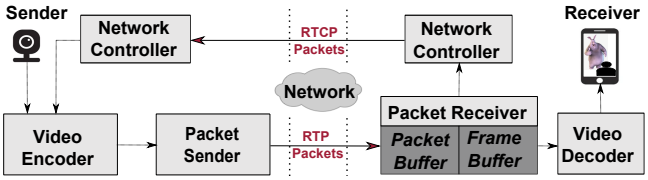


Figure 2: WebRTC system architecture

received from the receiver, with the feedback frequency dependent on the path bandwidth.

Encoding and decoding pipeline. At the sender, the encoder converts the raw image frames from the camera into video frames at the rate determined by the network controller (Figure 2). Two types of video frames exist: keyframe (I-frame) and delta frame. Keyframes contain complete frame information, while delta frames capture the changes from the keyframes. These frames are packetized as RTP packets and transmitted over the network. Most RTP packets carry media data, while a few carry control information. For example, Picture Parameter Set (PPS) RTP packets contain decoding information for a frame, while Sequence Parameter Set (SPS) RTP packets hold decoding information for a group of frames. Even though the receiver receives all media data, without these control packets, a while frame or a group of frames becomes non-decodable at the receiver. The sender also generates FEC packets to protect RTP packets from loss. WebRTC uses an XOR-based FEC mechanism [31]. Note that FEC decoding incurs non-negligible latency in the decoding pipeline at the receiver [27, 55, 60].

Receive buffers in WebRTC. The receiver employs two receive buffers: the packet buffer and the frame buffer. The packet buffer has a size limit and accumulates all RTP packets belonging to a specific frame to construct the frame. If any packet related to a frame never arrives or arrives too late, the packet buffer may discard packets from that frame to make room for newly arriving packets. We call the total time required to gather all the packets of a given frame the *gathering delay*, and this delay can increase if there are FEC packets to be processed. Once a frame is constructed, it is pushed into the frame buffer. The frame buffer also has a size limit and can purge old frames if there is insufficient space for incoming frames. Furthermore, the frame buffer can also drop packets in the packet buffer if they belong to missing and purged frames. The frame buffer collects frames and pushes them to the decoder. We call the inter-arrival time between frames in the frame buffer the *interframe delay*. In §3, we will explain how we utilize the gathering delay and interframe delay in our system.

Video QoE and parameters. The Quality of Experience (QoE) in video conferencing is determined by objective quality metrics, including frame rate, freeze duration, E2E latency, media throughput, and image quality [14, 21, 36, 58, 66, 88]. These metrics are used to measure and evaluate visual quality, and we utilize the same parameters to define video QoE in Converge. Two intermediate parameters, namely gathering delay and interframe delay, have a significant impact on video QoE [75]. These parameters are affected by the receiver buffers during the frame construction process at the receiver, as previously explained. In §3.2, we will provide further details on how these intermediate parameters play a crucial role in mitigating potential QoE degradation caused by multipath effects.

2.2 Existing Multipath Protocols

WebRTC, by itself, can not exploit multiple network paths. However, there are several multipath transport protocols available. **MPTCP** [65] is the first multipath TCP protocol practically deployed. The scheduler of MPTCP distributes packets across available paths, with the default scheduler being minRTT, which prefers the path with minimum RTT over others. **MPQUIC** [13] is a user-space multipath extension of the QUIC protocol. MPQUIC uses the default minRTT scheduler as MPTCP. Both MPTCP and MPQUIC have been applied to video streaming [15, 28, 63, 70, 76, 81, 89], which is less sensitive to latency than video conferencing. **MPRTP** [71] is a UDP-based multipath extension to RTP. MPRTP is used for real-time media communication that is sensitive to latency. However, the MPRTP protocol does not provide any feedback and uses all available paths to distribute packets based on their loss characteristics.

In multipath transport protocols, including the ones mentioned above, the heterogeneous nature of path properties when using multiple paths can lead to head-of-line blocking (HoL) issues that quickly impact multipath performance. Several recent works [17, 48, 69, 70] have addressed these issues.

We implemented multipath WebRTC variants using three schedulers: Musher [69], minRTT [62], and MPRTP [71] to assess their ability to meet the QoE requirements for video conferencing. These schedulers are used in multipath protocols like MPTCP, MPQUIC, and MPRTP. We refer to these variants as M-TPUT, SRTT, and M-RTP, respectively. The summary of these protocols is presented in §7, and implementation details are provided in §5.

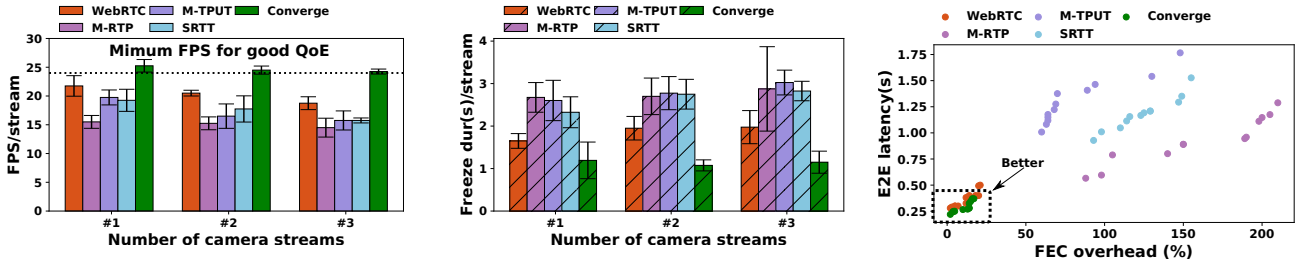
2.3 Multipath Is Not Enough

We conducted experiments on video conferencing, involving up to three camera streams, using the multipath WebRTC variants described in the previous section. These experiments were performed in an emulated network using traces collected from two cellular carriers (T-Mobile and Verizon) while driving along the highway, similar to the scenario depicted in Figure 1. The maximum encoding rate of each camera stream is 10 Mbps with Full HD resolution.

As shown in Figure 3, it is clear that standard WebRTC fails to provide uninterrupted video conferencing with satisfactory QoE for both T-Mobile and Verizon. Moreover, the QoE deteriorates as the number of camera streams increases. The use of multipath was expected to solve this issue by utilizing the aggregate bandwidth. However, we found that using schedulers not designed for real-time video conferencing resulted in poor performance. On the other hand, Converge consistently delivers better QoE, irrespective of the number of camera streams used.

The recommended FPS for good video QoE is 24 [67]. However, WebRTC was unable to achieve this target, and although we hoped that multipath WebRTC variants could achieve this FPS, they dropped FPS below the performance of single-path WebRTC and continued to decline with additional camera streams. We expected that video QoE would improve with multipath compared to WebRTC. However, as shown in Figure 3(b), the average freeze duration is larger than that of WebRTC, and it increases as the number of camera streams increases.

Our detailed analysis indicated that all multipath WebRTC variants dropped a significant number of frames and requested more



(a) **Video FPS** (b) **Average freeze duration** (c) **E2E latency & FEC overhead**

Figure 3: WebRTC and Multipath WebRTC variants' performance degrades compared to Converge

Average number of frame drops					
# streams	**WebRTC**	**M-RTP**	**M-TPUT**	**SRTT**	**Converge**
1	146± 38	2269± 213	1420 ± 235	1148±127	154±30
2	102 ± 95	2082± 428	949±291	1125± 269	104±53
3	101 ± 10	1995±220	794± 86	843± 74	143±29
Total number of keyframe requests					
1	3± 1	7± 1	6± 1	7±1	1± 1
2	3± 1	6± 1	6±1	6± 1	1± 1
3	2 ± 1	6±1	6± 1	7± 1	2± 1

Table 1: Performance degradation of WebRTC and multipath-WebRTC variants compared to Converge

keyframes compared to WebRTC, as shown in Table 1. This explains why we observed lower FPS and more video freezes in Figure 3. Packets arriving out of order disrupted the decoding sequence of video frames at the receiver, leading to their dropping instead of being sent to the decoder. Therefore, the packet scheduler at the sender needs to consider the video structure when multipath is used. Furthermore, the receiver only sends network performance metrics, such as delay and loss reports for each path, which are unrelated to the user's QoE in terms of how well video frames are decoded at the receiver. In this case, incorporating QoE feedback from the receiver could help the sender adjust its packet scheduling decisions, considering the inevitable path asymmetry in multipath. Converge utilizes a video-aware scheduler with a QoE feedback mechanism, effectively avoiding frame drops and providing better FPS with minimum video freezes.

Additionally, we observed excessive FEC overhead when utilizing multiple paths. As shown in Figure 3(c), the FEC overhead (ratio of FEC to media packets) of multipath WebRTC variants is high, reaching or exceeding 60%. This excessive overhead takes away video bandwidth and incurs significant processing delays, resulting in elevated E2E latency compared to WebRTC (>500 ms), without fully utilizing the available bandwidth. Converge adopts a path-specific approach to determine the FEC rate based on path loss, effectively mitigating the increase in E2E latency.

To enable effective multipath video conferencing, Converge addresses three major challenges:

- **C1:** The multipath scheduler at the sender lacks information about the structure of the encoded video frames, such as which packets belong to which frames and their relative importance. This limited information prevents the scheduler from selecting the best path for each packet to improve video QoE at the receiver. How can we leverage the information of video frames to schedule packets on each path to improve the video QoE at the receiver?
- **C2:** Latency and packet loss asymmetry is inevitable in multipath networks and can lead to packet or frame drops in the receive buffers, which can impact video QoE at the receiver. The current feedback mechanism carried over RTCP packets is insufficient for the sender to make informed decisions about scheduling packets over multiple paths. How can we develop a feedback mechanism that captures changes in video QoE at the receiver, in addition to the network performance metrics provided by RTCP feedback?
- **C3:** Packet protection through FEC is crucial, particularly in rapidly changing networks. However, protection packets can reduce the available bandwidth for media packets. Therefore, deciding the optimal protection rate for video frames by considering their importance and the QoE tradeoff is critical. How can we protect video frames on each path with the minimum possible overhead to improve the overall video QoE in a multipath scenario?

3 DESIGN OF CONVERGE

This section presents three crucial designs that address the challenges outlined in the previous section. Many of the resulting decisions affect which path is used for specific packets among the multiple available paths. For simplicity, we will use the terms "fast path" and "slow path" to distinguish the path that, at a given point in time, is more or less able to reliably deliver packets with sufficiently low latency. Later, we will describe how we determine which paths are considered "fast" or "slow".

3.1 Video-aware Packet Scheduling

Different types of frames and packets have different priorities and determine the real-time sensitivity of video [26]. The QoE can quickly deteriorate if this video construct is disturbed when multipath is used. To prevent this, we design three levels of control in the scheduler based on the priorities and dependencies of packets in video frames.

Frame-level control. A "keyframe" carries the complete information of the frame of a scene and can be decoded independently, while delta frames become unusable without their associated keyframes being decodable at the receiver. This provides a guideline for how the multipath scheduler prioritizes packets from different frames over multipath.

In Figure 4, F_{i-1} is the keyframe, and F_i and F_{i+1} are delta frames. Suppose two packets, 3 and 4, of F_{i-1} are scheduled on the slow path, and packets of F_i and F_{i+1} are on the fast path. If packet 3

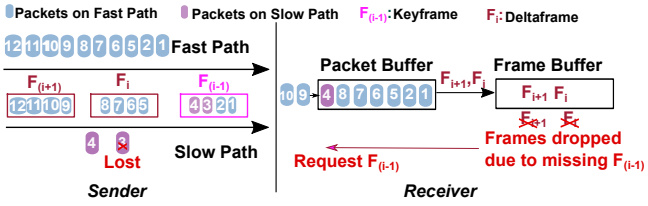


Figure 4: Impact of scheduling keyframe packets (3 and 4) on the slow path

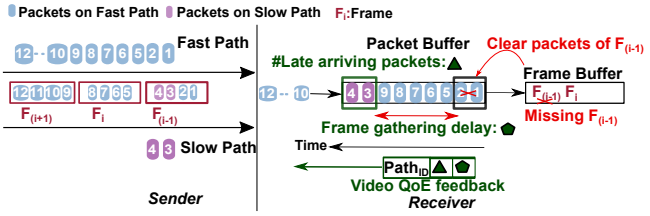


Figure 5: QoE feedback carrying information about the path, # of late packets, and frame gathering delay

is lost, but the packets of successive delta frames F_i and F_{i+1} are received via the fast path, an issue arises. As the packets of F_i and F_{i+1} are received, they are accumulated and pushed into the frame buffer. However, the same cannot be done for F_{i-1} due to the missing packet 3. Since the decoding of F_i and F_{i+1} depends on F_{i-1}, these two frames F_i and F_{i+1} are purged from the receiver buffer, and the keyframe F_{i-1} is requested from the sender. Until the keyframe is received, the video will freeze, negatively impacting user QoE.

The freezing of the video could be prevented if the scheduler had knowledge that F_{i-1} is the keyframe and is required for decoding its subsequent delta frames. In that case, packets 3 and 4 could be scheduled on the fast path, while packets from F_i or F_{i+1} could be scheduled on the slow path. This video-aware or video structure-aware scheduling would prevent video freezing due to missing keyframes and improve the QoE for users at the receiver. We incorporate this video dependency into our design when the scheduler needs to place packets over multipath.

Packet-level control. After the raw video is encoded into keyframes and delta frames, these frames are transmitted as RTP packets through packetization. In addition to the inter-frame dependency, there are dependencies within the frame. When the video frames are packetized into RTP packets, essential information for successfully decoding the frame or group of frames is distributed across two different RTP packets. These packets are included in both keyframes and delta frames. The PPS packet is necessary for each keyframe or delta frame, while a group of delta frames requires the SPS packet. Since keyframes, PPS packets, and SPS packets are critical for decoding, we need to ensure their successful reception.

In Figure 5, suppose packet 3 is the PPS packet and packet 4 is the SPS packet, both scheduled on the slow path and experiencing loss or delay. The decoding of frame F_{i-1} is affected due to the lost packet 3, and the subsequent frames F_i and F_{i+1} are impacted by the delayed packet 4. This delay prompts the receiver to request a keyframe, as the decoding of F_{i-1} fails, resulting in video freezing and negatively impacting the user's QoE.

If the scheduler is aware of the importance of these packets (3 and 4) and selects the fast path instead of the slow one, the QoE drop could have been avoided. The scheduler might not be able to allocate all the packets on the fast path, but placing at least the essential RTP packets on the fast path can prevent frame drops caused by decoding dependencies. We employ this principle when developing a video-aware scheduler.

Reliability-level control: Even though the packet scheduler can make optimal decisions regarding packet placement on each path using video awareness and detailed feedback from the receiver,

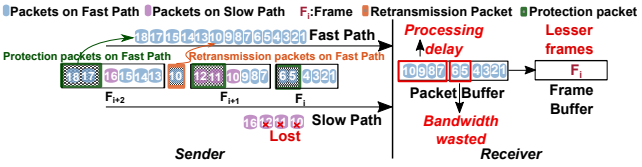


Figure 6: Impact of scheduling protection packets (11 and 12) on the slow path on video QoE

packet loss can still occur. Since packet loss can significantly impact the user's QoE, protection mechanisms such as FEC and NACK are employed to mitigate packet loss. FEC packets are generated proactively at the sender to recover lost packets. As these packets are transmitted and decoded with media packets, they require additional network bandwidth to send and introduce processing delays at the receiver. On the other hand, NACK is sent by the receiver to the sender to request the retransmission of a specific packet.

In Figure 6, frames (F_i, F_{i+1}, and F_{i+2}) have two protection packets in addition to the four media packets, resulting in a 33% reduction in the quality of each frame. This reduction occurs because both media and FEC packets are subject to the bandwidth constraints of the network paths. In this scenario, the scheduler assigns media packet 10 and protection packets 11 and 12 of frame F_{i+1} to a slow path that experiences packet loss. As a result, frame F_{i+1} cannot be reconstructed at the receiver due to the insufficient number of packets required for decoding, leading to wasted bandwidth. Only frame F_i is successfully processed and stored in the frame buffer.

If the scheduler had placed packets 11 and 12 on a fast path with a lower packet loss rate, frame F_{i+1} could have been reconstructed at the receiver. A similar approach is applied to frame F_{i+2}, with its media packet 16 assigned to the slow path and protection packets assigned to the fast path. The protection packets from the fast path would assist in recovering packet 16 in case it gets lost. The same principle can be extended to the retransmitted RTP packet 10 as requested by NACK. Placing this packet on the fast path would contribute to the successful reconstruction of the frame, resulting in a higher frame rate.

In §4.1, we provide a detailed description of our video-aware scheduler, which leverages information about frame and packet types to enhance the user's QoE across multiple paths.

3.2 Multipath-aware QoE Feedback

Understanding the impact of multipath on the receiver's QoE is crucial for effective video-aware packet scheduling at the sender.

Drops in the receive buffers due to multipath asymmetry. As explained in §2.1, the receiver utilizes two buffers: the packet

buffer and the frame buffer, both of which have limited sizes. When multipath is used, path asymmetry causes drops in these buffers, leading to increased frame gathering delay and interframe delay. In Figure 5, the first two packets, 1 and 2, of frame F_{i-1} take the fast path, while packets 3 and 4 take the slow path. These packets arrive late in the packet buffer. Until packets 3 and 4 arrive, frame F_{i-1} cannot be moved to the frame buffer. However, packets of F_i are received and pushed into the frame buffer. As packets of frame F_{i+1} continue to arrive, the packet buffer must drop a few packets to make space for the newly arriving ones. If packets 1 and 2 are chosen for dropping, frame F_{i-1} is never constructed. When the frame buffer realizes that frame F_{i-1} is missing but more incoming frames, F_i and F_{i+1}, are present, it drops the packets of frame F_{i-1} from the packet buffer. Although all packets of frame F_{i-1} eventually arrive in the packet buffer, the frame is never built and, consequently, not pushed into the frame buffer. The absence of packets in the packet buffer increases the frame gathering delay in the packet buffer and the interframe delay in the frame buffer.

Feedback to avoid the QoE drop over multipath. To mitigate drops in the buffers, it is necessary to reduce the level of path asymmetry. As depicted in Figure 5, we can track the order and arrival times of packets to determine the frame gathering delay, interframe delay, and the number of late-arriving packets that caused drops in the packet buffer. Both late and lost packets negatively impact decoding performance, affecting the FPS and, subsequently, the QoE. Therefore, we provide this QoE feedback to the sender, including this information along with the path responsible for the packet drops. Upon receiving this feedback, the sender reduces the number of in-flight packets allowed to be scheduled on the path that caused the drops.

3.3 Protection for QoE over Multipath

Protection of packets against loss comes at the cost of QoE. Protected packets deprive the bandwidth of video frames and incur additional decoding delays. On the other hand, fewer protection packets can significantly drop the QoE due to missing frames or packets at the receiver. We introduce two types of protections: path-specific and video-specific.

Path-specific protection: We adopt path-specific protection, where FEC is generated specifically for each path, as opposed to application-level protection where FEC is based on the combined loss across all paths. The aggregated loss of application-level protection can result in unnecessary overhead, leading to higher video QoE tradeoff costs. Consider an example where the loss rate of two available paths, P1 and P2, is 0% and 10%, respectively, and the total number of media packets is 60. Suppose the scheduler splits 30 packets to P1 and the remaining 30 to P2. With the aggregate loss of application-level protection, around 6 FEC packets would be generated, whereas with a path-specific approach, only 3 FEC packets would be required. Furthermore, the video QoE feedback from the receiver helps in reducing the proportion of FEC packets. If there are 20 late-arriving packets due to path asymmetry, the sender can utilize this feedback to allocate 20 packets to P1, thus requiring only a single FEC packet for the remaining 10 packets on P2. This path-based FEC mechanism, combined with video QoE feedback, allows for a more effective tradeoff between FEC rate and QoE compared to the static table-based FEC mechanism used in WebRTC.

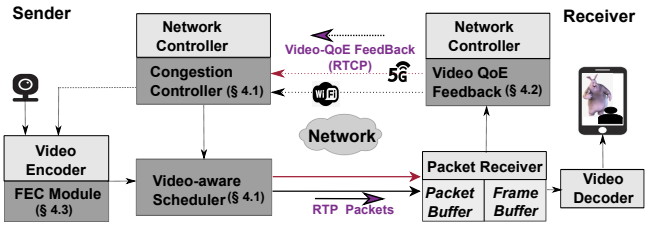


Figure 7: Converge **architecture**

Protection based on the video structure. We have observed how the dependency on video frames affects the QoE in video conferencing. The existing FEC mechanism in standard WebRTC applies the same FEC rate to different packets, except for doubling the FEC rates for keyframes. However, this approach is often ineffective with multipath. Let's consider a scenario with paths P1 and P2, where P1 has a 5% loss rate, P2 has a 10% loss rate, and P2 has a significantly higher bandwidth. In this case, the sender's scheduler can prioritize protecting packets of higher priority with higher FEC rates on path P1. If the receiver's feedback indicates that path P2 is not significantly impacting the QoE, the scheduler can still send keyframes with higher FEC rates (despite their larger size) over path P2 due to its higher loss rate. Conversely, if the loss rate of P1 becomes negligible, the scheduler can send keyframes on path P1 and delta frames on path P2 to ensure that users do not experience a drop in QoE. Keyframes delivered over the path with lower loss rate will enable uninterrupted video playback. In §4, we will provide a detailed explanation on the FEC mechanism used by Converge.

4 CONVERGE AS SYSTEM

This section explains the three novel components of Converge in Figure 7. The encoder uses the rate derived from the congestion control module (§4.1) to encode the raw frames from the camera stream into RTP packets of different priorities. The **video-aware scheduler** (§4.1) sends priority packets mainly on the fast path and the remaining packets on all the available paths, as long as this strategy improves the user QoE at the receiver. Since path asymmetry is inevitable in multipath, the **video QoE feedback** (§4.2) module at the receiver identifies if this asymmetry adversely affects video QoE and sends feedback back to the sender when a decline in QoE is observed. The video-aware scheduler uses this feedback to further tune the number of packets on each path. The **FEC controller** defends packets against loss (§4.3) by generating FEC packets, considering the tradeoff between quality, latency, and their impact on QoE.

4.1 Video-aware RTP Scheduler

The congestion control module provides the encoding rate for the video and helps the packet scheduler infer the fast path and sending rate for each path. The video-aware scheduler checks the priority level of each RTP packet and chooses the fast path for priority packets. Non-priority packets are distributed among the remaining paths, and the sending rate for each path is further adjusted if there is video QoE feedback specific to that path.

Priority level	Packets type
1 (highest)	Retransmitted Packets.
2	Keyframes RTP packets
3	SPS RTP packets
4	PPS RTP packets
5	FEC packets

Table 2: Priority setting of video packets

Congestion control. We use the uncoupled congestion control [41, 80] approach. Individual path statistics, such as loss and delay, are utilized to derive the encoding rate for each path. The GCC [6] algorithm, which is the congestion control mechanism in WebRTC, determines the encoding rate for the encoder as explained in §2.1. The congestion control module of Converge extends the the GCC algorithm for every available path and determines the encoding rate for the encoder. The congestion control algorithm updates its sending/encoding rate whenever there is RTCP feedback from the receiver for any of the used paths. The sending rate S_i for path i is computed based on the feedback. The aggregate rate is the sum of the rates for each path, denoted as $\sum_{i=1}^{n} S_i$. The encoder then utilizes the minimum value between this aggregate rate and the maximum rate set by the conferencing application to generate N RTP packets from the incoming frames of the camera device. These RTP packets are subsequently sent to the scheduler. It is important to note that the value of n indicates the number of active paths out of all available paths since some paths may be disabled based on the feedback. Additionally, we determine the maximum number of packets allowed for each path based on its S_i, denoted as $\boldsymbol{P_{max}}$. This value serves as the upper threshold for adjusting the number of packets at the scheduler.

Priority packets. Only the encoder knows the types and priorities of packets that carry video frames, and we expose this information to the scheduler to prevent QoE deterioration. The scheduler sends RTP packets with priorities over the fast path. We introduce five sets of priority levels in the encoder during packet generation, as shown in Table 2. Retransmitted packets (sent in response to NACK requests) have the highest priority level as they assist in packet recovery and, consequently, frame recovery. Keyframes, essential for decoding delta frames, have the second-highest priority level. This is followed by PPS and SPS packets, which define the decoding dependencies for individual frames or groups of frames. Lastly, FEC packets have the lowest priority since they are only used in the event of packet loss.

The fast path is determined based on the RTCP statistics received for each path, as shown in Algorithm 1. Given the strict time restrictions in video conferencing, we choose the path with the shortest transmission completion time as the fast path. We define the completion time $cpt_i(N) = (N * k/\text{rate}_i) + \text{rtt}_i/2$, representing the time required to transmit N packets on the i^{th} link at the measured goodput rate (which accounts for packet loss) and round-trip time. If the number of priority packets exceeds the maximum number of allowed packets P_{max} on the fast path, the remaining packets are distributed among the other paths based on their priority levels. One exception is that if FEC packets cannot be accommodated, they are sent on the path on which they were generated.

Media packets. Media packets without any priority level are distributed among the available paths in proportional to the determined sending rate for each link S_i to the total available rate, as shown in Equation 1.

Algorithm 1 Selecting Fast Path

Input: RTCP feedback of each $Path_i$.
  rtt_i: RTT of the path i.
  $rate_i$: Encoding rate of the path i in bytes/sec.
  N: Total number of generated RTP packets.
  k: The maximum RTP packet size in bytes.
Variables: cpt_i: Completion time of path i.
Output: $Path\_id$: Path ID of the fast path.
for each $Path_i$ **do**
  $cpt_i = (N * k/rate_i) + (rtt_i/2)$
end for
$path\_id = \underset{i}{argmin}\ (cpt_i)$
return $path\_id$

$$P_i = (S_i / \sum_{j=1}^{n} S_j) \times N \tag{1}$$

The path rate P_i is further adjusted to account for the impact of using multipath on video QoE, as we discussed in §3.2. If P_i becomes zero, the path will be disabled. In §4.2, we will explain how the receiver's QoE feedback α is calculated and how the paths are managed.

$$P_i = \begin{cases} P_i, & \alpha == 0, \text{No feedback} \\ \min(P_{max}, P_i + \alpha), & \alpha > 0, \text{Positive feedback} \\ \max(0, P_i + \alpha), & \alpha < 0, \text{Negative feedback} \end{cases} \tag{2}$$

Let's consider an example where path $P1$ has $rate_1 = 15Mbps$, $P2$ has $rate_2 = 5Mbps$, and 40 media packets need to be transmitted by the scheduler. There is feedback from the receiver for P_2, carrying α of 5. First, $\lceil 15/20 \times 40 \rceil = 30$ packets will be estimated for $P1$, and $\lceil 5/20 \times 40 \rceil = 10$ for $P2$. Furthermore, when the scheduler notices α for P_2, it sends 35 packets on $P1$ and 5 packets on $P2$ so that QoE deterioration is avoided with the help of feedback.

4.2 QoE-based Feedback

The QoE feedback module keeps track of the frame construction process to detect QoE deterioration caused by multipath asymmetry and sends a signal to the scheduler to prevent such incidents. The sender's scheduler then adjusts the packets on each path and enables/disables specific paths if necessary.

Video QoE parameters. We use two parameters, Frame Construction Delay (FCD) and InterFrame Delay (IFD), to represent the video QoE and measure the impact of multipath asymmetry at the receiver. FCD is the delay in gathering all the packets of a frame, and IFD is the time interval between the arrival of two consecutive frames in the frame buffer, as explained in §2.1. When a packet arrives late or is lost, the FCD increases because the frame can only be sent to the frame buffer for decoding once all its packets have arrived. As the FCD increases, the number of frames in the frame buffer decreases, leading to an increase in IFD. An increased IFD indicates QoE deterioration due to a lower frame rate. Therefore, we use these parameters to build the video QoE feedback.

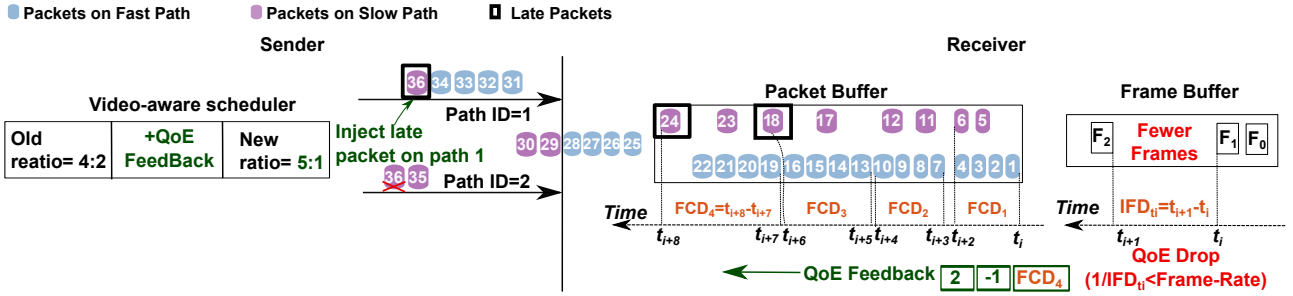


Figure 8: The video QoE feedback mechanism of Converge: when the receiver detects a drop in the frame rate below the expected value, it sends feedback indicating the number of late packets and the path that caused the delay. The scheduler takes this feedback into account and reassigns the late packets to the fast path (Path 1).

Video QoE feedback. For each slow path, we count the number of packets that arrive before and after the media packets on the fast path. If all the packets on the slow path arrive before the frame is constructed, it indicates that we can send more packets along that slow path. If the packets arrive after the frame is constructed, it means that the late arriving packets increase the FCD and indicate that we need to send fewer packets on that slow path. We store the arrival time and FCD of each frame over time. When a new frame enters the frame buffer, we calculate the IFD and record the number of late packets for each path using the reference path. To obtain the expected IFD_{exp}, we invert the frame rate set by the sender. Note that the sender's frame rate is reported using a source description RTCP (SDES) message.

Packets may arrive earlier or later than expected, so we only send feedback when the packet's arrival time starts to deteriorate the QoE. This feedback is sent when $IFD > IFD_{exp}$. The feedback includes three parameters: the number of early/late packets, FCD, and path id indicating the path causing the impact. The number of early/late packets is denoted as α in Equation 2 and can be either positive or negative, indicating whether the number of packets on the path should be increased or decreased, respectively. The FCD parameter helps the sender manage the path.

In Figure 8, FCD_1 and FCD_2 represent the FCD of the first two frames sent to the frame buffer, which are computed using the arrival times (t_i) of their packets. As packets 18 and 24 arrive too late on path 2, FCD_3 and FCD_4 start to increase, leading to an increase in IFD. When IFD exceeds IFD_{exp}, it indicates a QoE drop, and the feedback is sent to reduce the number of packets on path 2 by one packet. The feedback includes path $id = 2$, $\alpha = -1$, and FCD_4. During packet scheduling and distribution, the sender uses this feedback to adjust the number of packets on each path to prevent QoE deterioration. As a result, the new packet ratio becomes $5:1$ instead of $4:2$, and packet 36 is sent on path 1. If the packets arrived early on the given path compared to the reference and a QoE drop occurred, then we would need to increase the number of packets on that path. The positive sign in α indicates an increase in the number of packets.

Sender's reaction to feedback. When the sender receives QoE feedback for a path, it adjusts the number of packets on that path according to Equation 2. If the number of packets becomes zero, the sender disables the path. The path should be re-enabled when its network condition improves for multipath, which can be determined using the received FCD as specified in Equation 3. The smoothed RTT used to reduce the effect of bursts. To measure the RTT of the disabled path, we employ a simple approach of duplicating one of the packets from the fast path as a probing packet.

$$[rtt_{fast} - rtt_i]/2 <= FCD \tag{3}$$

4.3 Reliability via FEC Controller

We have designed a path-specific FEC mechanism that avoids unnecessary overhead for better QoE by using the QoE feedback. The WebRTC system uses a constant FEC rate table based on empirical experiments, but this static table with multipath often results in excessive overhead, which can significantly deteriorate video QoE due to lower media rate and processing delays caused by excessive FEC packets, as we saw in §2.3. Instead, Converge reduces FEC overhead by incorporating video QoE feedback to determine the number of FEC packets for packets destined for each path. We also use NACK requests to further adjust the FEC rate if needed.

The total number of packets (P_i) destined for path i with a loss percentage of l_i is computed based on path statistics and video QoE feedback, as described in Equation 2. Rather than defending packets using a higher FEC rate, if there is a path with a higher loss rate, we reduce the number of packets on that path. We use a video-aware scheduler that considers the importance of packets and the loss rate of each path to determine the FEC rate. This approach achieves a better tradeoff between cost and reliability.

We compute the number of FEC packets (FEC_i) using $(l_i \times P_i \times \beta)$, where β is adjusted dynamically using NACK. If the FEC packets are sufficient to recover the lost packets, we do not receive any NACK requests. If we do receive a NACK request, we need to increase the number of FEC packets by $\beta = [1 + (NACK_i/(P_i - FEC_i))]$.

5 IMPLEMENTATION

WebRTC is a complex codebase that utilizes C++ at its core and a JavaScript application layer. This section provides an overview of the modifications made to over 250 C++ files and the client applications (C++ and JavaScript for browsers) developed to verify the functionality of Converge.

Connections management. In order to support multipath connections and encryption, we made modifications to three WebRTC protocols: Session Description Protocol (SDP) [72], Interactive Connectivity Establishment (ICE) [37], and RTP/RTCP/SRTP [9, 16]. We extended the ICE protocol to obtain possible network connections for multiple paths and modified the SDP to advertise the multipath

capabilities of each peer. We also extended the RTP/RTCP protocols to ensure multipath usage with and without encryption, and the RTP/SRTP protocols to enable multipath usage using the WebRTC keys. To prevent disruptions between Converge's multipath management and WebRTC's existing connection migration (CM), we added a wrapper to monitor the connection status and synchronize it with the WebRTC connection management system.

Transport layer. We enhanced the GCC algorithm to provide specific delay and loss statistics for each path. Additionally, we utilized the original sequence numbers to order packets and construct frames at the receiver.

Media layer. The media layer is closely tied to the transport layer, so we extended it to incorporate the enhanced statistics from the transport layer. We also made modifications to the FEC packet generator and the Media Optimization (MO) module to include the video-aware and path-specific loss recovery mechanisms.

Receiver feedback. We modified the receiver to include a path ID for feedback specific to each path. Additionally, we introduced two additional RTCP messages: one for the sender to indicate the expected frame rate and another for the receiver to send QoE-based feedback to the sender.

Applications and devices. By integrating Converge into Chromium, we developed two client applications (native and web-based) that utilize multiple camera devices and multiple networks. We conducted extensive evaluations of the applications on laptops and mobile devicess (Android). To facilitate the integration into legacy applications, we also provided a user API to explicitly enable or disable the Converge extensions.

Multipath WebRTC variants. As explained in §2.2, we implemented three multipath WebRTC variants for performance comparison with Converge: M-TPUT, SRTT, and M-RTP. In M-TPUT, we incorporated a throughput-based scheduler from Musher [69] into WebRTC. This involved distributing packets in proportion to the throughput of each path. SRTT uses the minRTT scheduler, which is the default scheduler of MPTCP and MPQUIC. For M-RTP, we adhered to the MPRTP specification and utilized a scheduler that sends packets using a loss-based estimated sending rate. In all of these variants, we utilized WebRTC's default FEC module.

6 EVALUATION

This section provides a detailed evaluation of Converge for enhanced video conferencing in real-world network conditions and controlled environments.

Depending on the scenario, we use a single-path WebRTC implementation running on WiFi (WebRTC-W), T-Mobile (WebRTC-T), and Verizon (WebRTC-V) networks. We also compare WebRTC with connection migration (WebRTC-CM), which involves dropping and then re-establishing connections in the event of a connection failure. The CM system switches between the two available networks only when directed by the WebRTC connection migration module. Additionally, we compare Converge with multipath WebRTC variants (SRTT, M-TPUT, and M-RTP), as described in §2.3. All of these variants use two available network paths but differ in their scheduling. They utilizes WebRTC's default FEC management and lack video-aware prioritization.

We measure user QoE using several metrics, including received throughput, Peak Signal-to-Noise Ratio (PSNR) [78], video stall duration, quantization parameter (QP), and E2E latency. PSNR is measured using the technique described in [24]. A higher PSNR and/or lower QP indicates better image quality. To simplify comparison, we normalized QoE metrics by dividing the results from all three cameras by 10 Mbps (the maximum encoding rate of each stream at the sender) for throughput, 24 (the minimum required FPS for good video QoE [67]) for FPS, and 60 (the lowest video quality) for QP. FEC overhead represents the percentage of extra FEC packets generated, while FEC utilization represents the fraction of generated FEC packets actually used for recovering lost packets.

All experiments are conducted using a sample conference call lasting 3 minutes, with a video resolution of 1280x720 and a maximum encoding rate of 10 Mbps per stream, using the same video source [3].

6.1 Converge in the Wild

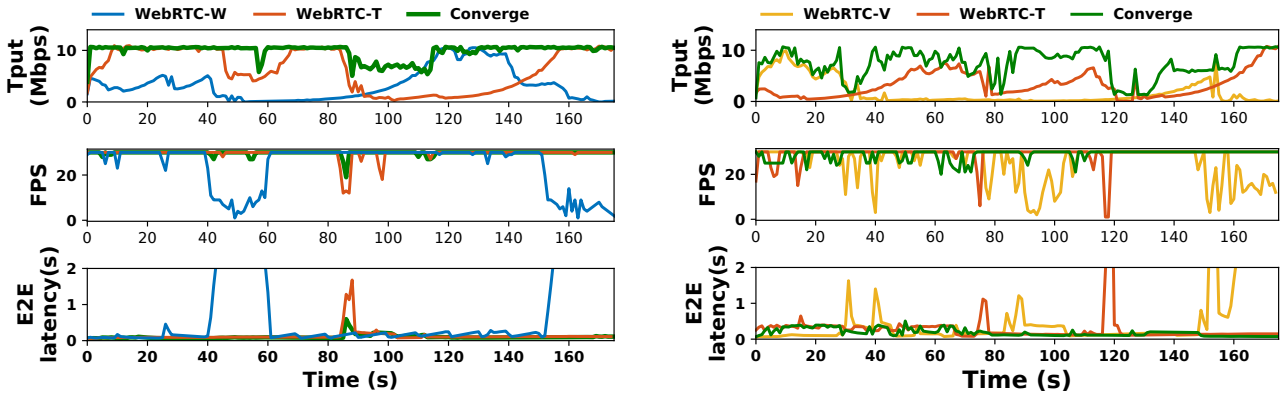
We compared Converge to WebRTC in both walking and driving scenarios. In the walking scenario, Converge utilized two networks, WiFi and T-Mobile, while in the driving scenario, it used Verizon and T-Mobile. In addition, we conducted evaluations of Converge in a stationary scenario, and the detailed results can be found in Appendix A. The cellular networks reported support for 5G along with LTE from their respective vendors.

Figure 9 shows how WebRTC and Converge adapt to time-varying network conditions. Specifically, in the walking scenario (Figure 9(a)), Converge maintains high video throughput, around 19-24 FPS, and low E2E latency, providing a highly interactive video conference experience, except during periods where both T-Mobile and WiFi networks have poor coverage (e.g., 90 sec). In contrast, WebRTC using either network alone exhibits significantly lower throughput and experiences video stalls as the laptops move in and out of network coverage.

The driving scenario (Figure 9(b)), which utilizes two 4G/5G networks, poses greater challenges due to increased network variability as shown in Figure 22 in Appendix D). However, Converge effectively combines the two networks to maintain a high FPS rate and low E2E latency while adjusting the video resolution to match the lower throughput provided by the networks, striking a balance to ensure an interactive conference. In comparison, WebRTC using T-Mobile and Verizon separately encounters periods of zero FPS and increased E2E latency due to coverage issues.

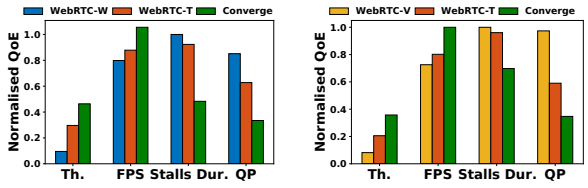
Figure 10 shows normalized QoE metrics. Converge consistently provides video calls with less variations in FPS and E2E latency compared to WebRTC, as evident in Figure 10(b). While walking and driving, Converge achieves average FPS rates of 25 and 24, respectively, representing 41% and 31% improvement over WebRTC. Additionally, Converge reduces video stalls by approximately 33% and 52% compared to WebRTC over WiFi or Verizon, respectively.

Table 3 demonstrates that Converge delivers better QoE for multiple camera streams by aggregating the path. It achieves this by reducing FEC overhead and improving FEC utilization compared to WebRTC in both walking and driving cases. Even with multiple camera streams, Converge maintains low FEC overhead. It



(a) **Walking scenario** (b) **Driving scenario**

Figure 9: Performance of Converge *vs.* WebRTC for walking and driving scenarios. Converge uses both network paths while WebRTC uses a single network.



(a) **Walking** (b) **Driving**

Figure 10: Normalized QoE comparison

End-to-end latency (s)						
	Walking			Driving		
#	WebRTC-W	WebRTC-T	Converge	WebRTC-V	WebRTC-T	Converge
1	0.429 ± 0.153	.234± .060	**.212 ± .089**	.386±.081	0.332±.072	**.304±.085**
2	.505± .212	.355± .101	**.282 ± .093**	.436±.058	.372±.081	**.333±.068**
3	.494± .080	.385± .070	**.321 ± .086**	.429±.042	.368±.040	**.345±.035**
FEC overhead (%)						
1	26.5± 16.7	20.5± 13.2	**9.6 ± 3.1**	19.0±11.1	11±8.1	**8.4±6.3**
2	26.8± 12.3	14.5± 5.1	**8.7 ± 2.6**	22.8±3.6	14.2±6	**7.2±3.8**
3	30.4± 15.6	10.3± 2.2	**8.2 ± 3.6**	15.8±6.2	15.3±6.6	**12.3±5.4**
FEC utilization (%)						
1	11.8± .8	25.0± 9	**64.5 ± 18.5**	13.3±1.5	15.8±2.4	**37±8.2**
2	15.8± 4.3	25.5± 9.1	**60 ± 18.7**	12.8±1.5	20.5±1.1	**33.3±2.9**
3	17.0± 2.5	26.0± 8.7	**37.5 ± 8.6**	15.3±2.3	12±2.2	**25.3±3.3**

Table 3: Converge vs. WebRTC (#: # of camera streams)

improves FEC overhead by 68% and 51%, with 2.8× and 1.3× better utilization while walking and driving, respectively, compared to WebRTC over WiFi and Verizon. By minimizing the IFD and FCD and using less FEC overhead, Converge achieves lower E2E latency compared to WebRTC. Overall, Converge exhibits an improvement of approximately 42% in latency compared to WebRTC over WiFi (WebRTC-W) and approximately 20% compared to WebRTC over Verizon (WebRTC-V). The reduced latency enhances the interactivity of the video conferencing experience.

6.2 Converge in Controlled Environment

In this section, we present a component analysis of Converge and compare it to single-path and multi-path alternatives.

For the component analysis, we created a controlled network environment. Our baseline system utilizes the video-aware scheduler from §4.1, which utilizes GCC to infer the path metrics and splits media packets based on the video structure and individual path sending rate. We then compare this baseline with a system that incorporates QoE feedback as described in §4.2. Finally, we evaluate the performance with and without the FEC module, as explained in §4.3, thus highlighting the advantages of our QoE feedback and FEC methods.

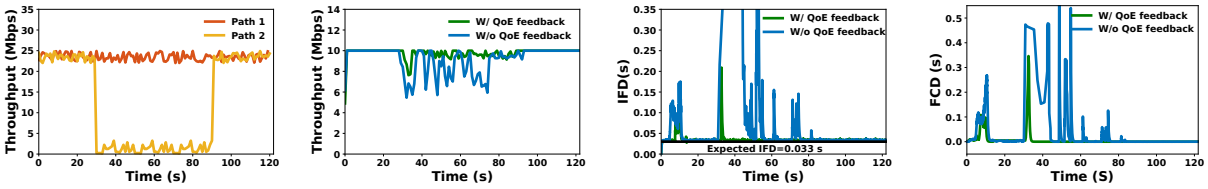
The network environments used in this analysis were collected at various times of the day using iperf3 [34], resulting in 12 scenarios with different bandwidth and loss characteristics. Appendix D provides a description of the bandwidth dynamics for three representative scenarios: stationary, walking, and driving.

The benefit of QoE feedback. To demonstrate the benefit of our QoE feedback, we utilize network dynamics depicted in Figure 11(a). Path 1 maintains a consistent capacity of approximately 25 Mbps, while Path 2 initially has a capacity similar to Path 1 but experiences a drop to around 0.5-2.5 Mbps from 30 to 90 seconds of the conference.

During the interval when the capacity of Path 2 drops suddenly, several events result in lower QoE unless feedback is employed. As observed in Figure 11(b), even though Path 1 has a capacity of 25 Mbps, the overall received rate falls below the maximum encoding rate of 10 Mbps when feedback is not utilized.

Without the Converge feedback, both paths continue to be used. This leads to larger IFDs (Figure 11(c)) and FCDs (Figure 11(d)) due to the loss of necessary packets. In contrast, the QoE metrics in Converge identify that the IFD value exceeds the expected value of 33ms (1/30 of the frame rate) as shown in Figure 11(c), and utilize the measured network metrics (number of late packets, path ID, and gathering delay) to provide feedback to the sender. The sender then begins to reduce the number of late packets scheduled on Path 2. This rapidly provides the necessary packets for frame construction, and the IFD reverts to the target of 33ms. Duplicate packets are still sent on Path 2 to measure network metrics (loss and delay), even though they are not used for video conferencing. This helps to enable the return of the path around the 90th second. With the feedback, the FCD is reduced as late arriving packets are avoided, and the FCD is minimized, as shown in Figure 11(d).

Table 4 demonstrates the improved key QoE metrics achieved by incorporating QoE feedback. The end-to-end metrics indicate



(a) Network path dynamics (b) Video throughput (c) Inter Frame Delay (IFD) (d) Frame Construction Delay (FCD)

Figure 11: The performance of video-aware scheduler with and without video QoE feedback. The inclusion of feedback helps prevent an increase in FCD and IFD, ultimately leading to higher video QoE.

QoE-Parameters	With QoE feedback	Without QoE feedback
Average # of frame drops	560	55 (90% ↑)
Average freeze duration (ms)	1051	302 (71% ↑)
Total # of keyframe requests	14	1 (92 % ↑)

Table 4: Converge with feedback improves video QoE.

	% Improvement									
Loss rate (%)	1	2	3	4	5	6	7	8	9	10
Average # of frame drops	97	93	95	92	97	87	95	98	95	97
Average freeze duration	95	48	11	47	60	50	50	52	50	69
Total # of keyframe requests	66	66	50	50	66	66	50	80	50	66

Table 5: % QoE improvement between Converge with path-specific FEC module and WebRTC's table-based FEC module.

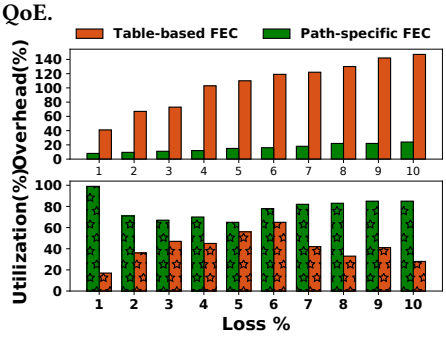


Figure 12: FEC overhead and utilization between Converge's path-specific FEC and WebRTC's table-based FEC.

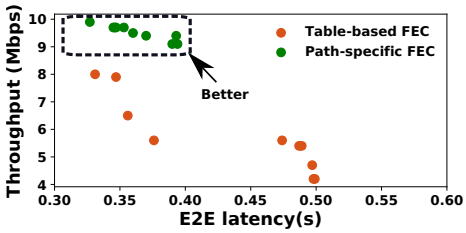


Figure 13: QoE trade-Off analysis of Converge with path-specific FEC module *vs.* WebRTC's table-based FEC module

that Path 2 is unreliable, resulting in approximately 10 times fewer frames being dropped with Converge compared to not using end-to-end feedback. As a result, only a single keyframe retransmission is requested with Converge, in contrast to 14 keyframes without the end-to-end feedback, leading to significantly reduced video freeze duration.

QoE trade-off analysis of FEC. The benefit of Converge extends beyond quickly reacting to network fluctuations; it also involves adapting the FEC mechanism used to repair lost packets. In this analysis, we compare our path-specific loss-based FEC controller with the default WebRTC mechanism. For comparison, we create a controlled environment with two paths of 15 Mbps capacities, 100ms propagation delay, and varying loss levels from 0-10%.

WebRTC employs a simple table-driven FEC rate, which is overly aggressive, as depicted in Figure 12. For instance, at a low loss rate of 1%, the table-driven approach of WebRTC would send an additional 40 FEC packets for every 100 media packets, resulting in an overhead of approximately 40%.

Figure 12 also illustrates the utilization of the FEC packets. With a 1% loss rate, WebRTC sends an additional 40% packets, but less than 20% of them are actually utilized. In comparison Converge sends approximately 5% additional FEC packets, and almost all of them are used.

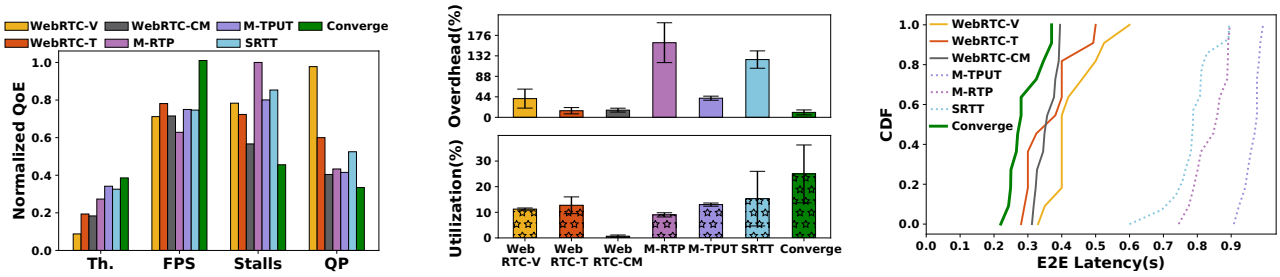
By dynamically adjusting the number of FEC packets based on loss, utilization, and video QoE feedback, as described in §4.3, Converge provides an adequate amount of FEC to meet frame-rate and latency goals while reducing bandwidth and processing requirements. Without this adjustment, excessive FEC overhead increases E2E delay and reduces media throughput. Figure 13 depicts the trade-off between throughput and end-to-end video delay. Converge operates in the upper-right corner, offering high media throughput while simultaneously reducing end-to-end delay. In comparison, WebRTC's table-driven approach adds excessive FEC overhead, resulting in both reduced media throughput and increased end-to-end latency.

Table 5 highlights the benefits of Converge's FEC controller on QoE metrics. For example, at a 10% packet loss rate (right-most column), the Converge method reduces frame drops by 97% compared to the WebRTC controller, leading to a 67% reduction in freeze duration and 66% fewer keyframe requests.

Comparison with existing solutions. We compared the performance of Converge in both the "walking" and "driving" scenarios, but we present only the results for the "driving" scenario as it more prominently showcases the advantages of the mechanisms employed in Converge. We provide the average values for each metric across every camera.

Figure 14(a) illustrates that Converge achieves an average delivered throughput of 38% of the encoding rate at the sender, which is significantly higher than all other single and multipath solutions. Converge utilizes this higher throughput to achieve a high FPS and reduce video stalls, resulting in improved interactivity and better video quality due to QP.

From Figure 14(b), we observe the FEC overhead of Converge's path-specific control is smaller with more prominent utilization; it sends less FEC data and uses more of what it sends. Its FEC overhead



(a) **Normalized QoE** (b) **FEC utility and overhead** (c) **E2E delay**

Figure 14: Performance comparison between Converge and existing solutions. WebRTC uses a single network path while Converge, M-RTP, M-TPUT, SRTT use multiple paths. E2E shows the latency of all video frames.

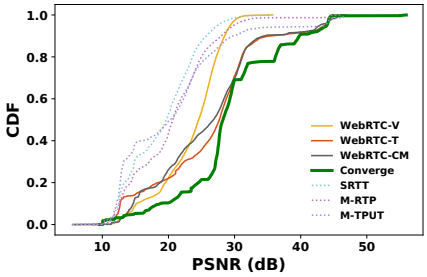


Figure 15: PSNR of Converge vs. existing solutions

is 10% during driving due to increased mobility with higher loss. It helps recover ~25% of packets from the received FEC packets. On the other hand, all multipath variants have an overhead of a minimum 41% with a maximum utilization of 15%.

In video conferencing, the E2E latency is an essential factor that defines the interactivity of the video – higher E2E latency leads to poorer interactions. From Figure 14(c), we see that Converge achieves the least E2E latency compared to the alternate solutions and that the other multipath implementations have *qualitatively higher* E2E latency. This is surprising because, during the driving scenario (Figure 14(c)), the Verizon path has video conferencing latency between 0.3 to .6 seconds. But, when that path is combined with the T-mobile path, the aggregate E2E latency has increased to ~1 second. The reason is that all the multipath variants cause the FCD to increase significantly and fail to recognize its impact on the frame rate at the receiver. By comparison, Converge uses path-specific feedback to reduce these delays and hence achieves better E2E latency that provides better interaction.

We measured the video QP (lower is better) and PSNR (higher is better) to quantify the improved image quality. The QP is reduced for Converge as shown in Figure 14(a). Furthermore, we measured the PSNR using a single camera stream in the Chromium browser. As shown in Figure 15, Converge outperforms WebRTC and all other multipath implementations significantly in terms of PSNR.

7 RELATED WORK

Multipath Protocols. MPTCP, MPQUIC, and MPRTP are protocols that use multiple network interfaces for data transmission and have been widely studied for their use in video streaming. MPTCP has been explored for its use in video streaming. In particular, Han et al. developed a multipath framework called MP-DASH [28] that is aware of network interfaces.

On the other hand, MPQUIC [13, 83] extends the QUIC protocol for multipath. It uses multiple streams of HTTP over multiple independent paths and has been explored for use in video streaming. MRTP [53] is a multipath extension to the real-time transport protocol (RTP) used for media transmission. Singh et al. extended MRTP to create a new standard called MPRTP [71], which uses a constant bit rate to transmit media packets on all paths.

In contrast to MPTCP, MPQUIC, and MPRTP, Converge is the first multipath *video conferencing* system to go beyond just packet scheduling. It is built with a focus on time sensitivity and small, fixed-size buffers and provides a complete conferencing system.

Video Enhancement. Converge is inspired by cross-layer video enhancement techniques [20, 33, 40, 43, 45, 52, 77, 84, 90]. However, it is tailored for multipath systems rather than single network path systems and uses the more fine-grained real-time video and its construct to optimize video QoE that suits conferencing. Additionally, Converge uses FEC techniques that are closely related to loss-based techniques [31, 32, 44], but for multipath-based conferencing.

8 CONCLUSION

We identified growing demands for video conferencing and how existing solutions fail to provide uninterrupted video calls with superior QoE, even with advances in next-generation networks. Therefore, we built Converge, a practical video conferencing system that seamlessly supports multipath and multiple cameras and is compatible with any device and existing WebRTC protocol. Converge solves three significant challenges using a video-aware scheduler, video QoE feedback, and a path-specific loss-based FEC module for enhanced video conferencing. We tested Converge in both emulated and real-world networks and demonstrated how it provides video calls with minimal interruptions, regardless of the number of camera streams, and provides better video QoE by effectively utilizing multiple network connections.

ACKNOWLEDGMENT

We express our gratitude to our shepherd, Anna Brunstrom, and the anonymous reviewers for their valuable comments and feedback. Their insightful input has significantly enhanced the quality of this paper. This research was partly supported by Cisco Systems (Grant 1368170) and the MSIT (Ministry of Science and ICT) of Korea under the ITRC support program (IITP-2021-0-02048) of the IITP.

REFERENCES

[1] Shivang Aggarwal, Moinak Ghoshal, Piyali Banerjee, Dimitrios Koutsonikolas, and Joerg Widmer. 2021. 802.11ad in Smartphones: Energy Efficiency, Spatial Reuse, and Impact on Applications. In *IEEE INFOCOM 2021 - IEEE Conference on Computer Communications*. 1–10. https://doi.org/10.1109/INFOCOM42981.2021.9488763

[2] M. Alfano and R. Sigle. 1996. Controlling QoS in a collaborative multimedia environment. In *Proceedings of 5th IEEE International Symposium on High Performance Distributed Computing*. IEEE, 340–347. https://doi.org/10.1109/HPDC.1996.546204

[3] Blender Foundation:www.bigbuckbunny.org. 2008. Big Buck Bunny. (2008).

[4] J.-C. Bolot and T. Turletti. 1996. Adaptive error control for packet video in the Internet. In *Proceedings of 3rd IEEE International Conference on Image Processing*, Vol. 1. IEEE, 25–28 vol.1. https://doi.org/10.1109/ICIP.1996.559424

[5] Andrew Campbell, Alexandros Eleftheriadis, and Cristina Aurrecoechea. 1996. *End-to-End QoS Management for Adaptive Video Flows*. Springer US, Boston, MA, 105–115. https://doi.org/10.1007/978-1-4613-0403-6_14

[6] Gaetano Carlucci, Luca De Cicco, Stefan Holmer, and Saverio Mascolo. 2016. Analysis and Design of the Google Congestion Control for Web Real-Time Communication (WebRTC). In *Proceedings of the 7th International Conference on Multimedia Systems (MMSys '16)*. Association for Computing Machinery, New York, NY, USA, Article 13, 12 pages. https://doi.org/10.1145/2910017.2910605

[7] Gaetano Carlucci, Luca De Cicco, and Saverio Mascolo. 2018. Controlling Queuing Delays for Real-Time Communication: The Interplay of E2E and AQM Algorithms. *SIGCOMM Comput. Commun. Rev.* 46, 3, Article 1 (jul 2018), 7 pages. https://doi.org/10.1145/3243157.3243158

[8] Lucas Chaufournier, Ahmed Ali-Eldin, Prateek Sharma, Prashant Shenoy, and Don Towsley. 2019. Performance Evaluation of Multi-Path TCP for Data Center and Cloud Workloads. In *Proceedings of the 2019 ACM/SPEC International Conference on Performance Engineering (ICPE '19)*. Association for Computing Machinery, New York, NY, USA, 13–24. https://doi.org/10.1145/3297663.3310295

[9] Colin Perkins, Magnus Westerlund and Joerg Ott. 2016. Web Real-Time Communication (WebRTC): Media Transport and Use of RTP draft-ietf-rtcweb-rtp-usage-26 . https://datatracker.ietf.org/doc/html/draft-ietf-rtcweb-rtp-usage-26. (2016).

[10] Quentin Coninck, Matthieu Baerts, Benjamin Hesmans, and Olivier Bonaventure. 2016. A First Analysis of Multipath TCP on Smartphones. 57–69. https://doi.org/10.1007/978-3-319-30505-9_5

[11] Quentin De Coninck, Matthieu Baerts, Benjamin Hesmans, and Olivier Bonaventure. 2016. Observing Real Smartphone Applications over Multipath TCP. *IEEE Communications Magazine, Network Testing Series* 54, 3 (March 2016).

[12] Quentin De Coninck. 2022. The Packet Number Space Debate in Multipath QUIC. *SIGCOMM Comput. Commun. Rev.* 52, 3 (sep 2022), 2–9. https://doi.org/10.1145/3561954.3561956

[13] Quentin De Coninck and Olivier Bonaventure. 2017. Multipath QUIC: Design and Evaluation. In *Proceedings of the 13th International Conference on Emerging Networking EXperiments and Technologies (CoNEXT '17)*. Association for Computing Machinery, New York, NY, USA, 160–166. https://doi.org/10.1145/3143361.3143370

[14] Florin Dobrian, Vyas Sekar, Asad Awan, Ion Stoica, Dilip Joseph, Aditya Ganjam, Jibin Zhan, and Hui Zhang. 2011. Understanding the Impact of Video Quality on User Engagement. In *Proceedings of the ACM SIGCOMM 2011 Conference (SIGCOMM '11)*. Association for Computing Machinery, New York, NY, USA, 362–373. https://doi.org/10.1145/2018436.2018478

[15] Anis Elgabli, Ke Liu, and Vaneet Aggarwal. 2020. Optimized Preference-Aware Multi-Path Video Streaming with Scalable Video Coding. *IEEE Transactions on Mobile Computing* 19, 1 (2020), 159–172. https://doi.org/10.1109/TMC.2018.2889039

[16] Eric Rescorla. 2019. Security Considerations for WebRTC draft-ietf-rtcweb-security-12 . https://datatracker.ietf.org/doc/html/draft-ietf-rtcweb-security. (2019).

[17] Simone Ferlin, Özgü Alay, Olivier Mehani, and Roksana Boreli. 2016. BLEST: Blocking estimation-based MPTCP scheduler for heterogeneous networks. In *2016 IFIP Networking Conference (IFIP Networking) and Workshops*. 431–439. https://doi.org/10.1109/IFIPNetworking.2016.7497206

[18] Sally Floyd, Van Jacobson, Steve McCanne, Ching-Gung Liu, and Lixia Zhang. 1995. A Reliable Multicast Framework for Light-Weight Sessions and Application Level Framing. In *Proceedings of the Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication (SIGCOMM '95)*. Association for Computing Machinery, New York, NY, USA, 342–356. https://doi.org/10.1145/217382.217470

[19] Sally Floyd, Van Jacobson, Steve McCanne, Ching-Gung Liu, and Lixia Zhang. 1995. A Reliable Multicast Framework for Light-Weight Sessions and Application Level Framing. *SIGCOMM Comput. Commun. Rev.* 25, 4 (oct 1995), 342–356. https://doi.org/10.1145/217391.217470

[20] Sadjad Fouladi, John Emmons, Emre Orbay, Catherine Wu, Riad S. Wahby, and Keith Winstein. 2018. Salsify: Low-Latency Network Video through Tighter Integration between a Video Codec and a Transport Protocol. In *Proceedings of the 15th USENIX Conference on Networked Systems Design and Implementation (NSDI'18)*. USENIX Association, USA, 267–282.

[21] Sadjad Fouladi, Riad S. Wahby, Brennan Shacklett, Karthikeyan Vasuki Balasubramaniam, William Zeng, Rahul Bhalerao, Anirudh Sivaraman, George Porter, and Keith Winstein. 2017. Encoding, Fast and Slow: Low-Latency Video Processing Using Thousands of Tiny Threads. In *14th USENIX Symposium on Networked Systems Design and Implementation (NSDI 17)*. USENIX Association, Boston, MA, 363–376. https://www.usenix.org/conference/nsdi17/technical-sessions/presentation/fouladi

[22] Ron Frederick. 1994. Experiences With Real-Time Software Video Compression. (1994).

[23] freeappsforme. 2020. 7 Best Dual Camera Apps for Android. (2020).

[24] Boni García, Francisco Gortázar, Micael Gallego, and Andrew Hines. 2020. Assessment of QoE for Video and Audio in WebRTC Applications Using Full-Reference Models. *Electronics* 9, 3 (2020). https://doi.org/10.3390/electronics9030462

[25] Google Apps. 2020. DelfieArt: Double-View Selfie Live Overlay Camera. (2020).

[26] Google-WebRTC Open Source. 2021. Video coding in WebRTC. https://chromium.googlesource.com/external/webrtc/+/master/modules/video_coding/g3doc/index.md. (2021).

[27] Boris Grozev, Emil Ivov, Arnaud Budkiewicz, Ludovic Roux, and Alexandre Gouaillard. 2017. PERC double media encryption for WebRTC 1.0 sender simulcast. In *2017 Principles, Systems and Applications of IP Telecommunications (IPTComm)*. 1–5. https://doi.org/10.1109/IPTCOMM.2017.8169752

[28] Bo Han, Feng Qian, Lusheng Ji, and Vijay Gopalakrishnan. 2016. MP-DASH: Adaptive Video Streaming Over Preference-Aware Multipath. In *Proceedings of the 12th International on Conference on Emerging Networking EXperiments and Technologies (CoNEXT '16)*. Association for Computing Machinery, New York, NY, USA, 129–143. https://doi.org/10.1145/2999572.2999606

[29] Ahmad Hassan, Arvind Narayanan, Anlan Zhang, Wei Ye, Ruiyang Zhu, Shuowei Jin, Jason Carpenter, Z. Morley Mao, Feng Qian, and Zhi-Li Zhang. 2022. Vivisecting Mobility Management in 5G Cellular Networks. In *Proceedings of the ACM SIGCOMM 2022 Conference (SIGCOMM '22)*. Association for Computing Machinery, New York, NY, USA, 86–100. https://doi.org/10.1145/3544216.3544217

[30] Jia He, Mostafa Ammar, and Ellen Zegura. 2023. A Measurement-Derived Functional Model For Interaction Between Congestion Control QoE Video Conferencing. Springer-Verlag, Berlin, Heidelberg, 129–159. https://doi.org/10.1007/978-3-031-28486-1_7

[31] Stefan Holmer, Mikhal Shemer, and Marco Paniconi. 2013. Handling packet loss in WebRTC. In *2013 IEEE International Conference on Image Processing*. 1860–1864. https://doi.org/10.1109/ICIP.2013.6738383

[32] Han Hu, Sheng Cheng, Xinggong Zhang, and Zongming Guo. 2021. LightFEC: Network Adaptive FEC with a Lightweight Deep-Learning Approach. In *Proceedings of the 29th ACM International Conference on Multimedia (MM '21)*. Association for Computing Machinery, New York, NY, USA, 3592–3600. https://doi.org/10.1145/3474085.3475528

[33] Te-Yuan Huang, Ramesh Johari, Nick McKeown, Matthew Trunnell, and Mark Watson. 2014. A Buffer-Based Approach to Rate Adaptation: Evidence from a Large Video Streaming Service. *SIGCOMM Comput. Commun. Rev.* 44, 4 (aug 2014), 187–198. https://doi.org/10.1145/2740070.2626296

[34] Iperf.fr. 2016. iPerf - The ultimate speed test tool for TCP, UDP and SCTP. (June 2016). https://iperf.fr/iperf-download.php.

[35] Bart Jansen, Timothy Goodwin, Varun Gupta, Fernando Kuipers, and Gil Zussman. 2018. Performance Evaluation of WebRTC-Based Video Conferencing. *SIGMETRICS Perform. Eval. Rev.* 45, 3 (mar 2018), 56–68. https://doi.org/10.1145/3199524.3199534

[36] Junchen Jiang, Vyas Sekar, Henry Milner, Davis Shepherd, Ion Stoica, and Hui Zhang. 2016. CFA: A Practical Prediction System for Video QoE Optimization. In *Proceedings of the 13th Usenix Conference on Networked Systems Design and Implementation (NSDI'16)*. USENIX Association, USA, 137–150.

[37] Jonathan Rosenberg. 2010. Interactive Connectivity Establishment (ICE): A Protocol for Network Address Translator (NAT) Traversal for Offer/Answer Protocols . https://datatracker.ietf.org/doc/html/rfc5245. (2010).

[38] Just2US. 2020. DualGram. (2020).

[39] Alireza Keshavarz-Haddad, Ehsan Aryafar, Michael Wang, and Mung Chiang. 2017. HetNets Selection by Clients: Convergence, Efficiency, and Practicality. *IEEE/ACM Transactions on Networking* 25, 1 (2017), 406–419. https://doi.org/10.1109/TNET.2016.2587622

[40] Jaehong Kim, Youngmok Jung, Hyunho Yeo, Juncheol Ye, and Dongsu Han. 2020. 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 (SIGCOMM '20)*. Association for Computing Machinery, New York, NY, USA, 107–125. https://doi.org/10.1145/3387514.3405856

[41] Hitoshi Aida Kultida Rojviboonchai. 2004. An Evaluation of Multi-path Transmission Control Protocol (M/TCP) with Robust Acknowledgement Schemes. IEICE TRANSACTIONS on Communications Vol.E87-B No.9 pp.2699-2707.

[42] Lars Rehm. 2018. Multi-camera smartphone segment growing at record pace. (2018).

[43] HyunJong Lee, Jason Flinn, and Basavaraj Tonshal. 2018. RAVEN: Improving Interactive Latency for the Connected Car. In *Proceedings of the 24th Annual International Conference on Mobile Computing and Networking (MobiCom '18)*. Association for Computing Machinery, New York, NY, USA, 557–572. https://doi.org/10.1145/3241539.3241571

[44] Insoo Lee, Seyeon Kim, Sandesh Sathyanarayana, Kyungmin Bin, Song Chong, Kyunghan Lee, Dirk Grunwald, and Sangtae Ha. 2022. R-FEC: RL-Based FEC Adjustment for Better QoE in WebRTC. In *Proceedings of the 30th ACM International Conference on Multimedia (MM '22)*. Association for Computing Machinery, New York, NY, USA, 2948–2956. https://doi.org/10.1145/3503161.3548370

[45] Jinyang Li, Zhenyu Li, Ri Lu, Kai Xiao, Songlin Li, Jufeng Chen, Jingyu Yang, Chunli Zong, Aiyun Chen, Qinghua Wu, Chen Sun, Gareth Tyson, and Hongqiang Harry Liu. 2022. LiveNet: A Low-Latency Video Transport Network for Large-Scale Live Streaming. In *Proceedings of the ACM SIGCOMM 2022 Conference (SIGCOMM '22)*. Association for Computing Machinery, New York, NY, USA, 812–825. https://doi.org/10.1145/3544216.3544236

[46] Li Li, Ke Xu, Tong Li, Kai Zheng, Chunyi Peng, Dan Wang, Xiangxiang Wang, Meng Shen, and Rashid Mijumbi. 2018. A Measurement Study on Multi-Path TCP with Multiple Cellular Carriers on High Speed Rails. In *Proceedings of the 2018 Conference of the ACM Special Interest Group on Data Communication (SIGCOMM '18)*. Association for Computing Machinery, New York, NY, USA, 161–175. https://doi.org/10.1145/3230543.3230556

[47] Hyoyoung Lim, Jinsung Lee, Jongyun Lee, Sandesh Dhawaskar Sathyanarayana, Junseon Kim, Anh Nguyen, Kwang Taik Kim, Youngbin Im, Mung Chiang, Dirk Grunwald, Kyunghan Lee, and Sangtae Ha. 2023. An Empirical Study of 5G: Effect of Edge on Transport Protocol and Application Performance. *IEEE Transactions on Mobile Computing* (2023), 1–16. https://doi.org/10.1109/TMC.2023.3274708

[48] Yeon-sup Lim, Erich M. Nahum, Don Towsley, and Richard J. Gibbens. 2017. ECF: An MPTCP Path Scheduler to Manage Heterogeneous Paths. In *Proceedings of the 13th International Conference on Emerging Networking EXperiments and Technologies (CoNEXT '17)*. Association for Computing Machinery, New York, NY, USA, 147–159. https://doi.org/10.1145/3143361.3143376

[49] A. Limongiello, R. Melen, M. Rocuzzo, V. Trecordi, and J. Wojtowicz. 1997. An experimental open architecture to support multimedia services based on CORBA, Java and WWW Technologies. In *Intelligence in Services and Networks: Technology for Cooperative Competition*, Al Mullery, Michel Besson, Mario Campolargo, Roberta Gobbi, and Rick Reed (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 69–75.

[50] Ruofeng Liu and Nakjung Choi. 2023. A First Look at Wi-Fi 6 in Action: Throughput, Latency, Energy Efficiency, and Security. *Proc. ACM Meas. Anal. Comput. Syst.* 7, 1, Article 25 (mar 2023), 25 pages. https://doi.org/10.1145/3579451

[51] Kyle MacMillan, Tarun Mangla, James Saxon, and Nick Feamster. 2021. Measuring the Performance and Network Utilization of Popular Video Conferencing Applications. In *Proceedings of the 21st ACM Internet Measurement Conference (IMC '21)*. Association for Computing Machinery, New York, NY, USA, 229–244. https://doi.org/10.1145/3487552.3487842

[52] Hongzi Mao, Ravi Netravali, and Mohammad Alizadeh. 2017. Neural Adaptive Video Streaming with Pensieve. In *Proceedings of the Conference of the ACM Special Interest Group on Data Communication (SIGCOMM '17)*. Association for Computing Machinery, New York, NY, USA, 197–210. https://doi.org/10.1145/3098822.3098843

[53] Shiwen Mao, D. Bushmitch, S. Narayanan, and S.S. Panwar. 2006. MRTP: a multiflow real-time transport protocol for ad hoc networks. *IEEE Transactions on Multimedia* 8, 2 (2006), 356–369. https://doi.org/10.1109/TMM.2005.864347

[54] François Michel, Alejandro Cohen, Derya Malak, Quentin De Coninck, Muriel Médard, and Olivier Bonaventure. 2023. FlEC: Enhancing QUIC With Application-Tailored Reliability Mechanisms. *IEEE/ACM Transactions on Networking* 31, 2 (2023), 606–619. https://doi.org/10.1109/TNET.2022.3195611

[55] Marcin Nagy, Varun Singh, Jörg Ott, and Lars Eggert. 2014. Congestion Control Using FEC for Conversational Multimedia Communication. In *Proceedings of the 5th ACM Multimedia Systems Conference (MMSys '14)*. Association for Computing Machinery, New York, NY, USA, 191–202. https://doi.org/10.1145/2557642.2557649

[56] Arvind Narayanan, Eman Ramadan, Jason Carpenter, Qingxu Liu, Yu Liu, Feng Qian, and Zhi-Li Zhang. 2020. A First Look at Commercial 5G Performance on Smartphones. In *Proceedings of The Web Conference 2020 (WWW '20)*. Association for Computing Machinery, New York, NY, USA, 894–905. https://doi.org/10.1145/3366423.3380169

[57] Arvind Narayanan, Xumiao Zhang, Ruiyang Zhu, Ahmad Hassan, Shuowei Jin, Xiao Zhu, Xiaoxuan Zhang, Denis Rybkin, Zhengxuan Yang, Zhuoqing Morley Mao, Feng Qian, and Zhi-Li Zhang. 2021. A Variegated Look at 5G in the Wild: Performance, Power, and QoE Implications. In *Proceedings of the 2021 ACM SIGCOMM 2021 Conference (SIGCOMM '21)*. Association for Computing Machinery, New York, NY, USA, 610–625. https://doi.org/10.1145/3452296.3472923

[58] Vikram Nathan, Vibhaalakshmi Sivaraman, Ravichandra Addanki, Mehrdad Khani, Prateesh Goyal, and Mohammad Alizadeh. 2019. End-to-End Transport for Video QoE Fairness. In *Proceedings of the ACM Special Interest Group on Data Communication (SIGCOMM '19)*. Association for Computing Machinery, New York, NY, USA, 408–423. https://doi.org/10.1145/3341302.3342077

[59] Satya Ganesh Nutan Dev C, Goutham Ponnamreddy, and Debabrata Das. 2021. A method to boost throughput in 5G and 4G mobiles using MultiSIM subscription on the same network. In *2021 IEEE International Conference on Electronics, Computing and Communication Technologies (CONECCT)*. 01–06. https://doi.org/10.1109/CONECCT52877.2021.9622612

[60] Fakher Oueslati and Jean-Charles Grégoire. 2015. An adaptation mechanism for robust OTT video transmission. In *2015 IEEE 16th International Symposium on A World of Wireless, Mobile and Multimedia Networks (WoWMoM)*. 1–6. https://doi.org/10.1109/WoWMoM.2015.7158218

[61] Christoph Paasch, Gregory Detal, Fabien Duchene, Costin Raiciu, and Olivier Bonaventure. 2012. Exploring Mobile/WiFi Handover with Multipath TCP. In *ACM SIGCOMM workshop on Cellular Networks (Cellnet'12)*.

[62] Christoph Paasch, Simone Ferlin, Ozgu Alay, and Olivier Bonaventure. 2014. Experimental Evaluation of Multipath TCP Schedulers. In *Proceedings of the 2014 ACM SIGCOMM Workshop on Capacity Sharing Workshop (CSWS '14)*. Association for Computing Machinery, New York, NY, USA, 27–32. https://doi.org/10.1145/2630088.2631977

[63] Sohee Kim Park, Arani Bhattacharya, Mallesham Dasari, and Samir R. Das. 2018. Understanding User Perceived Video Quality Using Multipath TCP Over Wireless Network. In *2018 IEEE 39th Sarnoff Symposium*. 1–6. https://doi.org/10.1109/SARNOF.2018.8720402

[64] P.Eardley. 2013. http://blog.multipath-tcp.org/blog/html/2018/12/15/apple_and_multipath_tcp.html. (2013).

[65] Costin Raiciu, Christoph Paasch, Sebastien Barre, Alan Ford, Michio Honda, Fabien Duchene, Olivier Bonaventure, and Mark Handley. 2012. How Hard Can It Be? Designing and Implementing a Deployable Multipath TCP. In *9th USENIX Symposium on Networked Systems Design and Implementation (NSDI 12)*. USENIX Association, San Jose, CA, 399–412. https://www.usenix.org/conference/nsdi12/technical-sessions/presentation/raiciu

[66] Devdeep Ray, Jack Kosaian, K. V. Rashmi, and Srinivasan Seshan. 2019. Vantage: Optimizing Video Upload for Time-Shifted Viewing of Social Live Streams. In *Proceedings of the ACM Special Interest Group on Data Communication (SIGCOMM '19)*. Association for Computing Machinery, New York, NY, USA, 380–393. https://doi.org/10.1145/3341302.3342064

[67] Riverside. 2022. The Ultimate Frame Rate Guide for Beginners. https://riverside.fm/blog/frame-rate-guide. (2022).

[68] Bryan Robinson. [n. d.]. Remote Work Is Here To Stay And Will Increase Into 2023, Experts Say. https://www.forbes.com/sites/bryanrobinson/2022/02/01/remote-work-is-here-to-stay-and-will-increase-into-2023-experts-say/. ([n. d.]).

[69] Swetank Kumar Saha, Shivang Aggarwal, Rohan Pathak, Dimitrios Koutsonikolas, and Joerg Widmer. 2019. MuSher: An Agile Multipath-TCP Scheduler for Dual-Band 802.11ad/Ac Wireless LANs. In *The 25th Annual International Conference on Mobile Computing and Networking (MobiCom '19)*. Association for Computing Machinery, New York, NY, USA, Article 34, 16 pages. https://doi.org/10.1145/3300061.3345435

[70] Sandesh Dhawaskar Sathyanarayana, Jinsung Lee, Jihoon Lee, Dirk Grunwald, and Sangtae Ha. 2021. Exploiting Client Inference in Multipath TCP Over Multiple Cellular Networks. *IEEE Communications Magazine* 59, 4, 58–64. https://doi.org/10.1109/MCOM.001.2000911

[71] Varun Singh, Saba Ahsan, and Ott. 2013. MPRTP: Multipath Considerations for Real-Time Media. In *Proceedings of the 4th ACM Multimedia Systems Conference (MMSys '13)*. Association for Computing Machinery, New York, NY, USA, 190–201. https://doi.org/10.1145/2483977.2484002

[72] Suhas Nandakumar, Cullen Jennings. 2021. Annotated Example SDP for WebRTC); SDP for WebRTC (Release 14). https://www.ietf.org/archive/id/draft-ietf-rtcweb-sdp-14.txt. (2021).

[73] Y.S. Sun, Chin-Fu Ku, Yu-Chun Pan, Chia-Hui Wang, and Jan-Ming Ho. 1996. Performance analysis of application-level traffic shaping in a real-time multimedia conferencing system on Ethernets. In *Proceedings of LCN - 21st Annual Conference on Local Computer Networks*. IEEE, 433–442. https://doi.org/10.1109/LCN.1996.558172

[74] Xuanxuan Tian, Shuo Jia, Ping Dong, Tao Zheng, and Xiaoyun Yan. 2018. An Adaptive Bitrate Control Algorithm for Real-Time Streaming Media Transmission in High-Speed Railway Networks. In *2018 10th International Conference on Communication Software and Networks (ICCSN)*. 328–333. https://doi.org/10.1109/ICCSN.2018.8488286

[75] Varun Singh,Rachel Huang,Roni Even,Dan Romascanu. 2018. RFC8451: Considerations for Selecting RTP Control Protocol (RTCP) Extended Report (XR) Metrics for the WebRTC Statistics API. (2018). https://datatracker.ietf.org/doc/html/rfc8451.

[76] Bing Wang, Wei Wei, Zheng Guo, and Don Towsley. 2009. Multipath Live Streaming via TCP: Scheme, Performance and Benefits. *ACM Trans. Multimedia Comput. Commun. Appl.* 5, 3, Article 25 (aug 2009), 23 pages. https://doi.org/10.1145/1556134.1556142

[77] Bo Wang, Mingwei Xu, Fengyuan Ren, Chao Zhou, and Jianping Wu. 2022. Cratus: A Lightweight and Robust Approach for Mobile Live Streaming. *IEEE Transactions on Mobile Computing* 21, 8 (2022), 2761–2775. https://doi.org/10.1109/TMC.2020.

3048826
[78] Wikipedia. 2019. Peak signal-to-noise ratio. https://en.wikipedia.org/wiki/Peak_signal-to-noise_ratio. (2019).
[79] Damon Wischik, Costin Raiciu, Adam Greenhalgh, and Mark Handley. 2011. Design, Implementation and Evaluation of Congestion Control for Multipath TCP. In *8th USENIX Symposium on Networked Systems Design and Implementation (NSDI 11)*. USENIX Association, Boston, MA. https://www.usenix.org/conference/nsdi11/design-implementation-and-evaluation-congestion-control-multipath-tcp
[80] Damon Wischik, Costin Raiciu, Adam Greenhalgh, and Mark Handley. 2011. Design, Implementation and Evaluation of Congestion Control for Multipath TCP. In *Proceedings of the 8th USENIX Conference on Networked Systems Design and Implementation (NSDI'11)*. USENIX Association, USA, 99–112.
[81] Yitao Xing, Kaiping Xue, Yuan Zhang, Jiangping Han, Jian Li, Jianqing Liu, and Ruidong Li. 2021. A Low-Latency MPTCP Scheduler for Live Video Streaming in Mobile Networks. *IEEE Transactions on Wireless Communications* 20, 11 (2021), 7230–7242. https://doi.org/10.1109/TWC.2021.3081498
[82] Dongzhu Xu, Anfu Zhou, Xinyu Zhang, Guixian Wang, Xi Liu, Congkai An, Yiming Shi, Liang Liu, and Huadong Ma. 2020. Understanding Operational 5G: A First Measurement Study on Its Coverage, Performance and Energy Consumption. 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 (SIGCOMM '20)*. Association for Computing Machinery, New York, NY, USA, 479–494. https://doi.org/10.1145/3387514.3405882
[83] Yanmei Liu, Yunfei Ma, Quentin De Coninck, Olivier Bonaventure, Christian Huitema, Mirja Kühlewind . 2023. Multipath Extension for QUIC. (2023). https://datatracker.ietf.org/doc/draft-ietf-quic-multipath/.
[84] Hyunho Yeo, Hwijoon Lim, Jaehong Kim, Youngmok Jung, Juncheol Ye, and Dongsu Han. 2022. NeuroScaler: Neural Video Enhancement at Scale. In *Proceedings of the ACM SIGCOMM 2022 Conference (SIGCOMM '22)*. Association for Computing Machinery, New York, NY, USA, 795–811. https://doi.org/10.1145/3544216.3544218
[85] Xinjie Yuan, Mingzhou Wu, Zhi Wang, Yifei Zhu, Ming Ma, Junjian Guo, Zhi-Li Zhang, and Wenwu Zhu. 2022. Understanding 5G Performance for Real-World Services: A Content Provider's Perspective. In *Proceedings of the ACM SIGCOMM 2022 Conference (SIGCOMM '22)*. Association for Computing Machinery, New York, NY, USA, 101–113. https://doi.org/10.1145/3544216.3544219
[86] Yunze Zeng, Parth H. Pathak, and Prasant Mohapatra. 2014. A first look at 802.11ac in action: Energy efficiency and interference characterization. In *2014 IFIP Networking Conference*. 1–9. https://doi.org/10.1109/IFIPNetworking.2014.6857103
[87] Songyang Zhang. 2018. Congestion Control for RTP Media: a Comparison on Simulated Environment. *CoRR* abs/1809.00304 (2018). arXiv:1809.00304 http://arxiv.org/abs/1809.00304
[88] Xu Zhang, Yiyang Ou, Siddhartha Sen, and Junchen Jiang. 2021. SENSEI: Aligning Video Streaming Quality with Dynamic User Sensitivity. In *18th USENIX Symposium on Networked Systems Design and Implementation (NSDI 21)*. USENIX Association, 303–320. https://www.usenix.org/conference/nsdi21/presentation/zhang-xu
[89] Zhilong Zheng, Yunfei Ma, Yanmei Liu, Furong Yang, Zhenyu Li, Yuanbo Zhang, Jiuhai Zhang, Wei Shi, Wentao Chen, Ding Li, Qing An, Hai Hong, Hongqiang Harry Liu, and Ming Zhang. 2021. XLINK: QoE-Driven Multi-Path QUIC Transport in Large-Scale Video Services. In *Proceedings of the 2021 ACM SIGCOMM 2021 Conference (SIGCOMM '21)*. Association for Computing Machinery, New York, NY, USA, 418–432. https://doi.org/10.1145/3452296.3472893
[90] Anfu Zhou, Huanhuan Zhang, Guangyuan Su, Leilei Wu, Ruoxuan Ma, Zhen Meng, Xinyu Zhang, Xiufeng Xie, Huadong Ma, and Xiaojiang Chen. 2019. Learning to Coordinate Video Codec with Transport Protocol for Mobile Video Telephony. In *The 25th Annual International Conference on Mobile Computing and Networking (MobiCom '19)*. Association for Computing Machinery, New York, NY, USA, Article 29, 16 pages. https://doi.org/10.1145/3300061.3345430

APPENDIX

Appendices are supporting material that has not been peer-reviewed.

A Converge in the Wild (Stationary Case)

In addition to the mobility scenario, we also evaluated the performance of Converge in the stationary case without mobility. In this scenario, Converge utilizes two networks, WiFi and T-Mobile, while WebRTC utilizes a single network (WebRTC-W for WiFi and WebRTC-T for T-Mobile). Figure 16 illustrates how WebRTC and

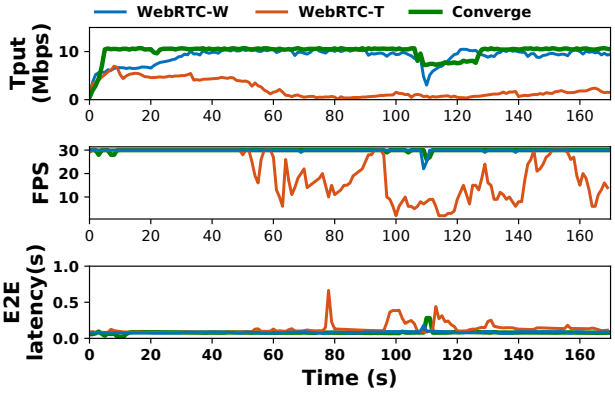


Figure 16: Performance of Converge *vs.* WebRTC for the stationary scenario. Converge uses both network paths while WebRTC uses a single network.

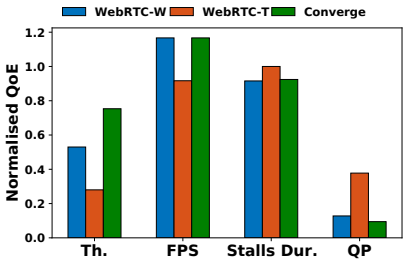


Figure 17: QoE comparison for the stationary scenario.

Converge adapt to time-varying network conditions. With Converge, we can maintain a consistent video throughput of around 10 Mbps, achieve a frame rate of approximately 30 FPS, and experience low variations in end-to-end latency. As expected, when the WiFi network is stable, Converge and WebRTC-W do not show much performance difference. However, there are occasional periods when the WiFi network's performance is poor, leading to disruptions in these metrics (e.g., at 110 seconds) for both Converge and WebRTC-W, with a more significant impact on WebRTC-W.

Figure 17 presents normalized QoE metrics across multiple camera streams. Converge outperforms WebRTC-W by approximately 41% in terms of throughput and surpasses WebRTC-T by a factor of 2.67. This improvement is achieved by leveraging two stable network paths and aggregating them into a single network pipe, providing enhanced throughput regardless of the number of camera streams. Although the gains in terms of FPS and video stalls are minimal compared to WebRTC-W due to the stable network conditions, the quality parameter (QP) is improved by 35% compared to WebRTC-W and is approximately 4 times better compared to WebRTC-T, as Converge achieves better video throughput.

Table 6 reveals that the end-to-end latency of Converge is approximately 7% higher than that of WebRTC-W. This difference can be attributed to the fact that Converge receives a higher number of media packets due to its better performance in terms of throughput and quality. Furthermore, the improvement over FEC is minimal since the stationary case exhibits minimal loss variations. Nevertheless, when compared to a single path, Converge demonstrates

End-to-end latency (ms)			
#	WebRTC-W	WebRTC-T	Converge
1	184 ± 6	224± 3	**176 ± 10**
2	197± 15	231± 4	**210 ± 20**
3	215± 13	254± 7	**249 ± 17**
FEC overhead (%)			
1	0.66± 0.6	4.1± 1.0	**0.05 ± 0.01**
2	1.17± 0.3	5.2± 1.2	**1.02± .06**
3	3.76± 1.6	6.2± 1.5	**1.7 ± 1.0**
FEC utilization (%)			
1	2.5± .95	.45± .02	**8.7 ± 1.5**
2	4.6± .23	.17± 0.08	**9.3 ± 2.7**
3	5.5± .57	0.13± 0.01	**7.5 ± 2**

Table 6: Converge vs. WebRTC (#: # of camera streams)

minimal FEC overhead and better utilization, further enhancing its overall effectiveness.

B RTP packet extension header

We have extended RTP to support multipath by adding fields for path ID, flow-level sequence number, and flow-level transport sequence number, as shown in Figure 18.

C RTCP Packet Header

To support multipath, we have added two fields to the RTCP header: one to identify the path and the other to indicate sequence numbers that are specific to each path for the RTCP report, as depicted in Figure 19.

D Traffic Traces

We have included throughput traces from three scenarios: stationary, walking, and driving. In the stationary case, as shown in Figure 20, T-Mobile only slightly fails to provide the required bandwidth in a few instances. In the walking scenario, as depicted in Figure 21, the variation is smaller compared to driving, but there are regions where either T-Mobile or WiFi falls below the required level. As demonstrated in Figure 22, there is a significant variation in the driving scenario, and even combining the two mobile networks fails to provide the required bandwidth for two short intervals.

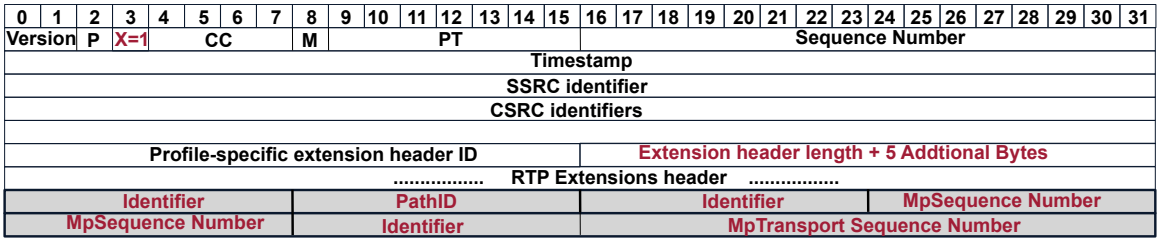


Figure 18: RTP header extension for Converge. Fields in red color indicate the multipath extension.

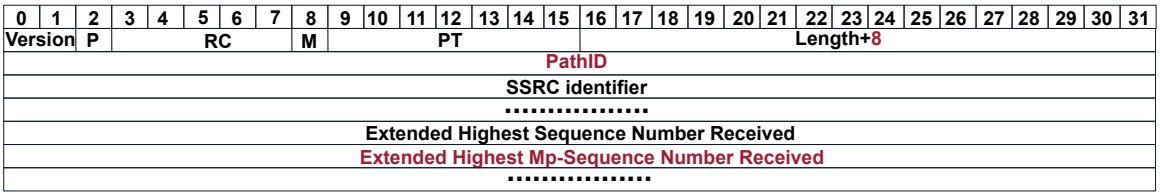
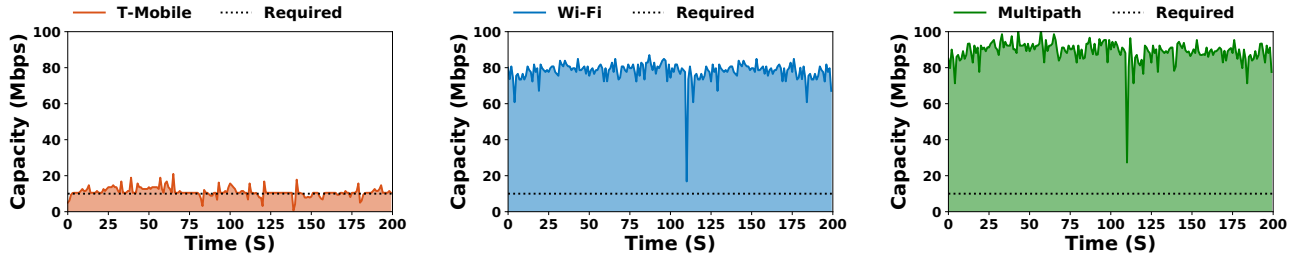
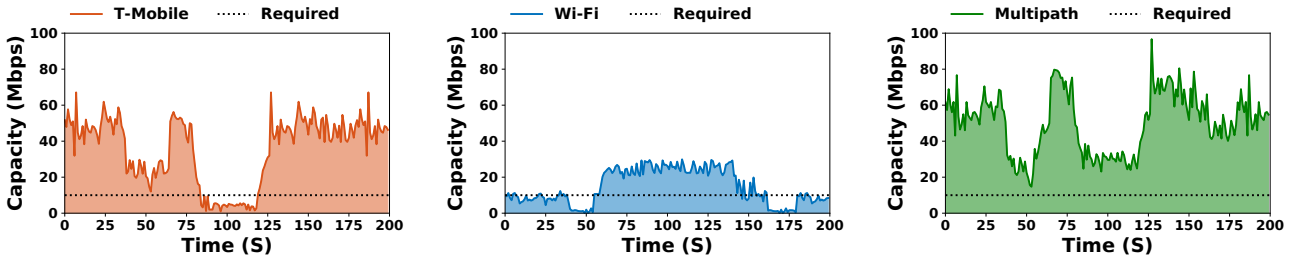


Figure 19: RTCP header extension for Converge. Fields in red indicate the multipath extension.



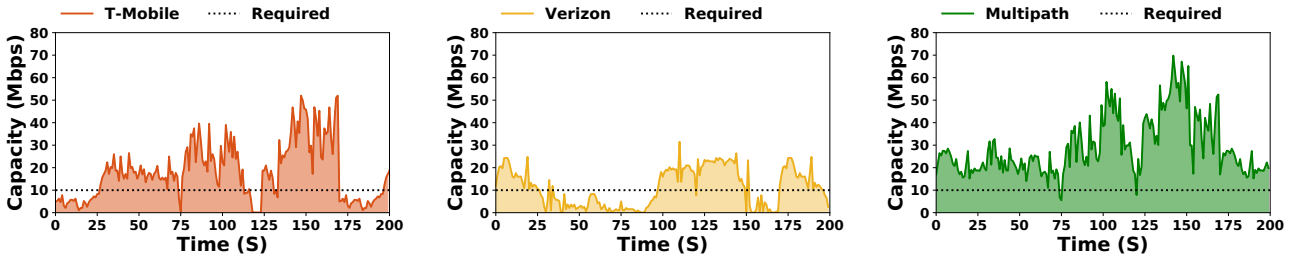
(a) T-Mobile (b) Wi-Fi (c) Multipath

Figure 20: Bandwidth dynamics measured in the stationary scenario.



(a) T-Mobile (b) Wi-Fi (c) Multipath

Figure 21: Bandwidth dynamics measured in the walking scenario.



(a) T-Mobile (b) Verizon (c) Multipath

Figure 22: Bandwidth dynamics measured in the driving scenario.