COMPACT: Content-aware Multipath Live Video Streaming for Online Classes using Video Tiles

Shubham Chaudhary, Navneet Mishra, Keshav Gambhir, Tanmay Rajore
Arani Bhattacharya, Mukulika Maity
Indraprastha Institute of Information Technology Delhi, India
{shubhamch, navneet23049, keshav19249, tanmay19118, arani, mukulika}@iiitd.ac.in

Abstract

The growing popularity of live online classes, even in remote areas, stresses the need for a good and seamless quality of experience to enhance learning. However, these bandwidth-hungry applications challenge the current cellular networks to maintain consistent bandwidth and latency. In this work, we, therefore, propose using the collaboration of multiple devices with their individual cellular networks to support such live video streaming. We design a content-aware system COMPACT that splits video into foreground and background using video tiles (independently encoded spatial blocks) and streams them over different paths. COMPACT depends on its scheduler, which exhaustively searches for the best quality based on the network estimates. We extensively evaluate our system using network traces while walking and traveling on the bus or car. Compared to the single path, COMPACT manages to reduce the median stall and E2E lag by 70.6% and 28.57%, and the tail stall and lag by 83.9% and ≈ 80% on a bus trace. Furthermore, we performed a live experiment to test COMPACT on the actual cellular network.

CCS Concepts

• **Information systems** → **Multimedia streaming**.

Keywords

Multipath, Live Video Streaming, Video Tiles, Live Online Class

ACM Reference Format:
Shubham Chaudhary, Navneet Mishra, Keshav Gambhir, Tanmay Rajore, Arani Bhattacharya, Mukulika Maity. 2025. COMPACT: Content-aware Multipath Live Video Streaming for Online Classes using Video Tiles. In *ACM Multimedia Systems Conference 2025 (MMSys'25), March 31-April 4, 2025, Stellenbosch, South Africa.* ACM, New York, NY, USA, 13 pages. https://doi.org/10.1145/3712676.3714451

1 Introduction

In recent years, the popularity of online learning has increased [16, 27, 65, 95]. Students prefer using handheld devices for online learning [1, 15] primarily due to the availability of easier internet access on smartphones. Mobile learning enables students of even remote areas to listen to lectures from renowned universities and/or colleges. Live video streaming is the technology serving online learning [22, 26, 27, 34, 69, 90], supported by platforms such as YouTube and Twitch. Such services typically provide video content with a latency of the order of 5-10 seconds [8]. However, this growth in live video streaming for online learning comes with the challenge of providing students with a good quality of experience (QoE) to enhance their learning. Since most live-streamed online classes are interactive, where the students can ask questions either verbally or through chat or Q&A, all the modalities (text, audio, and video) should be in synchrony to maintain the live experience [26, 34]. The current wireless infrastructure falls short of catering to these bandwidth-hungry video applications. Although the newer generations of wireless technologies like 5G, 6G, and WiFi-7 promise to offer high bandwidth [18, 93, 97, 98], live video streaming still suffers from high latency and poor QoE [40, 41, 55, 57].

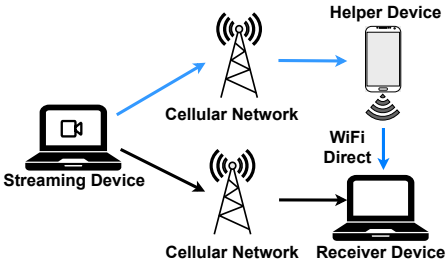


Figure 1: A schematic view of COMPACT indicating in blue the lines for additional connectivity we intend to add.

Several works [32, 33, 82, 83, 86, 87, 89, 91, 99] show the potential of using multiple connections to address the shortcomings of video streaming. Multipath streaming permits leveraging multiple available paths in conjunction. In addition, if one path goes down or has poor bandwidth (BW), the other can compensate, given it has relatively superior network characteristics [61]. Most students have access to multiple devices in their homes [3, 5, 21, 70, 77]. Live video streaming can benefit by aggregating all these devices' bandwidths. We show such a system in Fig. 1 where the user has two devices[1], a phone and a laptop connected to the Internet through two different Internet Service Providers (ISPs). Conventional techniques use only one device (called primary) at a time to retrieve video packets (shown with black arrows). Having another supporting device (called a helper) in parallel facilitates multipath transmission (shown with blue arrows). The user's devices communicate within themselves using WiFi Direct/WiFi Peer-to-Peer connection. The use case for such a multi-device setup has been utilized and

[1]As a proof of concept, this paper shows multipath with two devices only, which can be extended beyond. We leave it for future consideration.

MMSys'25, Stellenbosch, South Africa

ACM ISBN 979-8-4007-1467-2/25/03
https://doi.org/10.1145/3712676.3714451

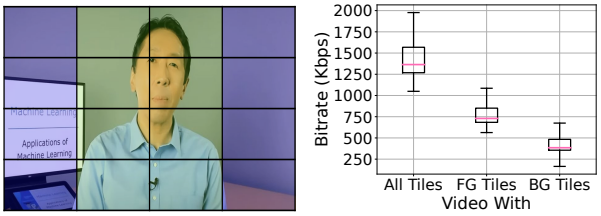
incentivized by MPBond [100], MicroCast [49], and OASIS [46], but they did not use it for live streaming.

Current multipath live streaming techniques utilize the extra bandwidth by distributing packets over the paths without considering the video content. Converge [32] used multipath WebRTC to meliorate video conferencing. Furthermore, they showed that if the current multipath protocols like MPTCP [66, 67, 85], MPQUIC [29–31], and MPRTP [72] is used as is for live video conferencing, they perform worse than single-path due to head-of-line (HoL) blocking caused by out-of-order packet arrival. TwinStar [83] sends frames in a round-robin fashion on the available paths, allocating bitrate jointly while encoding. AggDeliv [33] optimizes congestion control of the multiple paths to adapt them for mobile live video delivery. However, none of these techniques considers the relevance of spatial content. Therefore, these systems encode the whole content at a lower bitrate (BR) to accommodate it within the available BW.

The Case for Content-Aware Streaming: We first argue that the content of video matters in terms of how packets should be scheduled over a multipath network. To illustrate this argument, we consider the case of a classroom where an instructor is teaching. The body, face, and gestures of the instructor are what the students focus on the most, indicated by the fact that it has the highest saliency [50, 78, 84]. Thus, these portions of the frames/video should have the highest quality. Therefore, streaming should be content-aware. Note that just cropping the face or blurring the background is not prudent as it impairs the QoE. This is because online classes not only have scenes with the instructor talking but also slides and whiteboards. Besides, occasionally, we even see instructors' videos hovering over the background (BG) with slides. These use cases make sending and rendering the BG with the foreground (FG) indispensable. However, since the frequency of changes in the BG is lower and delays in rendering the changes in the background are not as significant as those in the instructor's body, face, and gestures, rendering an outdated BG suffices. Several works like [17, 20, 28, 48, 88] utilized content awareness in sports, lectures, gaming, and surgical videos. However, they focused only on playback speed control based on the content type and not on scheduling over multipath networks.

Our Solution: To resolve the problem of unreliable cellular networks, we propose Compact that uses content-aware multipath live video streaming for live online classes where we stream spatially segregated video content for transmission over different paths. For example, considering the case of the classroom, the portion of the video containing the teacher's face/body, the foreground (FG), is streamed at high quality through the path with the highest bandwidth and least latency (referred to as the primary path). The relatively static background (BG) content can be sent through the path with lower throughput, referred to as the secondary path. Compact reasons rendering the current FG with the previous BG in case the BG gets delayed, as the changes in BG are less frequent. Since modern smartphones have two interfaces suitable for video streaming – cellular and WiFi, it is sufficient to consider only a binary choice (FG and BG) to obtain an efficient solution.

To segregate a video into two different categories, FG and BG, we use High-Efficiency Video Codec (HEVC/H.265) [76] as it supports creating *spatially independent rectangular blocks in a video called tiles* (shown in Fig. 2a). Tiles can be removed, swapped, and/or



(a) (b)

Figure 2: (a) Video with 4×4 tiles. The foreground is highlighted in green and the background in blue. The video is publicly available under Creative Commons License [54]. (b) The bitrate of the full video is ≈1.7× and ≈4× the bitrate of foreground and background content, respectively.

added independently in real-time without affecting the rest of the video, provided the initial encoding uses tiles. Note that segregating only the speaker is computationally intensive, requiring image segmentation. Due to the changing BG, albeit with less frequency, BG subtraction gives poor accuracy. Using tiles gives us coarser yet faster FG detection using a face-detection model. We create two subsets of tiles; one has only the FG, typically containing the face and body of the speaker, while the other contains only the BG.

Content-aware Scheduler: Choosing the right path for the set of tiles is crucial because a wrong decision can lead to HoL blocking due to out-of-order delivery of tiles. For instance, if the BG tiles reach earlier, we would still have to wait for the FG tiles, causing unnecessary stalls. We found in our experiments that using conventional schedulers like MinRTT and Musher [68], used in MPQUIC and MPTCP, respectively, do not improve but lead to worse QoE (§2.3). Because the number of tiles to stream should depend on the asymmetric path characteristics. In cases where the FG set has fewer tiles than the BG set, a few tiles from the BG can be shifted to the FG if the path over which FG is being streamed has sufficient BW. Since conventional schedulers lack such knowledge, they fail to leverage the full potential of multiple available paths. Furthermore, the quality of the two tilesets can be varied independently based on the BW if streaming at a certain quality level is not possible.

Compact utilizes its own scheduler to fulfill the above requirements. The scheduler keeps track of the round-trip-time (RTT) and available bandwidth (BW) estimates of each path (we use two paths in this paper). Depending on the RTT and the BW estimates, it either shifts tiles from BG to FG, varies quality level, or does both. It finds an appropriate schedule by maximizing the utility function.

We implement Compact at the user level in Java using Stream Control Transmission Protocol (SCTP) [59, 75] under the hood. We use SCTP due to its in-order packet delivery with configurable retransmissions at the stream level to fix transport-level HoL blocking. Several prior works [86, 87, 89, 91] used it for video streaming applications. We evaluate Compact on the real network and in different scenarios using traces replayed using Mahimahi [60]. We find that Compact performs superior to the baselines, improving the median stall and E2E lag by 70.6% and 28.57%, and the tail stall and lag by 83.9% and ≈ 80% on a bus trace.

Contributions: Our main contributions are as follows:

(1) We propose to leverage the different levels of importance of the spatial regions in streaming videos for online classes. This helps

split the videos into foreground and background categories, which can be streamed through different network paths. To the best of our knowledge, we are the first to utilize such spatial segregation for multipath live video streaming.

(2) We implement our own content and path-aware scheduler to address the shortcomings of the past multipath schedulers. COMPACT is open-sourced on Github[2] to spur further research.

(3) We evaluate COMPACT on diverse network conditions, including a live test on the actual cellular network. It reduces the median stall by 3.4× and the 99*th* percentile tail stall by 6.2× compared to the single path. It further improves the median E2E lag by 28.57% and tail lag by ≈ 80%.

2 Background & Motivation

In this section, we first briefly discuss video tiles and their use cases in live video streaming. We then discuss the challenges of using tiles for multipath live streaming. Lastly, we study the use of conventional packet-level multipath schedulers at the tile level.

2.1 Video Tiles for Streaming

Modern video compression standards like HEVC [76], AV1 [39], and VVC [24] support encoding videos by splitting them spatially into a grid of rectangular blocks referred to as *tiles* (Fig. 2a). These tiles are treated as independent entities, i.e., they can be manipulated without affecting the rest of the video. Due to the ease and flexibility in manipulation, tiles are widely used in 360° and VR videos [42, 64, 74, 94]. Tiles also have the advantage that removing them reduces video filesizes [25]. We show this in Fig. 4, where shifting a certain number of tiles from BG to FG increases FG filesize while decreasing BG filesize. This allows adjusting the number of tiles in each path to handle the differences in network bandwidths. Since tiles allow independent decoding, the FG can be rendered without waiting for the BG to arrive, reducing the overall latency. Tiles can also be encoded at different quantization levels, thus allowing another parameter to adjust according to the available network BW.

The separation of spatial content into FG and BG tiles reduces the overall video streaming bandwidth requirement. We show this in Fig. 2b where for a total of 400 video segments (a bunch of frames when encoded is called a video segment), the FG tiles bitrate (BR) is ≈ 1.7×, and BG tiles bitrate (BR) is ≈ 4× lesser than the video with all tiles. This reduction in video bitrate demonstrates the utility of coupling tile-based streaming with multipath networking. This implies that if streaming a high-quality video over a single path is not viable, resorting to FG and BG tiles for multipath streaming makes it feasible without losing out on the quality of experience.

2.2 Tile-level versus Packet-level Streaming

Relying on tile-level streaming for live video delivery helps eliminate the packet-level HoL-blocking. We intuitively show this using Fig. 3. Consider the case of conventional multipath streaming at packet-level, here the encoded video segments are bifurcated into smaller packets which are distributed across the multiple paths based on the network characteristics. At the other end (receiver-side), all packets should be received to decode and render the streamed segment. If any packet gets delayed, lost, or arrives out

[2]https://github.com/shubhamchdhary/COMPACT

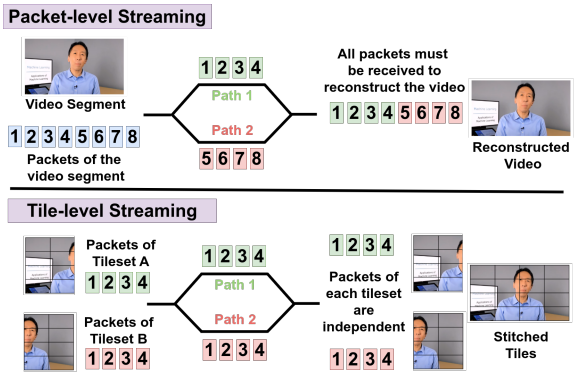


Figure 3: Difference between tile and packet-level multipath streaming. Streaming at the packet level distributes packets across the paths and mandates reception of all the video packets for video reconstruction. While in tile-level streaming, all packets of a tile-set are streamed on a single path, making it independently receivable and decodable.

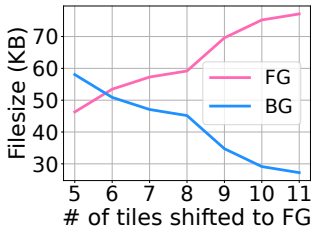


Figure 4: Shifting tiles from background (BG) to foreground (FG) reduces BG video size proportionally.

of order, the whole video reconstruction is hampered, leading to video stall/freeze. Moreover, if the paths are highly heterogeneous, the packets of the worst path will likely get delayed. Thus, besides reducing the video quality and scheduling fewer packets on the worst path, there is no way to reduce video stalls.

Piggybacking on tile-level streaming helps resolve the packet-level HoL-blocking. This is true because, unlike distributing packets across multiple paths, in tile-level streaming, all the packets of a tile are streamed through one single path. This eliminates the possibility of HoL-blocking due to inter-path packet delay. We show this in Fig. 3 where the video is divided into two tilesets, A and B. *Tileset A* is streamed over *Path* 1 while *Tileset B* is streamed over *Path* 2. All the packets of a tileset are sent over a single path. Since tiles are independently decodable, even if the paths are highly heterogeneous, the tileset reaching first can be rendered without waiting for the other. Thus, it reduces the video stalls and allows FG and BG to be encoded at different quality levels.

2.3 Limitations of Conventional Schedulers

Fig. 5 demonstrates the stall duration and the application level end-to-end (E2E) lag for different packet level and tile level schedulers (refer to §4 for details of the metrics and the schedulers' implementation). We run the experiment on a 5-minute lecture video. We replayed Pensieve's [58] open-sourced bus trace files (bandwidth patterns shown in Fig. 11b) on both paths. We note that all the packet level schedulers PMin (MinRTT), PBF (Balanced Subflow

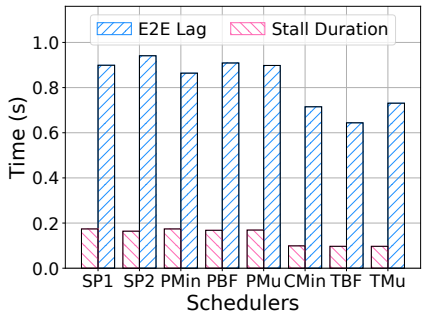


Figure 5: Median end-to-end (E2E) application-level lag and video stall duration of different packet-level (PMin, PBF, and PMu) and tile-level (CMin, TBF, and TMu) multipath schedulers compared to individual single paths (SP1 and SP2).

Completion), and PMu (Musher) incur the highest median lag and introduce median stall similar to single path streaming. This is evident because the path heterogeneity causes HoL blocking due to out-of-order packet arrivals. Thus, conventional packet-level schedulers fail to fit for live video streaming.

If these schedulers are modified to work at the tile level, then all of them (CMin, TBF, TMu) perform better than packet-level schedulers (PMin, PBF, PMu) and single paths (SP1 and SP2) in terms of both E2E lag and stall. However, this improvement is marginal because of the tile-level HoL blocking. Although the tiles sent over the better path reach earlier, it has to wait for the rest to arrive, causing unnecessary lag. This problem persists even if the scheduler is content-aware, as illustrated by CMin (Content-aware tile-level MinRTT) being worse than TBF (tile-level balanced subflow). This example shows that being content-aware is not enough. This is due to the dependence of the FG on the BG. TBF achieves a superior tradeoff among all conventional schedulers. It manages to do so because it tries to equalize the completion times (time to send all the tiles of a set) of tiles across paths. TMu (tile-level Musher) behaves similar to CMin even though it distributes tiles in proportion to the available bandwidths. Therefore, content-aware schedulers need to distribute the tiles well to minimize the completion time.

2.4 Challenges of Using Tiles

While using tiles for multipath live streaming has advantages, three major challenges are associated with tile-level live streaming.

C1) Content-Obliviousness of Tiles: Tiles themselves do not have any knowledge of their content. Using an arbitrary splitting technique to get FG and BG sets leads to suboptimal performance (discussed in §5). Since there is no foreground and background, the receiver device is, therefore, obliged to wait for all the tiles to arrive before stitching. This causes unwanted HoL blocking leading to high E2E lag and stall. Thus, it is imperative to divide tiles depending on the content so that if only the FG tiles reach, it is renderable with the recently cached BG tiles.

C2) Requirement of Fair Allocation of Tiles: After creating two sets of tiles, the BG can have more tiles than the FG. Since we stream BG over the low bandwidth path, it may not fit within the bandwidth limit. In such cases, instead of encoding the BG tiles with a lower bitrate, some of its tiles can be allocated to FG because shifting reduces filesize (Fig. 4). Such relocation of tiles gives the opportunity to stream BG tiles at relatively higher video quality as they are shifted to the FG tileset. It is, therefore, crucial for the system designer to consider appropriate tile allocation.

C3) Dependencies among FG and BG Tiles: Upon receiving all the tiles, the receiver stitches them as a single video and renders them on the screen. However, delayed reception of BG tiles can cause unwanted video stalls and HoL blocking. Although playing only the FG tiles is possible, it leads to worse QoE. It is important to break the strong dependency between FG and BG tiles.

3 Design of Compact

Compact's design is based on the following key observations: (1) at a particular time and location, all ISPs generally do not suffer equally from poor throughput [35], (2) a significant number of people often carry multiple smart devices that can be collaborated to leverage their aggregated bandwidth [77], (3) background in live online classes is mostly static, and changes are less frequent than the foreground, therefore, FG can be rendered with an outdated BG, (4) content-aware segregation of tiles allows streaming at a higher bitrate (better quality), and (5) BG can be encoded at a lower bitrate than FG because people pay more attention to FG [20, 48].

3.1 System Architecture

We show Compact's architecture and workflow in Fig. 6. At the streamer side, the camera sends raw frames to the system where *FG Detector* identifies the tiles containing any human face using a DNN (§4). Thereafter, the *Scheduler* uses the marked tiles to create FG and BG sets with tile indices. Based on the network statistics (RTT and BW), the scheduler decides what quality/bitrate to keep for the two categories (FG & BG), along with the number of tile shifts if needed (§3.2). Besides, it chooses the path with the minimum completion time = $\left(\frac{RTT}{2} + \frac{Filesize}{BW}\right)$, called the primary path, to stream the FG tiles. The other path is called the secondary. *BG tiles need not be streamed through the helper device only. It can be sent through the direct path too, depending on the network characteristics.*

Finally, *Scheduler* notifies the *Tile Segregator*, which passes the raw frames received from the camera to the FG and BG encoder to generate two tiled video segments, one for FG and another for BG. These encoded segments are stored in their corresponding sending queues (*FG Segment Queue* and *BG Segment Queue*). Although, called *FG Tile Encoder*, the encoded segments can also have a few relocated tiles of BG. Subsequently, the *Streamer* module streams the video segments from the queues over the network through primary and secondary paths. In addition, it performs another important task of maintaining the network's RTT and BW estimates of each path using separate probe channels discussed in the next subsection. The *Helper Device* just relays the received tiles to the *Primary Device*.

As on the streamer side, the received tiles are first stored in queues on the primary device (where the user watches the video). The decoders for FG and BG fetch the segments sequentially from these queues and pass on the decoded frames to the *Tile Stitcher*. On receiving the FG and BG frames, the stitcher clubs these two subframes together to play as a single video (shown in Fig. 7). At times when the BG segments get delayed, the stitcher uses the last

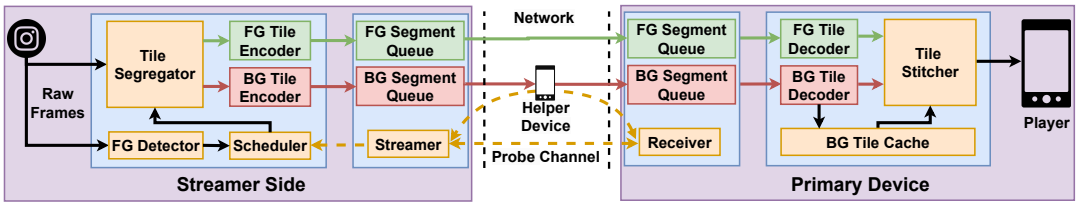


Figure 6: Architecture and workflow of COMPACT.

rendered BG stored in *BG Tile Cache* for rendering. The *Receiver* module communicates with the *Streamer* to echo back probe packets required for network RTT and BW estimation.

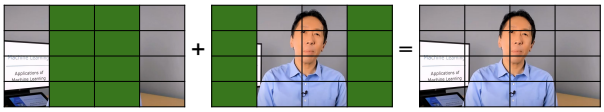
To get the right schedule, having a good enough estimate of network characteristics like RTT and BW is crucial. We create a total of four probe channels, two for RTT and two for BW. To estimate the RTT of each path, we send timestamps from the streamer machine to the primary, which is echoed back. The difference between the receiving time and the sent timestamp is the RTT. We attach timestamps in the custom headers of the sent video segment tiles to probe the network BW. The primary device dissects the header and replies with the demarcated timestamp. We use this timestamp to calculate the total sending time to infer network BW. The instantaneous RTT and BW are noisy in nature. Similar to prior works [45, 96], we use the harmonic mean (HM) of the past five values to smoothen the estimates.

Once the tiles are streamed from the sender based on an appropriate schedule, they might arrive at different moments of time at the primary device. If the BG tiles are delayed too much (more than inter-frame delay), it can lead to stalls because we cannot render the FG tiles alone. Therefore, we need some way to handle the out-of-order arrival of the two sets of tiles. In online classes, the background (slides, whiteboard, or texts) is mostly static, and the changes are less frequent than those of the speaker (foreground). Therefore, the same BG can be rendered for multiple foregrounds. This opens the possibility of breaking the tight coupling between FG and BG tiles. In cases where the BG tiles are delayed, the previous BG can be stitched along with the current FG tiles to render.

3.2 Tile Scheduler

We now discuss our content-aware scheduler. Once we get the two sets of tiles for foreground (FG) and background (BG), respectively, timely delivery of both of them to the player side is crucial. Any delay caused increases the stall duration between two consecutive segments. For a video encoded at $25fps$, the inter-frame delay δ is $40ms$. Any extra time required beyond $40ms$ is, therefore, considered a stall/freeze. To have the best quality of experience, the quality (bitrate) of the played video should be highest with the least stall and quality switches within and across the segments. Borrowing from prior works on tiled streaming [62], we define a utility function to capture users' QoE as a weighted sum of user-perceived video quality, stall, inter and intra-segment quality switches.

Let there be total N tiles with m tiles in FG and n tiles in BG set. The variables Q_{FG} and Q_{BG} denote the quality defined in terms of the video quantization parameter (QP) of tiles in FG and BG, respectively. Let the size of each tile in FG and BG be $R_{FG}(Q_{FG})$ and $R_{BG}(Q_{BG})$ for a certain quality level Q_{FG} and Q_{BG} of foreground and background.



(a) (b) (c)

Figure 7: Tile stitching: (a) background tiles and (b) foreground tiles stitched into a single video in subfigure (c).

Filesize of Tiles: We note that our scheduler maps all tiles of FG and a limited number of tiles of BG to the primary path. Assuming a total of o BG tiles use the primary (i.e., the higher BW) path, the data F_P and F_S sent through primary and secondary paths are:

$$F_P = R_{FG}(Q_{FG}) \times m + R_{BG}(Q_{BG}) \times o; F_S = R_{BG}(Q_{BG}) \times (n-o) \quad (1)$$

User-perceived Video Quality: The video quality Q_t of a segment t, the user observes, depends on the quality of the FG and BG tiles rendered. Mathematically, if β_1 and β_2 denote the weights given to the quality of FG and BG, respectively, the video quality Q_t is:

$$\begin{aligned} Q_t &= \beta_1 Q_{FG} + \beta_2 Q_{BG} \\ &= \beta_1 (m+o) q_t(l_{FG}) + \beta_2 (n-o)(\mathbb{1}(K) q_t(l_{BG}) - p(t)) \end{aligned} \quad (2)$$

Here, the function $q_t(.)$ returns a normalized quality value for a given quantization level l_{FG} and l_{BG} for the FG and BG, respectively. In case of delayed delivery of BG tiles beyond a threshold, the player can choose to render received FG tiles by stitching them with the previous segment's BG tiles, caching the current ones. This depends on the completion time captured using the indicator function $\mathbb{1}(K)$ which is 1 if $\left(\left(\frac{RTT_S}{2} + \frac{F_S}{BW_S}\right) - \left(\frac{RTT_P}{2} + \frac{F_P}{BW_P}\right)\right) < \delta$ else 0. The path with the least completion time is referred to as the primary path P where FG tiles are scheduled. Here, BW_P and BW_S are the estimated bandwidths, and RTT_S and RTT_P are estimated RTTs of primary and secondary paths, respectively. $p(t) = max(Q_t)$ penalizes if the BG tiles of past segments are rendered. We set β_1 and β_2 to 0.6 and 0.4, respectively to prioritize FG tiles.

Quality Switches: The quality levels of the tiles on the player side can be different across and within a segment. There should be a graceful degradation of quality for better QoE. We use I_1 and I_2 to penalize abrupt changes to control inter-segment and intra-segment quality switches, respectively. We define these as:

$$I_1 = |Q_t - Q_{t-1}| = |q_t(l_{FG+BG}) - q_{t-1}(l_{FG+BG})| \quad (3)$$

$$I_2 = |Q_{FG} - Q_{BG}| = |q_t(l_{FG}) - q_t(l_{BG})| \quad (4)$$

Stall Duration: The stall duration depends on the completion time of a video segment. A stall occurs when the tiles of the current segment reach after the play time of the previous segment P_{t-1}. Therefore, assuming primary path (P) is the fast path, for a segment

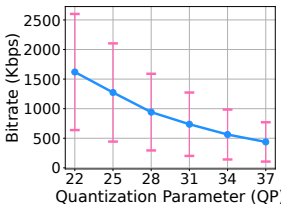


Figure 8: Video bitrate reduces on increasing quantization parameter (QP).

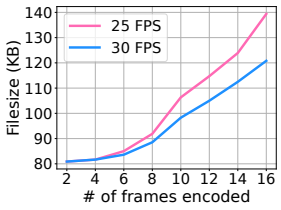


Figure 9: Increasing the # of frames in the encoded video segment increases filesize.

t the stall L_t can defined as:

$$K_{max} = max\left(\frac{RTT_P}{2} + \frac{F_P}{BW_P}, \mathbb{1}(K)\left(\frac{RTT_S}{2} + \frac{F_S}{BW_S}\right)\right) \quad (5)$$

$$L_t = max((K_{max} - P_{t-1} - \delta), 0) \quad (6)$$

Here, K_{max} is the maximum of the sending times on the primary and secondary paths, indicator function $\mathbb{1}(K)$ becomes zero when all BG tiles are shifted to FG, otherwise, it is one. The objective of the scheduler is to split the total tiles into primary and secondary paths in such a way as to maximize the utility function below.

$$\textbf{Maximize } Utility = \alpha_1 Q_t - \alpha_2 I_1 - \alpha_3 I_2 - \alpha_4 L_t - \alpha_5 S_t \quad (7)$$

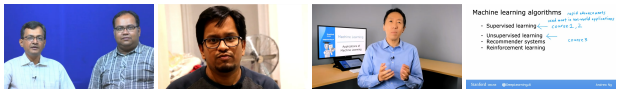
Here, S_t is the absolute difference in the number of tiles shifted from the background to the primary path between the last and the current segment. It refrains from abrupt tile relocations between two adjacent segments. We borrow the weights assignment strategy of [19, 58] keeping $\alpha_1 = max(Q_t)$, α_2 to α_4 equal to 1, and make $\alpha_5 = 0.1$. We use an exhaustive search over all possible tile shifts, FG, and BG QP changes to get the right schedule for every segment, which takes <= $1ms$.

Moreover, we need to estimate the filesize of tiles for each QP value while scheduling because the segment is encoded after getting the right schedule. To estimate the filesize, we rely on Fig. 8, which depicts that for a QP reduction of 3, the filesize/bitrate increases by 1.2×. We use the filesize of the previously streamed segment and a multiplying factor of 1.2 to approximate the filesize for every QP during the exhaustive search.

4 Implementation

We implement the host/streamer and the joinee/viewer side of the Compact in Java. We use a Linux machine as the host device and another two Linux PCs as helper and primary devices. We now discuss each of the components in detail.

(1) **Video Encoding:** Since Compact uses tiled videos for streaming, we use an open-source HEVC encoder, Kvazaar [80], to encode the raw frames into a grid of 4×4 tiles. Other configurations are also possible. However, we empirically found this configuration works well for us as it balances encoding overhead and stitching complexities similar to prior works utilizing tiles [25, 36]. Our encoding pipeline begins by capturing raw frames from the screen (simulating camera) using FFmpeg [2]. We then encode these frames into a tiled HEVC video. We use GPAC [52] for tile manipulation and packing segments into an mp4 file. This process repeats for each video segment. We keep the framerate fixed at 25 (used by most live video streaming works [9, 32]). Each segment contains four frames balancing encoding time and compression efficiency. We note from Fig. 9 that segments with 4 frames have approximately the same mean filesize as 2 frames, but it escalates faster from 6.



(a) Video 1 **(b) Video 2** **(c) Video 3** **(d) Video 4**

Figure 10: Screenshots of the four videos used for evaluations.

(2) **Foreground Detection:** We argued in §3 that Compact is content aware. We use a pre-trained ResNet model available at [7] for face detection to detect the FG content. Once we have the coordinates of the bounding boxes of the detected faces, we identify the tiles containing these boxes and consider them the FG tiles and the rest the BG tiles. In the case of presentation slides, the model detects no human face, we keep a horizontal split with eight tiles in both FG and BG.

(3) **Transport Layer:** We use Stream Control Transmission Protocol (SCTP) to stream the video segments because it provides in-order delivery, configurable retransmission, flow, and congestion control. We use UDP under the hood for all the control channels. Apart from SCTP, QUIC [51] is another transport protocol that precludes transport layer HoL-blocking. QUIC is built on top of UDP borrowing ideas from SCTP. Since SCTP is a relatively older protocol, multiple live-streaming techniques [44, 81, 86, 87, 89, 91, 92] have used it. QUIC, till date, is mostly explored in video-on-demand setups. We intend to explore the feasibility of its multipath variant [29–31] in the future.

(4) **Helper & Primary Device:** The helper's job is just to relay the received tiles to the primary device. To keep the overhead minimal, we use the Linux utility *socat (SOcket CAT)* [6], which acts as a socket-based bidirectional stream relay running on the helper. The primary device receives all the tiles, stitches, and plays them. Once the tiles of both paths are synchronized, we use OpenCV to stitch/join and render the FG and BG tiles.

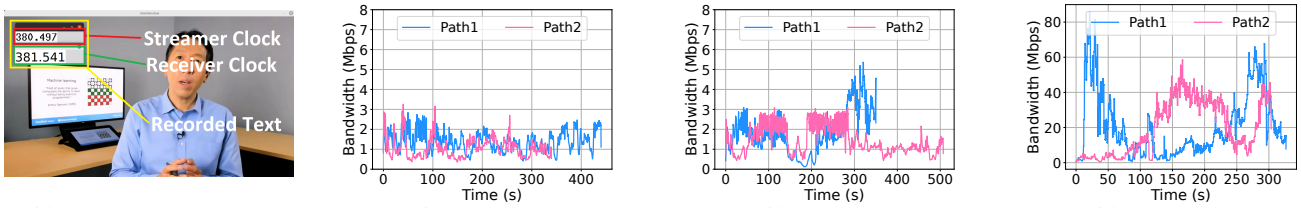
The different tasks performed on the streamer and the receiver devices are asynchronous. We use multi-threading to parallelize the tasks. For instance, the video segment encoder can independently capture and encode videos, while the sender thread can parallelly stream the encoded videos. The encoder should not wait for the sender thread to complete sending. We use evicting queues[3] of size two to exchange information between threads.

Baselines: We compare Compact with the different conventional schedulers at both the packet level and tile level. All the tile-level schedulers, except CMin, are unaware of the video content.

1) Single Path: For the single path, we always stream on one of the paths without any scheduler. We denote as SP1 when using only Path 1 and SP2 when streamed on Path 2 only.

2) MinRTT: This is the default scheduler in MPQUIC and MPTCP. At the packet level (PMin), it sends all the segments' chunks (MTU size packets) through the path with minimum RTT. We even make it content-aware at the tile level (CMin) to show that without tile shifting and out-of-sync FG and BG playback, it behaves similar to oblivious tile-level schedulers. CMin sends FG tiles to the minimum RTT path and BG tiles to the other one.

[3]FIFO queues with auto-dequeuing on exceeding queue size are referred to as evicting queues.



(a) E2E Lag Computation (b) Bus trace (c) Car trace (d) Walk Trace

Figure 11: (a) Screenshot showing both the streamed digital clock and the local one. We record only the portion inside the yellow box for text detection to calculate lag. (b), (c), and (d) depicts the bandwidth patterns of the replayed trace for the experiments.

3) Musher: This is another MPTCP scheduler that allocates packets proportional to paths' bandwidth (PMu) [68]. At the tile level (TMu), we assign tiles based on the available bandwidth of the paths.

4) BFlow: We implement balanced subflow completion (DEMS [37]) at both packet (PBF) and tile level (TBF). The core objective of BFlow is to minimize the difference between the completion times of the sent data (packet for PBF and tiles for TBF) on both paths.

Dataset: We selected four lecture videos from YouTube [11–13, 43], each lasting 4-5 minutes. Fig. 10 shows screenshots of the videos used for the experiments. The videos have a mix of scenes, with Video 1 having two speakers delivering a lecture. Video 2 and Video 3 have one speaker in the FG with different BGs. Video 4 is similar to Video 3, with the only difference being that it has slides showing up frequently. We used these videos as they represent the typical online class setup and provide us with sufficient diversity to check the robustness of our technique. For each session, we replay these downloaded videos with a live digital clock to calculate E2E lag.

Performance Metrics: For extensive evaluations, we employ E2E lag and video stall to quantify the quality of experience. We do not report the quality of baselines individually in the evaluation because we kept it identical to Compact. This ensures a fair comparison of factors, such as end-to-end latency and stalls. Thus, we first ran Compact, logged its FG and BG quality for each segment, and then reused it in all other baselines while streaming.

Our application-level E2E lag is defined as the difference between the time a frame appears on the streamer screen/camera and when it reflects on the viewer's screen. It depicts the actual lag that the user perceives. To measure E2E lag, we follow the strategy shown in [79], where we put a live digital clock on the video we stream. We also run the same clock on the receiver end and put it on the playing video. Afterward, we use a screen recorder to capture only the portion of the screen displaying the clocks (yellow box in Fig. 11a). Thereafter, we extract frames from the recorded video and run PyTessaract [53] OCR to recover the clocks' text. The difference of the detected texts is the E2E lag. While recording the clock texts, we reduced the capture rate to $10fps$ to reduce the number of OCR inferences, which is time-intensive. Note that we discard a few frames where the OCR fails to detect texts ($\approx$ 8% on average). Our stall represents the inter-segment delay, which is the interval between the render time of the last frame and the first frame of two consecutive video segments.

5 Evaluation

We now discuss Compact's performance under various network conditions and the overheads of its different components.

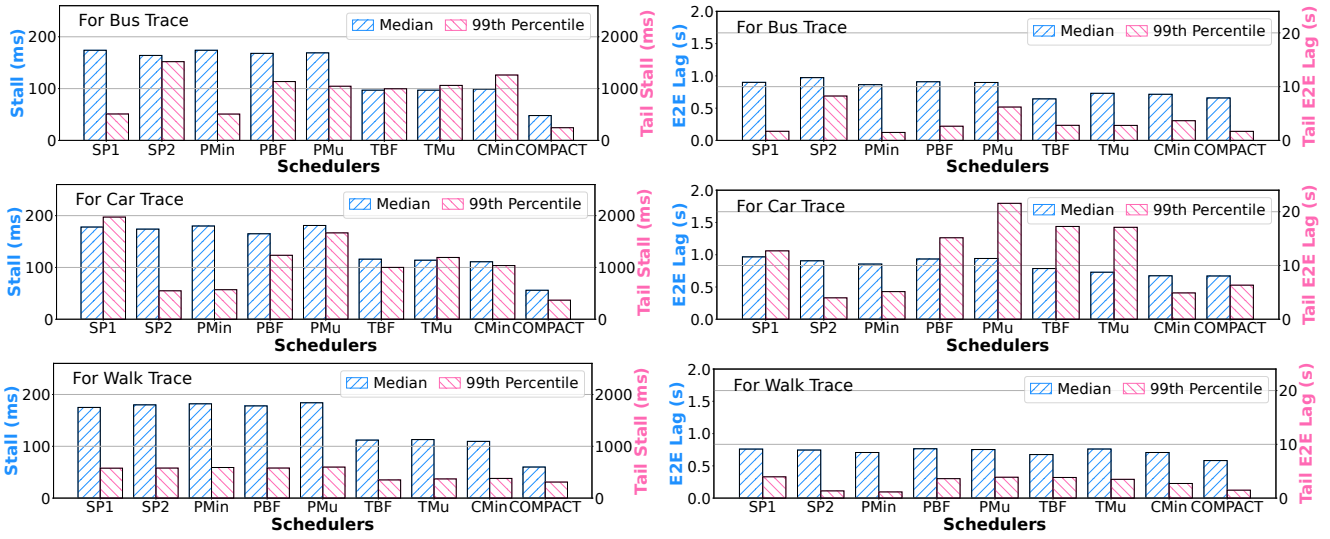
5.1 Trace-driven Results

To extensively evaluate Compact under various network conditions, we use trace files [14] open soured by Pensieve [58] collected while traveling in a bus and a car. Besides, we even collected two walking traces ourselves. We chose these traces to evaluate Compact under challenging network cases with high bandwidth fluctuations and RTTs. Since many countries in the world still witness poor bandwidth (< 3Mbps) [10, 73], we scale down the bus and car trace accordingly to the same range to evaluate for such scenarios. We report the bandwidth patterns in Fig. 11. In each case, we replay two traces of the same category on the two available paths using Mahimahi. If a trace ends early, it gets replayed automatically. We show the video stall and E2E lag in Fig. 12.

Stall: When streaming over a single path, we note that the median stall reaches 180ms on all traces. The median performance of packet-level schedulers (PMin, PBF, and PMu) is similar to or slightly worse than single paths. This is due to the HoL blocking caused by the out-of-order delivery of packets because of path heterogeneity. Therefore, packet-level schedulers fail to leverage the benefits of multiple paths. The network characteristics of bus and car traces are distinct, therefore, the tail values (99th percentile) of single paths differ by $\approx$ 3× while on walk trace, it is almost the same for packet-level schedulers and single paths. Except for the PMin, the rest of the packet-level baselines experience larger tail stalls. Since PMin picks the path with minimum RTT to stream all packets through that path only, it manages to avoid out-of-order packets.

At the tile level, conventional schedulers reduce the median stall by at least 35% compared to the best single path (single path with the least stall) and observe a lower stall than packet-level baselines. Even the basic content-aware tile-level MinRTT (CMin) scheduler shows a trend similar to oblivious tile-level schedulers on all traces, validating that mere content awareness is insufficient. Tile-level schedulers are superior to packet levels because tiles intended for a path are sent only through that path, thus completely removing inter-path HoL blocking because only smaller size BG is scheduled on poorer paths, unlike packet levels that do not consider this. However, the tail stall still dominates due to the dependency between FG and BG tiles.

Compared to all baselines Compact induces the least median and tail stall with 48ms and 245ms, respectively, on the bus trace, 59ms and 379ms on the car, and 59ms and 307ms on the walking trace. On the bus trace, it experiences 3.4× less median stall than the single path best, 3.5× lesser than the best packet-level scheduler (PBF), and 2× than the best tile-level scheduler (TBF). Similarly, the reduction is 2.9× (single path best), 2.4× (PBF), and 1.9× (CMin) on car trace and $\approx$ 3× (single path best), 3× (PMin), and 1.9× (CMin)



(a) Stall (b) E2E Lag

Figure 12: (a) Stall and (b) E2E lag with traces while traveling in a bus, traveling in a car, and walking

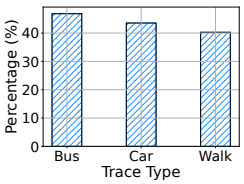


Figure 13: Out-of-sync Playback: shows the proportion of time FG and BG tiles played out-of-sync.

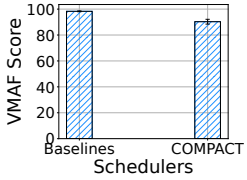


Figure 14: Perceived mean video quality (VMAF score) for all four videos on the bus trace.

for the walking trace. In terms of tail stalls, on the bus and car traces, Compact manages to reduce it by at least 1.5× compared to the single path best and packet-level best, while 2.6× in contrast to tile-level best. On walking trace, its tail is 44*ms* smaller than TBF (best) and 84*ms* than PMin. The asynchronous playback of FG and BG tiles of Compact helps alleviate the video stalls dramatically.

E2E Lag: In terms of E2E lag, on all traces Compact experiences either the least lag or is comparable to tile-level schedulers. However, in terms of tail lag, it is always better than TBF and TMu on all the traces. The lag (median E2E lag) of single paths and packet-level baselines is almost similar in all the scenarios. Tile-level schedulers are marginally better than packet levels in terms of lag. Except for MinRTT (PMin and CMin), the rest of the schedulers witness large tail lags on the car trace. Since the bandwidth is superior in our collected walking trace, all the schedulers manage to keep that median lag below 1*s* and tail lag below 4*s*.

Compact's median lag is 28% lower than single path best, 23% than the lowest of packet-level schedulers, and similar the lowest of tile-level. Compared to TMu and TBF (both perform almost the same), the 99*th* percentile lag is 1.7×, 1.1×, and 2.5× less on bus, car, and walking trace, respectively. We note that all the schedulers are consistently better on the walking trace due to the higher bandwidth (Fig. 11d). The evenness vanishes when the paths' characteristics differ under the bus and car trace. Due to the same streaming video quality, all schedulers encounter a similar median E2E lag. All the above observations vouch for using tiles and multipath-based live-streamed classes.

Out-of-order FG and BG stitching: In the cases where BG gets delayed, Compact stitches the current FG with the cached most recent BG. This helps significantly reduce video stalls and lag. We quantify in Fig. 13 the number of times such out-of-order stitching occurs while running each of the traces for all four videos. We note that these out-of-sync playbacks happened 46.8%, 43.6%, and 40.3% on the bus, car, and walk trace, respectively. These events are fewer on the walking trace due to the more consistent bandwidth compared to the bus and car traces.

Perceived Video Quality: For all the packet and tile-level schedulers, the player renders the streamed video without any changes. Therefore, there is no scope for quality degradation compared to the encoded video streamed from the streamer. However, Compact stitches the previous BG in case it gets delayed. Now, we show that such asynchronous stitching does not hurt the user experience because the BG remains mostly static. VMAF is known to capture human-perceived video quality more accurately than PSNR and SSIM. Therefore, we report the VMAF score. We show the mean video quality of all four videos perceived by the viewer in terms of VMAF score [23] for the bus trace in Fig. 14. We utilize the open source [4] implementation of VMAF for calculations. We note that compared to the baselines, the VMAF score of Compact drops marginally by 8.15% even though it renders a previous BG 46.8% (Fig. 13) of the time on the bus trace. This implies that although out-of-order FG and BG stitching theoretically degrades the video quality, it is marginal in practice. Note that the quality for all the baselines remains the same because we use the same streaming quality for all of them.

5.2 Adaptability to Path Fluctuations

To validate Compact's adaptability to drastic and abrupt network fluctuations, we create two traces for each path. We fixed *Path*1's

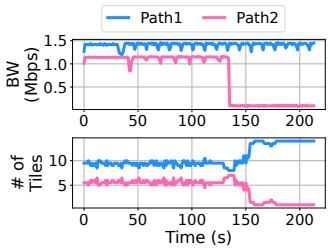


Figure 15: Number of tiles scheduled on each path when network changes drastically.

(a) Our Setup

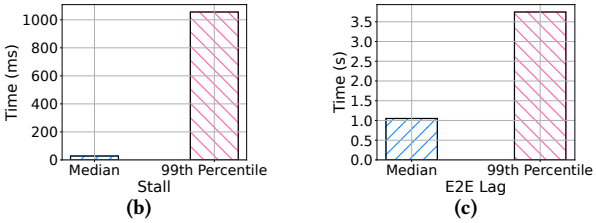


Figure 16: (a) shows our live experiment setup. COMPACT observed (b) stall and (c) E2E lag while streaming over the actual cellular network during the live experiment.

bandwidth to 1.6$Mbps$ and $Path2$'s to 1.2$Mbps$ with a standard deviation of 10$Kbps$. We make the $Path2$'s bandwidth suddenly dip to 100$Kbps$. We replay these traces and run COMPACT for 4 min on a video with only slides. To confirm the desired trace patterns, we run Iperf to get the bandwidths of both paths in Fig. 15.

To capture COMPACT's response to the abrupt change in the BW, we show the number of tiles scheduled on each path in Fig. 15. We observe that before 135s, our scheduler distributes the tiles proportionally to the paths' estimated bandwidths. However, soon after, when $Path2$ went down, COMPACT detected the change and adapted accordingly by gradually shifting most of the tiles to $Path1$ only, demonstrating robustness to sudden network variations.

5.3 Live Experiment

For the live experiment, we run COMPACT on the actual cellular network. As shown in Fig. 1, we keep the same setup and use three Linux desktops: one acting as the streamer to live stream videos (with 15th Gen Intel i7 having 16GB RAM), another as the helper (with AMD Ryzen 5 with 16GB RAM), and the third one as the client playing the video (equipped with 12 Gen Intel i7 with 32GB RAM). All machines access the internet through the cellular network obtained using three smartphones via USB tethering. Furthermore, we connect the primary and helper desktops together using a separate WiFi connection to relay the tiles. In actual deployment, the application on the primary can be made to discover and connect to available helper device. Like trace-driven experiments, we streamed one of the four videos running a live digital clock to get E2E lag and stall (reported in Fig. 16). We utilized the publicly accessible IPv6 addresses from the helper and the client to connect to the streamer. We note from Fig. 16 that the experienced median stall is 28ms with a tail of 1055ms. The median lag is close to 1s with a tail latency of 3.75s. The mean QP for the FG and BG tiles during the streaming was 33. Here, the lag values are worse than trace-driven results because of the poor cellular network with a mean BW and RTT in order of 833.4$Kbps$ and 425ms, respectively.

Table 1: Overheads of different components of COMPACT

Encoding	FG Detection	Scheduling	Stitching
277 ms	56 ms	1 ms	1 ms

5.4 Microbenchmarks

Overheads: We log in Table 1 the mean overheads (in ms) of each individual component of COMPACT. We observe that scheduling and stitching take almost negligible time of 1ms. Tiled video encoding incurs greatest latency due to the software-based encoder Kvazaar. We rule out using a faster preset as it degrades the final video quality. Moreover, screen capture using FFmpeg also contributes to the encoding overhead.

Our encoding happens in two steps: raw frame capture and HEVC encoding, and FG and BG generation, where the former takes most of the time (> 85%). On average, the model used for foreground detection takes 56ms to recognize human faces in the streamed videos. We parallelize detection with the frame capture thread to amortize this overhead.

Video-wise Results: We show COMPACT's median lag and stall on individual videos in Fig. 17 for the bus trace to show the effect of different video types. $Video1$ consists of two speakers with no slides, and the other three videos have only one speaker. Specifically, in $Video2$ and $Video3$, the instructor uses slides to convey whatever he is teaching. $Video4$ has animation playing in the video in between.

We note that COMPACT provides consistently lower lag and stall compared to the baseline schedulers on all the videos. All the packet-level schedulers incur stalls similar to single-path streaming, thus failing to benefit from multiple paths. Due to the absence of inter-path out-of-order packet delivery, tile-level schedulers are better than packet-level schedulers. Among the tile-level schedulers, CMin gives the least stall. The out-of-order playback helps COMPACT to have the least stall. Since only $Video2$, $Video3$, and $Video4$ have frequently changing screens from slides to the speaker, they are encoded at a relatively higher bitrate than $Video1$. Therefore, the E2E lag and stall are highest for them for SP2 (worse path).

6 Related Works

Content-aware video streaming: Streaming a video with consistent quality over a cellular network is difficult due to its random variations in bandwidth, loss, and latency. HotDASH [71] relies on the notion of temporal content awareness for adaptive video streaming. They decide the video's encoding rate based on the network characteristics. This can lead to poor streaming quality due

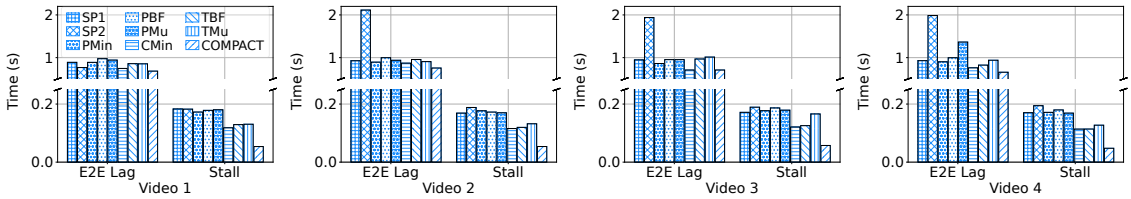


Figure 17: End-to-end (E2E) application-level lag and video stall duration of the different schedulers on individual videos used in the large-scale evaluation for the bus trace. Compact consistently performs better than the baseline schedulers

to the limited bandwidth of the single path. Works like [63, 94] prioritize content based on user viewport for quality adjustment in 360° videos using tiles. ZGaming [88] predicts frames based on content type to improve cloud gaming. However, none of these works employ multipath.

Multipath Video Streaming: Several works [56, 82, 86, 87, 89, 91] utilized multipath to bolster the video streaming performance. XLINK [99] improved the quality of short video streaming using MPQUIC. MPDASH [38] employs MPTCP [66, 67, 85] to use multiple interfaces while streaming video adaptively. MPRTP [72] creates multiple subflows to deliver media packets over multiple paths. However, live streaming is more challenging than video-on-demand due to its tight deadline requirement. Converge [32] and TwinStar [83] used multipath WebRTC for live video conferencing and live cloud gaming, respectively. AggDeliv [33] optimize congestion control in MPQUIC for mobile live video delivery. Chorus [56] integrates adaptive bitrate streaming with multipath networks to optimize the quality of experience. However, none of these works utilize content-awareness to address the packet-level HoL blocking.

Multipath Schedulers: There have been several multipath schedulers like Musher [68], DEMS [37], ECF [47], and DAMS [101] proposed, which work at the packet level. Although DEMS and DAMS consider the deadline requirements, they are incognizant of the content of the packets. Thus, they fail to mitigate inter-path HoL blocking at the application level. Employing these schedulers in live video streaming worsens the situation (§2.3). Therefore, the scheduler tailored for live streaming should be content-aware to distribute packets well across multiple paths.

7 Discussion and Limitations

Having multiple peers: Compact currently supports peer-to-peer video streaming with only one participant. Extending to a multi-participant setup will require getting the network statistics of each attendee for scheduling. This can incur overheads on the streamer side. We intend to look for possible mitigation strategies to amortize the overheads.

Catering to diverse applications: Currently, we show results for live online classes. However, the setup can serve different applications. The major challenge is correct foreground detection, which depends on the video application. A possible direction is to use a saliency-based FG segragator. Moreover, Compact captures a few frames and encodes them as segments (similar to HTTP low latency live streaming) that are streamed over different paths. Although the encoding is parallelized using pipelining, the encoding overhead of the first segment dominates, leading to large latency. One possible mitigation strategy is to stream on a frame basis, as done by WebRTC. Here, frames are captured, encoded, and streamed continuously in real time. Such an architecture can make Compact suitable for video conferencing-like applications.

Accurate bandwidth estimation: Our scheduling decision relies on bandwidth estimation. An underestimation causes poor quality, and overestimation can lead to unwanted video stalls. Compact currently uses a relatively simple technique of bandwidth estimation, leaving the use a of more sophisticated and accurate predictor for future work. An accurate predictor can provide better link utilization, albeit at the cost of more computation.

Scope of using feedback from primary device: One way to address inaccurate bandwidth estimation is to incorporate feedback from the primary device. On receiving a complete video segment, the receiver can send the completion time and other QoE metrics as part of the feedback, which the streamer can utilize to offset its bandwidth and scheduling inaccuracies. Such feedback has been shown to work well by works like Converge [32] and XLINK [99]. We intend to explore appropriate feedback metrics and mechanisms.

Energy aware scheduling over the other device: Utilizing multiple devices for collaboration comes with the cost of additional energy consumption. However, as validated by [100], the energy vs. performance improvement tradeoff amortizes the additional overheads. Compact can address such overheads by incorporating an appropriate utility function that puts a constraint on the amount of data and energy drawn from the helper device. We intend to explore designing such a power-aware scheduler in the future.

8 Conclusions

In this paper, we propose a system Compact for live video streaming intended for online classes over cellular networks that utilize multiple devices at the user end to distribute video tiles. We observe that the conventional packet-level scheduler incurs inter-path HoL blocking. Therefore, Compact utilizes a video content-aware scheduler that splits a video into foreground and background using tiles and sends these tiles over different network paths. We evaluate Compact under various network conditions using simulated, controlled, and real setups. On a bus trace Compact manages to limit the median stall by 3.4× and the tail stall by 6.2× compared to the single path. It also reduces the median E2E lag by 28.57% and tail lag by ≈ 80%. Compact, therefore, is likely to increase access to education in connectivity-challenged regions.

9 Acknowlegement

Shubham Chaudhary's research in this paper is funded by iHub Anubhuti-IIITD Foundation, New Delhi, India, through the CHANAKYA Doctoral Fellowship. This work is supported by the Science and Engineering Research Board, Department of Science and Technology, Government of India, under Grant CRG/2022/005096.

References

[1] 2020. 50 Online Education Statistics: 2024 Data on Higher Learning & Corporate Training.

[2] 2024. FFmpeg. https://www.ffmpeg.org/.

[3] 2024. Multiple device ownership means more smartphone usage. https://cdne.kantar.com/north-america/inspiration/technology/multiple-device-ownership-means-more-smartphone-usage. Accessed on April 8, 2024.

[4] 2024. Netflix/vmaf. original-date: 2016-02-08T18:41:38Z.

[5] 2024. Teens and Internet, Device Access Fact Sheet. https://www.pewresearch.org/internet/fact-sheet/teens-and-internet-device-access-fact-sheet/. Accessed on April 8, 2024.

[6] 2024-02-14. socat(1): Multipurpose relay - Linux man page. https://linux.die.net/man/1/socat.

[7] 2024-04-01. javacv/samples/DeepLearningFaceDetection.java at master · bytedeco/javacv. https://github.com/bytedeco/javacv/blob/master/samples.

[8] 2024-04-04. Live streaming latency - YouTube Help. https://support.google.com/youtube/answer/7444635. Accessed on April 4, 2024.

[9] 2024-04-08. Understanding and Choosing The Right Frame Rates for You. https://riverside.fm/blog/frame-rate-guide, https://riverside.fm/blog/frame-rate-guide. Accessed on April 8, 2024.

[10] 2024-04-09. Worldwide Broadband Speed League 2023. https://www.cable.co.uk/broadband/speed/worldwide-speed-league/.

[11] 2024-04-12. #2 Machine Learning Specialization [Course 1, Week 1, Lesson 1] - YouTube. https://www.youtube.com/watch?v=wiNXzydta4c.

[12] 2024-04-12. #3 Machine Learning Specialization [Course 1, Week 1, Lesson 2] - YouTube. https://www.youtube.com/watch?v=XtlwSmJfUs4.

[13] 2024-04-12. RL 3: Upper confidence bound (UCB) to solve multi-armed bandit problem. https://www.youtube.com/watch?v=RPbtzWgzD9M.

[14] 2024-11-14. 4G/LTE Bandwidth Logs. https://users.ugent.be/ jvdrhoof/dataset-4g/.

[15] 2024-11-14. The Evolving Landscape of Students' Mobile Learning Practices in Higher Education.

[16] 2024-11-14. Online Learning Statistics: The Ultimate List in 2024 | Devlin Peck.

[17] Irshad Abibouraguimane, Kakeru Hagihara, Keita Higuchi, Yuta Itoh, Yoichi Sato, Tetsu Hayashida, and Maki Sugimoto. 2019. CoSummary: adaptive fast-forwarding for surgical videos by detecting collaborative scenes using hand regions and gaze positions. In *Proceedings of the 24th International Conference on Intelligent User Interfaces* (2024-02-22). ACM, Marina del Ray California, 580–590. https://doi.org/10.1145/3301275.3302284

[18] 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* (2024-03-29). IEEE, Vancouver, BC, Canada, 1–10. https://doi.org/10.1109/INFOCOM42981.2021.9488763

[19] Zahaib Akhtar, Yun Seong Nam, Ramesh Govindan, Sanjay Rao, Jessica Chen, Ethan Katz-Bassett, Bruno Ribeiro, Jibin Zhan, and Hui Zhang. 2018. Oboe: auto-tuning video ABR algorithms to network conditions. In *Proceedings of the 2018 Conference of the ACM Special Interest Group on Data Communication* (2023-12-28). ACM, Budapest Hungary, 44–58. https://doi.org/10.1145/3230543.3230558

[20] Omer F. Aladag, Deniz Ugur, Mehmet N. Akcay, and Ali C. Begen. 2021. Content-Aware Playback Speed Control for Low-Latency Live Streaming of Sports. In *Proceedings of the 12th ACM Multimedia Systems Conference* (2024-02-22). ACM, Istanbul Turkey, 344–349. https://doi.org/10.1145/3458305.3478437

[21] Kisten Schmitt Alexus Bazen. 2023. Cell phone statistics 2024. https://www.consumeraffairs.com/cell_phones/cell-phone-statistics.html. Published on 28 September 2023; Accessed on April 8, 2024.

[22] Research Article, Arbaz Khan, Mubashir Saeed, Muhammad Anwar, and Lareb Kanwal. 2023. Journal of Mass Communication & Journalism Unleashing the Potential: A Study of the Effectiveness and Impact of YouTube Educational Content on Student Learning Outcomes. *Journal of Mass Communication and Journalism* 13:05, 2023 (September 2023). https://doi.org/10.37421/2165-7912.2023.13.538

[23] Netflix Technology Blog. 2017. Toward A Practical Perceptual Video Quality Metric.

[24] Benjamin Bross, Ye-Kui Wang, Yan Ye, Shan Liu, Jianle Chen, Gary J. Sullivan, and Jens-Rainer Ohm. 2021. Overview of the Versatile Video Coding (VVC) Standard and its Applications. *IEEE Transactions on Circuits and Systems for Video Technology* 31, 10 (October 2021), 3736–3764. https://doi.org/10.1109/TCSVT.2021.3101953

[25] Shubham Chaudhary, Aryan Taneja, Anjali Singh, Purbasha Roy, Sohum Sikdar, Mukulika Maity, and Arani Bhattacharya. 2024. TileClipper: Lightweight Selection of Regions of Interest from Videos for Traffic Surveillance. In *2024 USENIX Annual Technical Conference (USENIX ATC 24)*. 967–984.

[26] Di (Laura) Chen, Dustin Freeman, and Ravin Balakrishnan. 2019. Integrating Multimedia Tools to Enrich Interactions in Live Streaming for Language Learning. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems* (2024-11-15) *(CHI '19)*. Association for Computing Machinery, New York, NY, USA, 1–14. https://doi.org/10.1145/3290605.3300668

[27] Xinyue Chen, Si Chen, Xu Wang, and Yun Huang. 2021. "I was afraid, but now I enjoy being a streamer!": Understanding the Challenges and Prospects of Using Live Streaming for Online Education. *Proc. ACM Hum.-Comput. Interact.* 4, CSCW3 (January 2021), 237:1–237:32. https://doi.org/10.1145/3432936

[28] Kai-Yin Cheng, Sheng-Jie Luo, Bing-Yu Chen, and Hao-Hua Chu. 2009. SmartPlayer: user-centric video fast-forwarding. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (2024-02-22). ACM, Boston MA USA, 789–798. https://doi.org/10.1145/1518701.1518823

[29] Quentin De Coninck. 2022. The packet number space debate in multipath QUIC. *ACM SIGCOMM Computer Communication Review* 52, 3 (July 2022), 2–9. https://doi.org/10.1145/3561954.3561956

[30] 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* (2024-03-29). ACM, Incheon Republic of Korea, 160–166. https://doi.org/10.1145/3143361.3143370

[31] 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* (2022-11-16). ACM, Incheon Republic of Korea, 160–166. https://doi.org/10.1145/3143361.3143370

[32] Sandesh Dhawaskar Sathyanarayana, Kyunghan Lee, Dirk Grunwald, and Sangtae Ha. 2023. Converge: QoE-driven Multipath Video Conferencing over WebRTC. In *Proceedings of the ACM SIGCOMM 2023 Conference* (2023-09-08). ACM, New York NY USA, 637–653. https://doi.org/10.1145/3603269.3604822

[33] Jinlong E, Lin He, Zongyi Zhao, Yachen Wang, Gonglong Chen, and Wei Chen. 2024. AggDeliv: Aggregating Multiple Wireless Links for Efficient Mobile Live Video Delivery. In *IEEE INFOCOM 2024 - IEEE Conference on Computer Communications* (2024-09-25). 1173–1180. https://doi.org/10.1109/INFOCOM52122.2024.10621184

[34] Travis Faas, Lynn Dombrowski, Alyson Young, and Andrew D. Miller. 2018. Watch Me Code: Programming Mentorship Communities on Twitch.tv. *Proceedings of the ACM on Human-Computer Interaction* 2, CSCW (November 2018), 1–18. https://doi.org/10.1145/3274319

[35] Keshav Gambhir, Tanmay Rajore, Shubham Chaudhary, Taral Jain, Avishi Gupta, Mukulika Maity, and Arani Bhattacharya. 2023. NATIVE: Network Aggregation based Tiled Live Video Streaming. In *2023 15th International Conference on COMmunication Systems & NETworkS (COMSNETS)*. 219–221. https://doi.org/10.1109/COMSNETS56262.2023.10041371

[36] Hongpeng Guo, Shuochao Yao, Zhe Yang, Qian Zhou, and Klara Nahrstedt. 2021. CrossRoI: cross-camera region of interest optimization for efficient real time video analytics at scale. In *Proceedings of the 12th ACM Multimedia Systems Conference*. 186–199. https://doi.org/10.1145/3458305.3463381

[37] Yihua Ethan Guo, Ashkan Nikravesh, Z. Morley Mao, Feng Qian, and Subhabrata Sen. 2017. Accelerating Multipath Transport Through Balanced Subflow Completion. In *Proceedings of the 23rd Annual International Conference on Mobile Computing and Networking* (2024-04-03). ACM, Snowbird Utah USA, 141–153. https://doi.org/10.1145/3117811.3117829

[38] 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* (2024-04-10). ACM, Irvine California USA, 129–143. https://doi.org/10.1145/2999572.2999606

[39] Jingning Han, Bohan Li, Debargha Mukherjee, Ching-Han Chiang, Adrian Grange, Cheng Chen, Hui Su, Sarah Parker, Sai Deng, Urvang Joshi, Yue Chen, Yunqing Wang, Paul Wilkins, Yaowu Xu, and James Bankoski. 2021. A Technical Overview of AV1. *Proc. IEEE* 109, 9 (September 2021), 1435–1462. https://doi.org/10.1109/JPROC.2021.3058584

[40] 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* (2024-03-29). ACM, Amsterdam Netherlands, 86–100. https://doi.org/10.1145/3544216.3544217

[41] Jia He, Mostafa Ammar, and Ellen Zegura. 2023. A Measurement-Derived Functional Model for the Interaction Between Congestion Control and QoE in Video Conferencing. In *Passive and Active Measurement*, Anna Brunstrom, Marcel Flores, and Marco Fiore (Eds.). Springer Nature Switzerland, Cham, 129–159. https://doi.org/10.1007/978-3-031-28486-1_7

[42] Jeroen Van Der Hooft, Maria Torres Vega, Stefano Petrangeli, Tim Wauters, and Filip De Turck. 2019. Tile-based Adaptive Streaming for Virtual Reality Video. *ACM Transactions on Multimedia Computing, Communications, and Applications* 15, 4 (November 2019), 1–24. https://doi.org/10.1145/3362101

[43] IIT Kharagpur July 2018. 2018. Prof Soumya Kanti Ghosh & Prof Sandip Chakraborty. https://www.youtube.com/watch?v=O-rkQNKqls.

[44] J.R. Iyengar, P.D. Amer, and R. Stewart. 2006. Concurrent Multipath Transfer Using SCTP Multihoming Over Independent End-to-End Paths. *IEEE/ACM Transactions on Networking* 14, 5 (October 2006), 951–964. https://doi.org/10.1109/TNET.2006.882843

[45] Junchen Jiang, Vyas Sekar, and Hui Zhang. 2012. Improving fairness, efficiency, and stability in HTTP-based adaptive video streaming with FESTIVE. In *Proceedings of the 8th international conference on Emerging networking experiments and technologies* (2023-12-28). ACM, Nice France, 97–108. https://doi.org/10.1145/2413176.2413189

[46] Shuowei Jin, Ruiyang Zhu, Ahmad Hassan, Xiao Zhu, Xumiao Zhang, Z. Morley Mao, Feng Qian, and Zhi-Li Zhang. 2024. OASIS: Collaborative Neural-Enhanced Mobile Video Streaming. In *Proceedings of the 15th ACM Multimedia Systems Conference* (Bari, Italy) *(MMSys '24)*. Association for Computing Machinery, New York, NY, USA, 45–55. https://doi.org/10.1145/3625468.3647610

[47] Baptiste Jonglez, Martin Heusse, and Bruno Gaujal. 2020. SRPT-ECF: challenging Round-Robin for stream-aware multipath scheduling. In *2020 IFIP Networking Conference (Networking)*. 719–724.

[48] Chen-Tai Kao, Yen-Ting Liu, and Alexander Hsu. 2014. Speeda: adaptive speed-up for lecture videos. In *Proceedings of the adjunct publication of the 27th annual ACM symposium on User interface software and technology - UIST'14 Adjunct* (2024-02-22). ACM Press, Honolulu, Hawaii, USA, 97–98. https://doi.org/10.1145/2658779.2658794

[49] Lorenzo Keller, Anh Le, Blerim Cici, Hulya Seferoglu, Christina Fragouli, and Athina Markopoulou. 2012. MicroCast: cooperative video streaming on smartphones. In *Proceedings of the 10th international conference on Mobile systems, applications, and services* (2024-04-04). ACM, Low Wood Bay Lake District UK, 57–70. https://doi.org/10.1145/2307636.2307643

[50] René F Kizilcec, Kathryn Papadopoulos, and Lalida Sritanyaratana. 2014. Showing face in video instruction: effects on information retention, visual attention, and affect. In *Proceedings of the SIGCHI conference on human factors in computing systems*. 2095–2102. https://doi.org/10.1145/2556288.2557207

[51] Adam Langley, Alistair Riddoch, Alyssa Wilk, Antonio Vicente, Charles Krasic, Dan Zhang, Fan Yang, Fedor Kouranov, Ian Swett, Janardhan Iyengar, Jeff Bailey, Jeremy Dorfman, Jim Roskind, Joanna Kulik, Patrik Westin, Raman Tenneti, Robbie Shade, Ryan Hamilton, Victor Vasiliev, Wan-Teh Chang, and Zhongyi Shi. 2017. The QUIC Transport Protocol: Design and Internet-Scale Deployment. In *Proceedings of the Conference of the ACM Special Interest Group on Data Communication* (Los Angeles, CA, USA) *(SIGCOMM '17)*. Association for Computing Machinery, New York, NY, USA, 183–196. https://doi.org/10.1145/3098822.3098842

[52] Jean Le Feuvre, Cyril Concolato, and Jean-Claude Moissinac. 2007. GPAC: open source multimedia framework. In *Proceedings of the 15th ACM international conference on Multimedia*. 1009–1012. https://doi.org/10.1145/1291233.1291452

[53] Matthias A. Lee. 2022. Python Tesseract. https://github.com/madmaze/pytesseract. original-date: 2010-10-27T23:02:49Z.

[54] Creative Commons License. Dec 27, 2021. What is Machine Learning Andrew Ng. https://www.youtube.com/watch?v=yeII6KWenWU.

[55] 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. 2024. An Empirical Study of 5G: Effect of Edge on Transport Protocol and Application Performance. *IEEE Transactions on Mobile Computing* 23, 4 (April 2024), 3172–3186. https://doi.org/10.1109/TMC.2023.3274708

[56] Gerui Lv, Qinghua Wu, Yanmei Liu, Zhenyu Li, Qingyue Tan, Furong Yang, Wentao Chen, Yunfei Ma, Hongyu Guo, Ying Chen, and Gaogang Xie. 2024. Chorus: Coordinating Mobile Multipath Scheduling and Adaptive Video Streaming. In *Proceedings of the 30th Annual International Conference on Mobile Computing and Networking* (Washington D.C., DC, USA) *(ACM MobiCom '24)*. Association for Computing Machinery, New York, NY, USA, 246–262. https://doi.org/10.1145/3636534.3649359

[57] 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* (2024-03-29). ACM, Virtual Event, 229–244. https://doi.org/10.1145/3487552.3487842

[58] 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* (2023-12-27). ACM, Los Angeles CA USA, 197–210. https://doi.org/10.1145/3098822.3098843

[59] Preethi Natarajan, Professor Paul D. Amer, Jonathan Leighton, and Fred Baker. 2009. *Using SCTP as a Transport Layer Protocol for HTTP*. Internet Draft draft-natarajan-http-over-sctp-02. Internet Engineering Task Force. Num Pages: 13.

[60] Ravi Netravali, Anirudh Sivaraman, Somak Das, Ameesh Goyal, Keith Winstein, James Mickens, and Hari Balakrishnan. 2015. Mahimahi: Accurate Record-and-Replay for HTTP. In *2015 USENIX Annual Technical Conference (USENIX ATC 15)*. Santa Clara, CA, 417–429. https://www.usenix.org/conference/atc15/technical-session/presentation/netravali

[61] Yunzhe Ni, Feng Qian, Taide Liu, Yihua Cheng, Zhiyao Ma, Jing Wang, Zhongfeng Wang, Gang Huang, Xuanzhe Liu, and Chenren Xu. 2023. {POLYCORN}: Data-driven Cross-layer Multipath Networking for High-speed Railway through Composable Schedulerlets. In *20th USENIX Symposium on Networked Systems Design and Implementation (NSDI 23)*. 1325–1340.

[62] Sohee Park, Arani Bhattacharya, Zhibo Yang, Samir R. Das, and Dimitris Samaras. 2021. Mosaic: Advancing User Quality of Experience in 360-Degree Video Streaming With Machine Learning. *IEEE Transactions on Network and Service Management* 18, 1 (2021), 1000–1015. https://doi.org/10.1109/TNSM.2021.3053183

[63] Sohee Park, Arani Bhattacharya, Zhibo Yang, Samir R. Das, and Dimitris Samaras. 2021. *Mosaic* : Advancing User Quality of Experience in 360-Degree Video Streaming With Machine Learning. *IEEE Transactions on Network and Service Management* 18, 1 (March 2021), 1000–1015. https://doi.org/10.1109/TNSM.2021.3053183

[64] Sohee Park, Minh Hoai, Arani Bhattacharya, and Samir R. Das. 2021. Adaptive Streaming of 360-Degree Videos With Reinforcement Learning. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*. 1839–1848. https://doi.org/10.1109/WACV48630.2021.00188

[65] Genevieve Carlton Ph.D. 2024. 2024 Online Learning Statistics. Section: Online Colleges.

[66] Costin Raiciu, Mark Handley, and Damon Wischik. 2011. *Coupled congestion control for multipath transport protocols*. Technical Report.

[67] Costin Raiciu, Christoph Paasch, Sebastien Barre, Alan Ford, Michio Honda, Fabien Duchene, Olivier Bonaventure, and Mark Handley. [n. d.]. How Hard Can It Be? Designing and Implementing a Deployable Multipath TCP. ([n. d.]).

[68] 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* (2023-12-27). ACM, Los Cabos Mexico, 1–16. https://doi.org/10.1145/3300061.3345435

[69] Pranit Sarda. 2020. Why YouTube still rules the edtech roost. https://www.forbesindia.com/article/edtech-special/why-youtube-still-rules-the-edtech-roost/57761/1. Published on Feb 18, 2020; Accessed on Apr 12, 2024.

[70] Katherine Schaeffer. 2019. Most U.S. teens who use cellphones do it to pass time, connect with others, learn new things. https://www.pewresearch.org/short-reads/2019/08/23/most-u-s-teens-who-use-cellphones-do-it-to-pass-time-connect-with-others-learn-new-things/. Accessed on April 8, 2024.

[71] Satadal Sengupta, Niloy Ganguly, Sandip Chakraborty, and Pradipta De. 2018. HotDASH: Hotspot Aware Adaptive Video Streaming Using Deep Reinforcement Learning. In *2018 IEEE 26th International Conference on Network Protocols (ICNP)* (2024-04-12). IEEE, Cambridge, 165–175. https://doi.org/10.1109/ICNP.2018.00026

[72] Varun Singh, Saba Ahsan, and Jörg Ott. 2013. MPRTP: multipath considerations for real-time media. In *Proceedings of the 4th ACM Multimedia Systems Conference* (2024-03-29). ACM, Oslo Norway, 190–201. https://doi.org/10.1145/2483977.2484002

[73] Vibhaalakshmi Sivaraman, Pantea Karimi, Vedantha Venkatapathy, Mehrdad Khani, Sadjad Fouladi, Mohammad Alizadeh, Frédo Durand, and Vivienne Sze. 2023. Gemino: Practical and Robust Neural Compression for Video Conferencing. arXiv:2209.10507 [cs].

[74] Jangwoo Son, Dongmin Jang, and Eun-Seok Ryu. 2018. Implementing Motion-Constrained Tile and Viewport Extraction for VR Streaming. In *Proceedings of the 28th ACM SIGMM Workshop on Network and Operating Systems Support for Digital Audio and Video* (2024-03-07). ACM, Amsterdam Netherlands, 61–66. https://doi.org/10.1145/3210445.3210455

[75] Randall R. Stewart. 2007. *Stream Control Transmission Protocol*. Request for Comments RFC 4960. Internet Engineering Task Force. https://doi.org/10.17487/RFC4960 Num Pages: 152.

[76] Gary J Sullivan, Jens-Rainer Ohm, Woo-Jin Han, and Thomas Wiegand. 2012. Overview of the high efficiency video coding (HEVC) standard. *IEEE Transactions on circuits and systems for video technology* 22, 12 (2012), 1649–1668. https://doi.org/10.1109/TCSVT.2012.2221191

[77] Petroc Taylor. Jan 19, 2023. Average number devices and connections per person worldwide in 2018 and 2023. https://www.statista.com/statistics/1190270/number-of-devices-and-connections-per-person-worldwide/. Accessed on Nov 22, 2024.

[78] Margot Van Wermeskerken, Susanna Ravensbergen, and Tamara Van Gog. 2018. Effects of instructor presence in video modeling examples on attention and learning. *Computers in Human Behavior* 89 (2018), 430–438. https://doi.org/10.1016/j.chb.2017.11.038

[79] Matteo Varvello, Hyunseok Chang, and Yasir Zaki. 2022. Performance characterization of videoconferencing in the wild. In *Proceedings of the 22nd ACM Internet Measurement Conference* (2023-05-26). ACM, Nice France, 261–273. https://doi.org/10.1145/3517745.3561442

[80] Marko Viitanen, Ari Koivula, Ari Lemmetti, Arttu Ylä-Outinen, Jarno Vanne, and Timo D Hämäläinen. 2016. Kvazaar: open-source HEVC/H. 265 encoder. In *Proceedings of the 24th ACM international conference on Multimedia*. 1179–1182. https://doi.org/10.1145/2964284.2973796

[81] T. Daniel Wallace and Abdallah Shami. 2014. Concurrent Multipath Transfer Using SCTP: Modelling and Congestion Window Management. *IEEE Transactions on Mobile Computing* 13, 11 (November 2014), 2510–2523. https://doi.org/10.1109/TMC.2014.2307330

[82] Chengke Wang, Hao Wang, Feng Qian, Kai Zheng, Chenglu Wang, Fangzhu Mao, Xingmin Guo, and Chenren Xu. 2023. Experience: A Three-Year Retrospective of Large-scale Multipath Transport Deployment for Mobile Applications. In *Proceedings of the 29th Annual International Conference on Mobile Computing and Networking* (2023-07-15). ACM, Madrid Spain, 1–15. https://doi.org/10.1145/3570361.3592506

[83] Haiping Wang, Zhenhua Yu, Ruixiao Zhang, Siping Tao, Hebin Yu, and Shu Shi. 2023. TwinStar: A Practical Multi-path Transmission Framework for Ultra-Low Latency Video Delivery. In *Proceedings of the 31st ACM International Conference on Multimedia* (2024-03-18). ACM, Ottawa ON Canada, 9234–9242. https://doi.org/10.1145/3581783.3613443

[84] Jiahui Wang, Pavlo Antonenko, and Kara Dawson. 2020. Does visual attention to the instructor in online video affect learning and learner perceptions? An eye-tracking analysis. *Computers & Education* 146 (2020), 103779. https://doi.org/10.1016/j.compedu.2019.103779

[85] Damon Wischik, Costin Raiciu, Adam Greenhalgh, and Mark Handley. 2011. Design, Implementation and Evaluation of Congestion Control for Multipath TCP. (March 2011). https://www.usenix.org/conference/nsdi11/design-implementation-and-evaluation-congestion-control-multipath-tcp

[86] Jiyan Wu, Bo Cheng, and Ming Wang. 2018. Improving Multipath Video Transmission With Raptor Codes in Heterogeneous Wireless Networks. *IEEE Transactions on Multimedia* 20, 2 (February 2018), 457–472. https://doi.org/10.1109/TMM.2017.2741425

[87] Jiyan Wu, Bo Cheng, Chau Yuen, Yanlei Shang, and Junliang Chen. 2015. Distortion-Aware Concurrent Multipath Transfer for Mobile Video Streaming in Heterogeneous Wireless Networks. *IEEE Transactions on Mobile Computing* 14, 4 (April 2015), 688–701. https://doi.org/10.1109/TMC.2014.2334592

[88] Jiangkai Wu, Yu Guan, Qi Mao, Yong Cui, Zongming Guo, and Xinggong Zhang. 2023. ZGaming: Zero-Latency 3D Cloud Gaming by Image Prediction. In *Proceedings of the ACM SIGCOMM 2023 Conference* (New York, NY, USA) *(ACM SIGCOMM '23)*. Association for Computing Machinery, New York, NY, USA, 710–723. https://doi.org/10.1145/3603269.3604819

[89] Jiyan Wu, Chau Yuen, Ming Wang, and Junliang Chen. 2016. Content-Aware Concurrent Multipath Transfer for High-Definition Video Streaming over Heterogeneous Wireless Networks. *IEEE Transactions on Parallel and Distributed Systems* 27, 3 (March 2016), 710–723. https://doi.org/10.1109/TPDS.2015.2416736

[90] Man Wu and Qin Gao. 2020. Using live video streaming in online tutoring: Exploring factors affecting social interaction. *International Journal of Human-Computer Interaction* 36, 10 (2020), 964–977. https://doi.org/10.1080/10447318.2019.1706288

[91] Changqiao Xu, Zhuofeng Li, Jinglin Li, Hongke Zhang, and Gabriel-Miro Muntean. 2015. Cross-Layer Fairness-Driven Concurrent Multipath Video Delivery Over Heterogeneous Wireless Networks. *IEEE Transactions on Circuits and Systems for Video Technology* 25, 7 (July 2015), 1175–1189. https://doi.org/10.1109/TCSVT.2014.2376138

[92] Changqiao Xu, Tianjiao Liu, Jianfeng Guan, Hongke Zhang, and Gabriel-Miro Muntean. 2013. CMT-QA: Quality-Aware Adaptive Concurrent Multipath Data Transfer in Heterogeneous Wireless Networks. *IEEE Transactions on Mobile Computing* 12, 11 (November 2013), 2193–2205. https://doi.org/10.1109/TMC.2012.189

[93] 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* (2024-03-29). ACM, Virtual Event USA, 479–494. https://doi.org/10.1145/3387514.3405882

[94] Praveen Kumar Yadav and Wei Tsang Ooi. 2020. Tile Rate Allocation for 360-Degree Tiled Adaptive Video Streaming. In *28th ACM International Conference on Multimedia* (Seattle, WA, USA). 3724–3733. https://doi.org/10.1145/3394171.3413550

[95] Alex Yelenevych. [n. d.]. The Future Of EdTech. https://www.forbes.com/sites/forbesbusinesscouncil/2022/12/26/the-future-of-edtech/?sh=4ed280296c2f. Published on Dec 26, 2022; Accessed on Apr 12, 2024.

[96] Xiaoqi Yin, Abhishek Jindal, Vyas Sekar, and Bruno Sinopoli. 2015. A Control-Theoretic Approach for Dynamic Adaptive Video Streaming over HTTP. In *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication* (2023-12-28). ACM, London United Kingdom, 325–338. https://doi.org/10.1145/2785956.2787486

[97] 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* (2024-03-29). ACM, Amsterdam Netherlands, 101–113. https://doi.org/10.1145/3544216.3544219

[98] 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* (2024-03-29). IEEE, Trondheim, Norway, 1–9. https://doi.org/10.1109/IFIPNetworking.2014.6857103

[99] 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* (2022-09-30). ACM, Virtual Event USA, 418–432. https://doi.org/10.1145/3452296.3472893

[100] Xiao Zhu, Jiachen Sun, Xumiao Zhang, Y Ethan Guo, Feng Qian, and Z Morley Mao. 2020. MPBond: efficient network-level collaboration among personal mobile devices. In *Proceedings of the 18th International Conference on Mobile Systems, Applications, and Services*. 364–376. https://doi.org/10.1145/3386901.3396600

[101] Xutong Zuo, Yong Cui, Xin Wang, and Jiayu Yang. 2022. Deadline-aware Multipath Transmission for Streaming Blocks. In *IEEE INFOCOM 2022 - IEEE Conference on Computer Communications*. 2178–2187. https://doi.org/10.1109/INFOCOM48880.2022.9796942