Implementation of Live Video Streaming using Multipath over QUIC

By

Aruba Sood
(MT23022)

Under the supervision of

Dr. Arani Bhattacharya

Dr. Mukulika Maity

Indraprastha Institute of Information Technology Delhi

May , 2025

Implementation of Live Video Streaming using Multipath over QUIC

By

Aruba Sood
(MT23022)

Submitted in partial fulfillment of the requirements for the degree of

Master of Technology

To

Indraprastha Institute of Information Technology Delhi

May , 2025

Certificate

This is to certify that the thesis titled *Implementation of Live Video Streaming using Multipath over QUIC* being submitted by *Aruba Sood* to the Indraprastha Institute of Information Technology Delhi, for the award of the degree of Master of Technology, is an original research work carried out by him under my supervision. In my opinion, the thesis has reached the standards fulfilling the requirements of the regulations relating to the degree.

The results contained in this thesis have not been submitted in part or full to any other university or institute for the award of any degree/diploma.

Arani Bhattacharya

M. Maity.

May , 2025

Dr. Arani Bhattacharya

Dr. Mukulika Maity

Indraprastha Institute of Information Technology Delhi

New Delhi 110020

It's not the mountain we conquer, but ourselves.

Sir Edmund Hillary

The world will ask you who you are, and if you don't know, the world will tell you.

Carl Jung

Believe you can and you're halfway there.

Theodore Roosevelt

Acknowledgements

I would like to express my heartfelt gratitude to my advisors, Dr. Arani Bhattacharya and Dr. Mukulika Maity, for their invaluable guidance, encouragement, and support throughout this thesis. Their insights and mentorship have played a crucial role in shaping my research journey.

I am also deeply thankful to Shubham Chaudhary, for his constant help, technical suggestions, and collaborative spirit during the course of this work.

Finally, I would like to thank my friends Surendra Bishnoi, Navneet Mishra, Purbasha Barik, Mehakdeep Kaur and Prerna and others, as well as my family and Youtee for their unwavering support, patience, and encouragement, without which this endeavor would not have been possible.

Above all, I am deeply grateful to God for their blessings and guidance throughout this journey.

Abstract

The increasing demand for live online classes, especially in remote and underserved areas, underscores the importance of providing a seamless and high-quality experience to support effective learning. These real-time, bandwidth-intensive applications pose significant challenges for current cellular networks in terms of maintaining consistent bandwidth, low latency, and minimal stalls. A system COMPACT tackled these challenges by using a content-aware streaming strategy while leveraging multiple devices, each with its own cellular connection, to cooperatively stream video. COMPACT splits the video into foreground and background regions using independently encoded tiles and streams them over different paths based on network estimates.

The original implementation of COMPACT used SCTP (Stream Control Transmission Protocol) for multipath support. The key disadvantage of this version was that SCTP is often blocked by firewalls and is not supported by Android. This thesis, therefore, extends the original implementation of COMPACT to utilize QUIC, a modern transport protocol offering built-in multiplexing, reduced latency, and improved congestion control. We adapt COMPACT's scheduling and streaming logic to work efficiently over QUIC and evaluate the system using realistic network traces. Our results demonstrate that COMPACT over QUIC retains the benefits of multipath collaboration while offering better compatibility with today's internet infrastructure.

Contents

List of Figures

Chapter 1

Introduction

In recent years, the demand for online learning has surged dramatically [1–3], driven by improvements in internet access and the proliferation of handheld devices like smartphones and tablets[2, 3]. Students across the globe, including those in remote areas, are now able to attend live online classes [1, 4–8] from reputed institutions, thanks to mobile learning technologies. Live video streaming, supported by platforms such as YouTube and Twitch, serves as the backbone of this educational transformation. However, these services typically incur latencies ranging from 5 to 10 seconds [9], which significantly degrades the Quality of Experience (QoE)—especially in interactive educational settings where synchrony between audio, video, and text is crucial for maintaining the sense of a live classroom [4, 5].

Despite the emergence of high-bandwidth wireless technologies like 5G, 6G, and WiFi-7 [10–13], existing wireless infrastructures still struggle to provide low-latency and reliable streaming [14–17], especially during peak usage or in areas with network congestion. This necessitates more efficient mechanisms to optimize the delivery of live video content.

One promising solution is the use of multipath streaming, which aggregates bandwidth across multiple network interfaces or devices [18–26]. Students frequently have access to multiple internet-connected devices at home, such as laptops and smartphones. Prior systems like MPBond [27], MicroCast [28], and OASIS [29] have demonstrated the potential of leveraging these devices collaboratively to pool bandwidth. However, these solutions were primarily designed for non-live

streaming use cases and do not address the unique demands of real-time video delivery.

Recent studies (e.g., Converge, TwinStar, AggDeliv [18–20]) have demonstrated the benefits of multipath protocols such as MPTCP [30–32], MPQUIC ([33–35], and MPRTP [36] for video conferencing and streaming. These protocols distribute packets across multiple paths, offering resilience to path failure and better utilization of aggregate bandwidth. Nevertheless, they face a critical limitation: they lack content awareness. Packets are transmitted in a throughput-optimized manner, without considering the semantic importance or spatial relevance of video content. This often leads to Head-of-Line (HoL) blocking (Sec. 3.1) which means that we have to wait for certain P_{i-1} packets even if we have received P_i packets. This might occur due to reordering delays, or low-bitrate encoding of important visual regions to fit within available bandwidth, thereby degrading user QoE. To truly enhance live video streaming, especially for online classes a content-aware multipath transmission strategy is essential—one that allocates bandwidth preferentially to semantically important regions (e.g., a teacher's face, slides, or a blackboard) and leverages helper devices effectively.

A prior work proposes COMPACT [37], a content-aware multipath live streaming system designed specifically for interactive online classes. COMPACT exploits the fact that not all parts of a video are equally important: for example, in a virtual classroom, the instructor's face and gestures (foreground) are far more critical than the static background like slides or the wall. COMPACT uses this insight to separate the video into foreground (FG) and background (BG) using tiles via HEVC encoding. Each content type is then streamed over different network paths based on its importance and urgency — high-priority FG tiles go through the low-latency/high-BW path, while less time-sensitive BG tiles use the secondary path. To achieve this, COMPACT designed a custom scheduler that dynamically adapts based on real-time path characteristics like RTT and bandwidth. This content-aware tile scheduling not only improves stream reliability but also reduces stalls and end-to-end latency significantly.

The initial implementation of COMPACT leveraged the Stream Control Transmission Protocol (SCTP) to enable multipath communication. However, this approach faced significant limitations: SCTP is frequently restricted by firewalls and lacks native support on widely used platforms such as Android. These constraints hindered the deployment and broader adoption of the system.

To address these challenges, this thesis presents an enhanced version of COMPACT that replaces SCTP with QUIC—a modern, UDP-based transport protocol. QUIC offers several key advantages, including native support for multiplexed streams, lower connection establishment latency, and advanced congestion control mechanisms.

In this work, I adapted COMPACT 's core scheduling and streaming architecture to operate efficiently over QUIC, specifically PicoQuic[38], taking full advantage of its transport-layer innovations as well as application level control. I further evaluate the redesigned system using diverse and realistic network traces that simulate heterogeneous connectivity scenarios. I evaluate the new implementation over prior SCTP implementation and also against the baseline MinRTT over QUIC. Our experimental results (Sec 6.2) show that COMPACT over QUIC not only preserves the multipath benefits of the original design but also achieves improved compatibility, deployability, and robustness across modern internet environments, particularly in mobile and NAT-constrained networks.

Chapter 2

Related Work

Schedulers in multipath are responsible for deciding how data is split and transmitted across multiple network paths in multipath-enabled systems. Their primary goal is to improve throughput, reliability, and latency by intelligently distributing packets across available paths. However, the design of such schedulers is challenging due to differences in delay, bandwidth, and loss characteristics of each path. Over the years, both classic and advanced scheduling strategies have been proposed to address these challenges.

2.1 Classic Schedulers

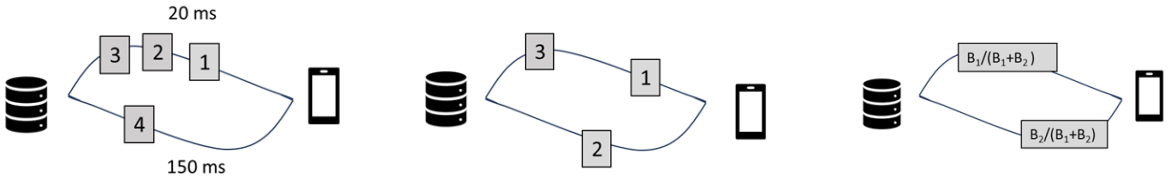
2.1.1 Min-RTT

MinRTT[39] always selects the path with the lowest round-trip time as shown in Fig. 2.1a), assuming that minimal latency ensures faster delivery. However, it can lead to congestion by overloading the fastest path and ignoring available capacity on other paths.

2.1.2 Round Robin

Round Robin distributes packets evenly across all paths without considering the characteristics of the paths as shown in Fig. 2.1b). While it ensures fairness, it often causes packet reordering at

the receiver, increasing buffering and reassembly delays.



(a) Min-RTT (b) Round Robin (c) Musher

Figure 2.1: Classic Schedulers

2.1.3 Musher

Musher scheduler [40] distribute packets on the paths according to each path's bandwidth ratio as shown in Fig. 2.1c). The load on path P_i is bandwidth BW_i of path P_i divided by the aggregate bandwidth of all the paths.

2.2 Advanced Schedulers

2.2.1 XLINK

XLINK [26] is a lightweight, QoE-driven multi-path extension for QUIC designed specifically for large-scale video services. Using real-time feedback on user-perceived video quality (such as frame cache status and frame rate), XLINK dynamically schedules data across multiple wireless interfaces (like Wi-Fi and cellular). It employs fine-grained, priority-based re-injection of packets at both stream and frame levels, effectively overcoming multi-path head-of-line blocking and network heterogeneity issues, thus improving responsiveness and playback smoothness.

2.2.2 CONVERGE

Converge's [18] architecture enhances video conferencing by combining a video-aware scheduler with real-time QoE feedback to dynamically distribute packets across multiple network paths. This adaptive system adjusts transmission rates based on network conditions and video structure,

ensuring optimal video quality and reduced latency. Its path-specific FEC module further improves reliability by balancing error correction with latency considerations, maintaining consistent video quality even in fluctuating network environments.

Supporting multipath transmission, Converge leverages detailed network and video feedback to optimize packet routing and latency, reducing frame drops and delays. This integration enables higher throughput, resilience, and seamless video experiences, outperforming existing solutions by providing uninterrupted, high-quality calls in diverse and dynamic network conditions.

2.2.3 CHORUS

Chorus [41] demonstrates significant improvements in QoE by effectively coordinating multipath scheduling with adaptive bitrate algorithms, making it a promising solution for mobile video streaming. Additionally, while designed for minimal modifications, deploying Chorus across a wide range of real-world systems could introduce complexity and overhead, impacting scalability and latency. Its reliance on stable path conditions and throughput prediction might limit its adaptability in highly dynamic environments.

2.2.4 TWINSTAR

TwinStar [19] is a multipath framework that enhances ultralow latency video streaming by concurrently using multiple network paths to counteract network jitter and packet loss. It employs strategies like independent encoding, dynamic bitrate allocation, and inter-path FEC recovery, making it robust against network fluctuations.

Deployed in a commercial cloud gaming platform, TwinStar significantly reduces stalls by up to 91% and improves video quality, outperforming traditional single-path methods. Its design effectively manages network variability, ensuring smooth, high-quality low-latency video delivery in challenging conditions.

Chapter 3

Background and Motivation

3.1 Drawbacks of prior work

3.1.1 Content Awareness

Prior multipath schedulers—such as MinRTT, Round Robin, and Musher—were primarily network-aware but video QoE-unaware. These strategies rely on metrics like RTT, bandwidth estimation, or static distribution schemes to schedule packets across multiple paths. However, they ignore the semantic structure of video frames, especially the visual importance of different regions.

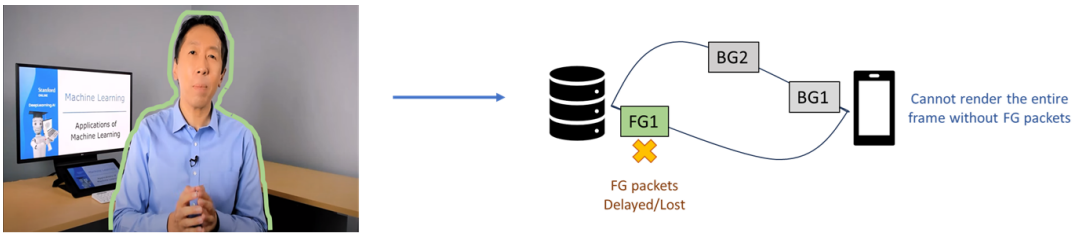


Figure 3.1: Head of Line Blocking w/o Content Awareness

Advanced schedulers like CHORUS, XLINK, CONVERGE and TWINSTAR introduced QoE-awareness by reacting to client feedback or prioritizing key frames. While they improved upon naive packet scheduling, they still lacked fine-grained content awareness, which means these schedulers do not differentiate between visually significant and insignificant parts within a

frame. In a live online class, the teacher or the speaker is the most important part of the video which is called as the Foreground and the slides or the notes, which change less frequently is called as the background. Fig. 3.1 shows that if the FG (Foreground) part is scheduled on the slower path and the BG (Background) is scheduled on the faster path, we might face Head of the Line Blocking for the FG part.

Moreover, some of the above works rely heavily on retransmission or re-injection strategies, which are inefficient for latency-sensitive applications like live classes. Therefore, they may still suffer from stalls, lag, and perceptible jitter, especially in heterogeneous networks where path characteristics (e.g., delay, jitter, loss) vary drastically.

3.1.2 Tile level versus Packet level Scheduling

In **Packet-level** video streaming, a video is encoded into segments and then split into smaller data packets. These packets are distributed across multiple network paths. While this helps utilize bandwidth from multiple links, it can lead to Head-of-Line (HoL) blocking: if even one packet is delayed or lost, the entire video segment can't be decoded and rendered, causing playback stalls.

In contrast, **Tile-level** video streaming divides the video frame spatially into independently encoded rectangular regions called tiles (e.g., foreground and background) as shown in Fig.3.2. Each tile is then streamed as a whole unit over a single path. Since tiles are independently decodable, the video player can render available tiles without waiting for all others, thus reducing latency and avoiding HoL blocking.

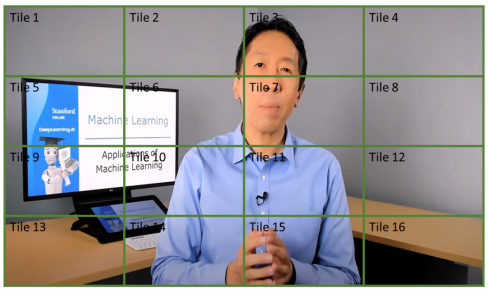


Figure 3.2: Understanding Tiles

Tile-level streaming overcomes the limitations of packet-level multipath streaming by reducing HoL blocking. In packet-level streaming, video segments are split into packets and

spread across multiple paths. A delay or loss in even a single packet can stall playback. Tile-level streaming solves this by sending complete tiles—each comprising a group of packets—through a single path. As each tile is independently decodable, rendering can proceed with whichever tileset arrives first, even over heterogeneous paths. This reduces video stalls and enables adaptive quality for foreground and background tiles.

3.2 COMPACT: Content Aware scheduler

COMPACT [37] (Content-aware Multipath Live Video Streaming for Online Classes using Video Tiles) is an already proposed and accepted work, overcome the drawbacks mentioned above.

COMPACT is a video streaming framework designed for live online classes over multipath networks. It enhances user Quality of Experience (QoE) by exploiting the visual importance of video regions. COMPACT divides each video frame into tiles using HEVC tiling and applies a deep learning-based foreground detector to classify tiles as either Foreground (FG) (e.g., instructor's face) or Background (BG) (e.g., slides or static elements).

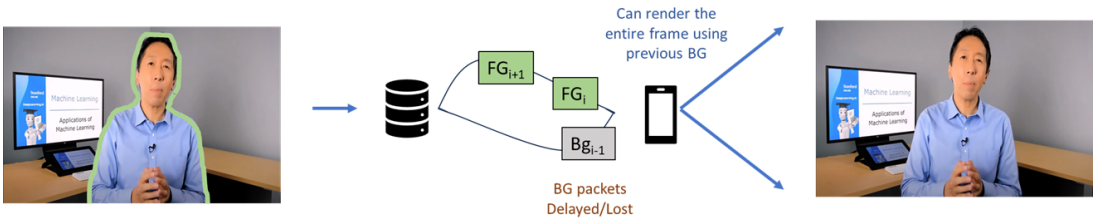


Figure 3.3: COMPACT's Content Awareness

The scheduler then sends FG tiles over the faster, more reliable path, while BG tiles are offloaded to slower paths, leveraging temporal redundancy (e.g., reusing BG from previous frames) as shown in Fig. 3.3. It aims to maximize utility by balancing tile quality, reducing stalls, and minimizing quality switches

3.3 COMPACT over SCTP

The prior work on COMPACT's framework [37] was implemented in JAVA. It runs over SCTP[42–44] (Stream Control Transmission Protocol), as the transport layer protocol. But SCTP has its own drawbacks.

Drawbacks of SCTP

SCTP, although it supports similar multiplexing features, faces challenges in widespread deployment due to its reliance on kernel support, potential middlebox interference, and less integration with existing web protocols.

My thesis work extends COMPACT's framework over a modern transport layer protcol called QUIC.

3.4 An Overview of QUIC

QUIC[45] was motivated by the need to address several fundamental issues with TCP and TLS that hinder web performance and deployment. These include high handshake latency, head-of-line blocking, and difficulties in deploying new protocols due to interference from the middle box and the infancy of the protocol. By designing a protocol from scratch that integrates cryptography, multiplexing, and connection migration, QUIC aims to significantly reduce connection setup times, improve loss recovery, and enhance security, all while enabling rapid deployment and evolution at the application level without requiring changes to middleboxes or network infrastructure.

Why QUIC in place of SCTP?

Consequently, QUIC's[45] architecture, built atop UDP and designed for Internet-scale deployment, provides a more practical and adaptable solution for modern web and application needs. Moreover, QUIC's integration with TLS specifically, TLS 1.3 simplifies security management and leverages widely adopted cryptographic standards, making it easier to deploy and update without requiring modifications to network intermediaries. QUIC's design

emphasizes application-level control, enabling faster iteration and deployment, especially in web environments where TCP and HTTP/2 are prevalent.

Also, QUIC supports all the main features provided by SCTP like in-order delivery of individual streams, stream-level multiplexing without head-of-line (HoL) blocking and configurable retransmissions and flow control.

Chapter 4

Framework

4.1 Design Overview

COMPACT is a video streaming framework optimized for multipath live video delivery as shown in Fig. 4.1. At the streamer side, raw video frames from a camera are analyzed by a deep neural network (DNN)-based Foreground (FG) detector, which identifies tiles containing human faces. These are then separated into FG and Background (BG) tile sets by a scheduler. The scheduler also determines the appropriate video quality and streaming path for each set, choosing the primary path (with the lowest estimated completion time based on RTT and bandwidth) for FG tiles and the secondary path for BG tiles.

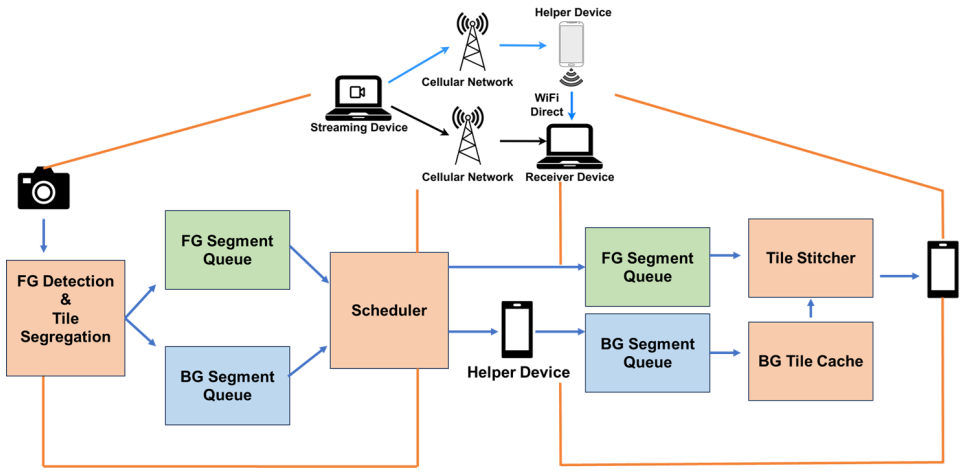


Figure 4.1: COMPACT's framework

Encoded FG and BG segments are stored in separate queues and sent through respective paths. The Streamer module handles the actual transmission and continuously updates network statistics via probe channels that measure RTT and bandwidth using timestamped packets.

On the receiver side (primary device), the segments are decoded and stitched back into full video frames. If BG tiles are delayed, the last successfully rendered BG frame from a cache is reused. This decoupling of FG and BG enables smoother playback even under network variability, especially in scenarios like online classes where the background changes infrequently.

4.2 COMPACT's Scheduler

To ensure high-quality video streaming over heterogeneous network paths, we design a content-aware scheduler that leverages tile-level encoding and classification of video frames into foreground (FG) and background (BG) regions. Once the sets of FG and BG tiles for each video segment are identified, the timely delivery of both sets becomes crucial to minimize playback stalls and enhance the user's Quality of Experience (QoE).

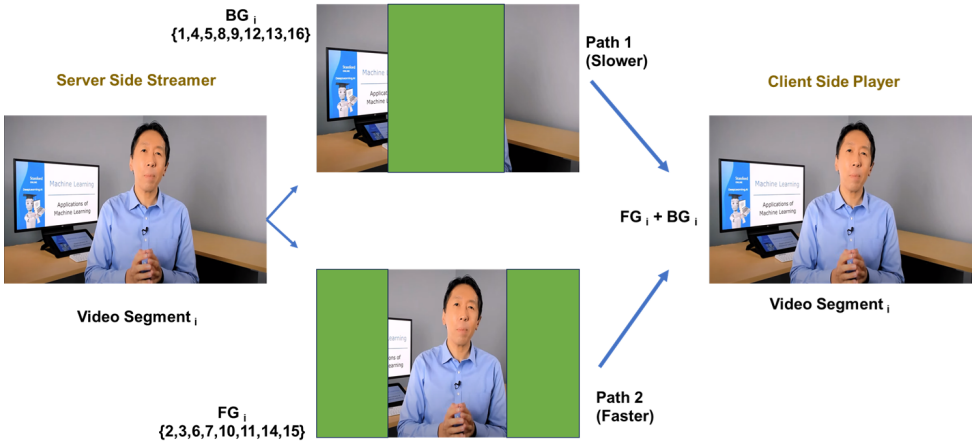


Figure 4.2: Content Aware Scheduler

Once both the FG_i and BG_i tiles are received on the player side, the entire frame is stitched and rendered. Fig. 4.2 shows how COMPACT achieves content- awareness matters while scheduling in a latency sensitive application like live video steaming. A second scenario possible, could be where the FG_i tiles have been received earlier than the BG_i tiles due to the scheduling of FG_i on the faster path as shown in Fig. 4.3. In this case COMPACT does not face any HOL for

the FG tiles because of the solution employed that FG_i can be rendered with previously received BG that is BG_{i-1} .

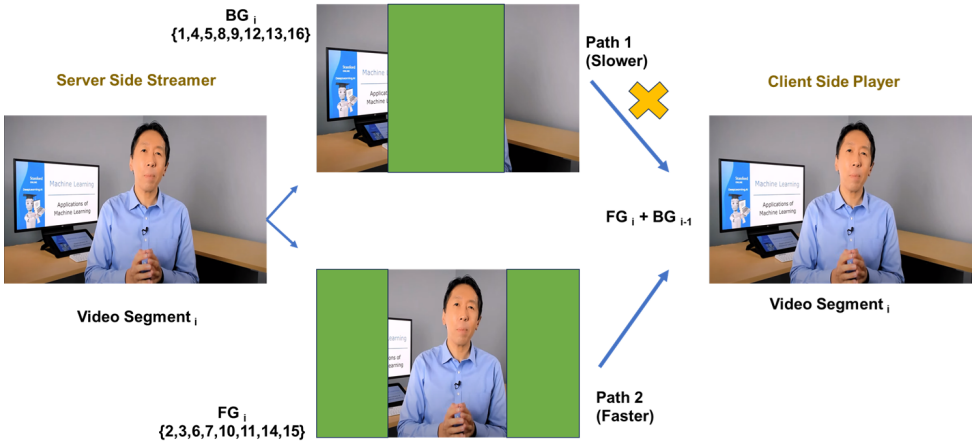


Figure 4.3: No HOL for FG

A third scenario possible could be when the FG_i itself is delayed due to real life network fluctuations. In this case we will have to wait for the FG tiles and it accounts for Stall. For a video encoded at 25 frames per second, the inter-frame display time (δ) is 40 ms. Any additional delay in receiving a segment beyond this time causes a stall or freeze in playback. Therefore, the goal of the scheduler is to maximize perceived video quality while minimizing stalls and quality fluctuations within and across segments.

4.2.1 Utility Function for Scheduling

We define a utility function that captures the trade-off between different QoE factors namely 1) video quality, 2) Intra Segment quality 3) Inter Segment quality and 4) Stall duration as shown in Fig. 4.4.

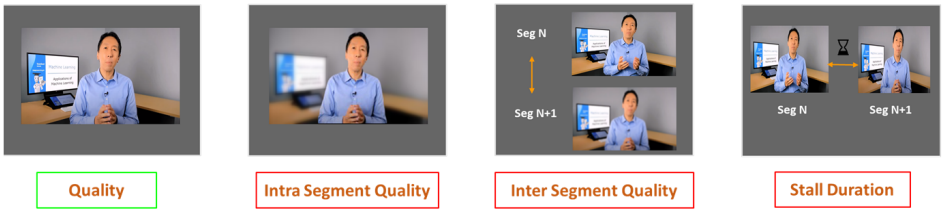


Figure 4.4: QoE parameters for scheduler

The utility is computed as:

$$\text{Maximize Utility}_t = \alpha_1 Q_t - \alpha_2 I_1 - \alpha_3 I_2 - \alpha_4 L_t \tag{4.1}$$

where:

- Q_t is the user-perceived video quality,
- I_1 and I_2 are penalties for inter- and intra-segment quality switches respectively,
- L_t is the stall duration,

The weights are empirically selected based on prior works [**?**, **?**], with $\alpha_1 = \max(Q_t)$ and $\alpha_2 = \alpha_3 = \alpha_4 = 1$.

4.2.2 Video Quality Estimation

The video quality Q_t at segment t is modeled as a weighted sum of the quality of FG and BG tiles:

$$Q_t = \beta_1 Q_{FG} + \beta_2 Q_{BG} \tag{4.2}$$

We set $\beta_1 = 0.6$ and $\beta_2 = 0.4$, giving higher priority to FG tiles. The normalized quality values are derived from the video quantization parameter (QP), with lower QP values indicating higher quality. If BG tiles are delayed beyond a threshold, the player may render the current FG tiles stitched with the BG tiles from the previous segment, penalized by $p(t) = \max(Q_t)$.

4.2.3 Data Transmission and Completion Time

Assume a total of N tiles, with m in FG and n in BG. If o BG tiles are transmitted over the primary (faster) path, the data volumes sent over the primary (F_P) and secondary (F_S) paths are:

$$F_P = R_{FG}(Q_{FG}) \cdot m + R_{BG}(Q_{BG}) \cdot o \tag{4.3}$$

$$F_S = R_{BG}(Q_{BG}) \cdot (n - o) \tag{4.4}$$

where R_{FG} and R_{BG} represent the estimated file size at a given QP.

Completion times over the primary and secondary paths are:

$$T_P = \frac{RTT_P}{2} + \frac{F_P}{BW_P}, \quad T_S = \frac{RTT_S}{2} + \frac{F_S}{BW_S} \tag{4.5}$$

An indicator function $1(K)$ is used to determine whether BG tiles can be rendered within the threshold time. If $T_S - T_P < \delta$, then $1(K) = 1$; otherwise, it is 0.

4.2.4 Quality Switch Penalties

To avoid jarring visual artifacts, we penalize quality switches as follows:

- **Inter-segment switch penalty**: $I_1 = |Q_t - Q_{t-1}|$
- **Intra-segment switch penalty**: $I_2 = |Q_{FG} - Q_{BG}|$

These are calculated using normalized QP-based quality values across tiles.

4.2.5 Stall Duration Estimation

The stall duration L_t is estimated using:

$$K_{max} = \max\left(T_P, 1(K) \cdot T_S\right) \tag{4.6}$$

$$L_t = \max(K_{max} - P_{t-1} - \delta, 0) \tag{4.7}$$

where P_{t-1} is the playout time of the previous segment.

4.2.6 Scheduling Mechanism

The scheduler performs an **exhaustive search** over:

- Possible QP levels for FG and BG,
- Tile allocations across paths.

We limit the runtime of this search to ≤ 1 ms per segment to ensure feasibility for real-time streaming. Filesize estimations during the search are guided by empirical data (e.g., Fig. **??**), assuming a QP reduction of 3 increases bitrate by approximately $1.2\times$.

Chapter 5

Implementation

I implemented both the host (streamer) and viewer (joinee) sides of **COMPACT** in **C/C++** with around 950- 1000 lines of code on both server and receiver side with 450-500 lines of code for PicoQuic. The code for scheduler, FG detection, RTT and BW channel code is written separately with a total of about 1000 lines of code. The setup involves three Linux machines— **Streamer**, **Player** and **Helper** devices. See Fig. 6.1 for more understanding. Below, is the description of the components of the system in detail.

5.1 Video Encoding

COMPACT employs **tiled HEVC encoding** [46] to facilitate adaptive streaming. We use the open-source encoder **Kvazaar** [47] to divide raw frames into a **4×4 grid** of tiles. This configuration was chosen empirically as it balances encoding overhead and stitching complexity, aligning with prior tile-based streaming works [48, 49].

- Raw frames are captured from the screen (simulating a camera) using **FFmpeg** [50].
- These frames are encoded into tiled HEVC segments.
- **GPAC** [51] is used to manipulate tiles and pack them into MP4 files.
- Each segment contains **4 frames at 25 fps**. The frame rate has been borrowed from

previous live streaming works like [18, 52]. 4 frames in a segment can be justified from an evaluation done in COMPACT which shows that increasing this number increases the file size as well as the encoding time. 4 frames per segment balances compression efficiency and encoding speed.

5.2 Foreground Detection

To make COMPACT **content-aware**, we needed a model to specifically detect human faces. We use a pre-trained **ResNet** model [53] for **face detection** because it's a lightweight model which completes it's one iteration in around 70ms on CPU with configuration: 11th Gen Intel(R) Core(TM) i7-11700 @ 2.50GHz with 16 cores. This time is useful for us as it enables us to parallelize this with screen capturing and encoding time. Additionally, ResNet has powerful feature extraction capabilities, robust training behavior, and state-of-the-art performance. Further we :

- Tiles containing detected face bounding boxes are labeled as **foreground (FG)**.
- Remaining tiles are considered **background (BG)**.
- In slide-based content where no faces are detected, we apply a horizontal split with 8 tiles each for FG and BG.

5.3 Transport Layer

We adopt **QUIC over UDP** as the transport layer protocol. The control messages are sent over **UDP**. The motivation for choosing QUIC specifically, is discussed in Sec. 3.4. There are multiple implementations of QUIC[54–57]. We selected PicoQuic[38] as the base for QUIC because of its minimalist and lightweight implementation. A recent study [58], has shown that PicoQuic shows one of the best results in terms of throughput when compared to other available QUIC's implementations. I extended PicoQuic's implementation to be compatible with our COMPACT's framework.

5.4 Asynchronous Processing

Tasks on both the streamer and receiver sides are **asynchronously executed** using **multi-threading**. We choose asynchronous execution so that different modules of the framework like the screen capturing, FG detection, scheduling, sending/receiving and playing do not have to wait for each other in consecutive iterations to complete their execution. These can be done in parallel. Obviously, for one iteration, these are pipelined.

We also use Evicting Queues to accommodate this asynchronous execution. These are mutually exclusive queues with a fixed size 2. These are shared by capture and sending threads on the streamer side and receiver and player threads on the player side. The size 2 is chosen to give the real time experience to the user. Since each segment is of 160 seconds, size 2 helps us to keep the real time experience at maximum 320 sec behind. Beyond this the lag starts to hurt the real time QoE. We do not pick the size as 1, due to the fact that the system might incur many losses since our capture thread is faster than the sending thread. Therefore, we stick to the value 2 for our evicting queues.

5.4.1 Player and Stitcher

On the player side, once the FG and BG tiles are received, their indices are matched. If both the BG tiles and BG tiles are of the same segment index, FG and BG tiles are simply stitched and played as a complete segment. If the BG tiles are delayed, we use the FG from the previous segment index and then stitch and play both as a whole segment. Trailing FG corresponds to Stalls. The stitching and playing on the player side is done using OpenCV.

Chapter 6

Evaluation and Results

6.1 Evaluation

6.1.1 Simulated Testbed for Evaluation

The testbed comprises three physical machines: Streamer PC, Player PC, and Helper PC, set up to evaluate the proposed COMPACT framework under realistic multipath conditions as shown in Fig. 6.1.

Streamer PC: Acts as the video server, running the modified COMPACT server implementation built over PicoQuic. It captures the Live video, handles foreground/background tile identification and segregation and finally, it schedules the tiles on each path and sends them over QUIC. The scheduling decision is dependent on the slower and faster paths which are computed by the Bandwidth and RTT estimates computed by the streamer. The traffic from this PC is routed to the Player PC via mahimahi traces on the Helper PC.

Player PC: Acts as the video client, running the COMPACT client which receives and renders video segments, reconstructing frames from separately transmitted foreground (FG) and background (BG) tiles. It initiates several communication channels and send connection requests to the Helper PC:

- **Data Channels:** It initiates two data channels, to simulate the multipath scenario. These

channels are used to send and receive the video traffic.

- **RTT Channels:** It initiates two RTT channels to estimate the round trip time incurred on each path.

- **Bandwidth Channels:** It initiates two control channels to estimate the bandwidth of each path.

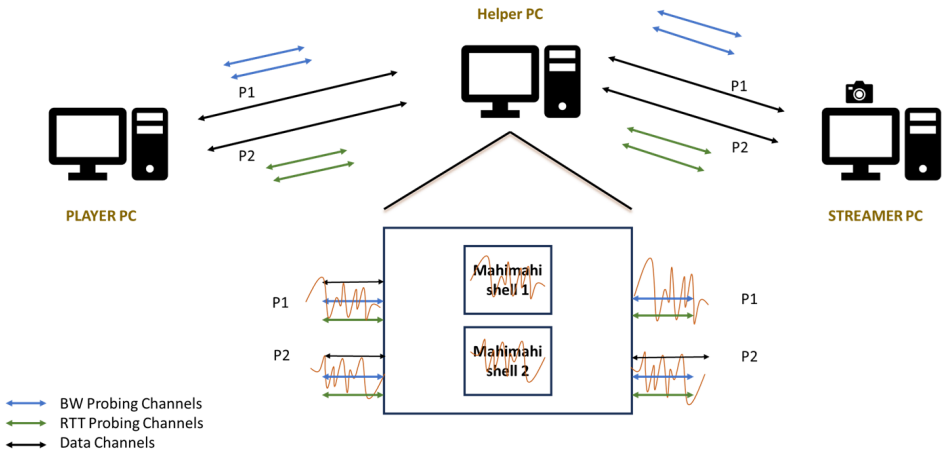


Figure 6.1: Testbed for Simulated Evaluation

Helper PC: Plays a critical role in simulating a multipath environment using network emulation. It also functions as a relay node, facilitating the forwarding of data between the server and client while introducing controlled network diversity. The most important usage of this PC, is to replay real world network fluctuations in a simulated environment using mahimahi.We run two instances of Mahimahi shell (Shell 1 and Shell 2) to simulate two independent network paths with distinct characteristics. These emulate real-world conditions by applying bandwidth throttling and delay to the traffic. Once the Hepler PC gets the traffic from the Player PC, it goes through these mahimahi traces and then further forwarded to the Streamer PC.

6.1.2 Traces for Evaluation

These three cellular traces — bus, car, and walk as show in Fig. 6.2 — collectively cover a broad spectrum of real-world network conditions, ranging from stationary to highly mobile scenarios, and capture diverse bandwidth dynamics typical of urban wireless environments.

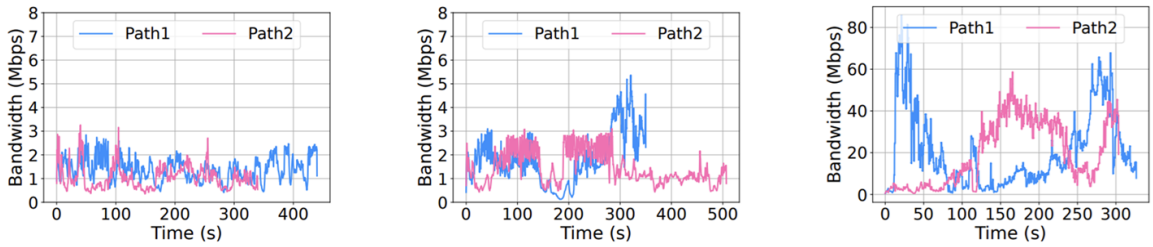


Figure 6.2: Traces for Simulated Evaluation a) Bus b) Car c) Walk

Bus:

- The bandwidth is consistently low (mostly 1–3 Mbps) with minor fluctuations.
- Both Path1 and Path2 follow a similar pattern, indicating limited variability in available bandwidth over time.

Buses typically follow fixed routes through urban/suburban areas and often move slowly or stop frequently. As a result, the device stays within the same cell or network conditions for a longer time, leading to stable but low-quality connections. Buses may also be shared environments with many users, leading to contention for available bandwidth.

Car:

- Bandwidth shows moderate values with frequent fluctuations, reaching peaks around 2–4 Mbps.
- Bandwidth for both the paths are rising and dipping occasionally, with path 1 going beyond 5 Mbps at a certain time.

This aligns with vehicular mobility, where rapid movement through various cells causes frequent handovers and transient signal quality drops. As the vehicle moves through varying terrain and network coverage, bandwidth can quickly shift — leading to some buffering events, but rarely for too long in most cases. However, in rare scenarios, a car may hit dead zones or have handover delays, causing long stalls or high lag.

Walk:

- Bandwidth for this trace is particularly good as compared to above traces.

- In this trace, path 1 reaches very high peaks up to 80 Mbps and path 2 reached upto 60 Mbps with sustained periods of good throughput, suggesting excellent network conditions for much of the session.

This is likely because walking users remain in strong coverage zones longer, such as on sidewalks, near Wi-Fi offload zones, or close to cell towers. The result is smooth streaming most of the time — very low median stalls and lag. However, when walking into deep coverage holes (e.g., buildings, tunnels), users can remain there for a long time, leading to massive stalls and high lag.

6.1.3 Videos for Evaluation

To evaluate the effectiveness of COMPACT in handling diverse content structures and visual dynamics in live video streaming, we selected three distinct video sequences, each emulating a different style of online classroom session. These videos (Fig. 6.3) help assess how foreground-background separation and content-aware scheduling adapt to varying scene complexities.

(a) Video 1 (b) Video 2 (c) Video 3

Figure 6.3: Videos for Evaluation

- **Video 1** – Simple Background with Two Instructors This video contains two instructors speaking in a split-screen layout against a plain, static background.

  The foreground regions (instructors' faces) remain relatively stable across frames, while the background changes infrequently.

  This scenario is ideal for evaluating COMPACT 's ability to reuse background tiles efficiently and prioritize consistent foreground delivery.

- **Video 2** – Complex Background with Slides This video includes a single instructor and a dynamic background comprising notes and slide transitions.

The visual complexity increases due to changing slide content, requiring more frequent updates of background tiles.

It provides a good test case for understanding how COMPACT adapts to frequent background changes while maintaining foreground stability.

- **Video 3** – Instructor with Animation and Movement

  This sequence features an instructor moving around, along with some animated elements (e.g., gestures, drawing on a virtual whiteboard).

  It poses challenges for both foreground detection and tile-level scheduling, as frequent motion leads to shifting FG regions and potential bandwidth bottlenecks.

  It is particularly useful for testing the scheduler's responsiveness and real-time adaptation to rapid visual changes.

6.1.4 QoE Metrics for Evaluation

To objectively assess the performance of COMPACT in live video streaming, we use two primary **Quality of Experience (QoE)** metrics: **Stall Duration** and **End-to-End (E2E) Lag**. These metrics reflect the smoothness and responsiveness of the video playback from a user's perspective.

6.1.4.1 Stall Duration

Stall Duration refers to the time the video playback is interrupted due to the delayed arrival of video tiles, particularly **foreground (FG) tiles** which are essential for rendering a complete frame. A stall occurs when the *player finishes displaying segment* P_i, but cannot begin playback of the next segment P_{i+1} because one or more of its required tiles are missing or delayed.

The stall is measured as the time difference between the end of playback of segment P_i and the start of playback of segment P_{i+1} as elaborated in Fig. 6.4.

$$\text{Stall Duration} = t_2 - t_1 \tag{6.1}$$

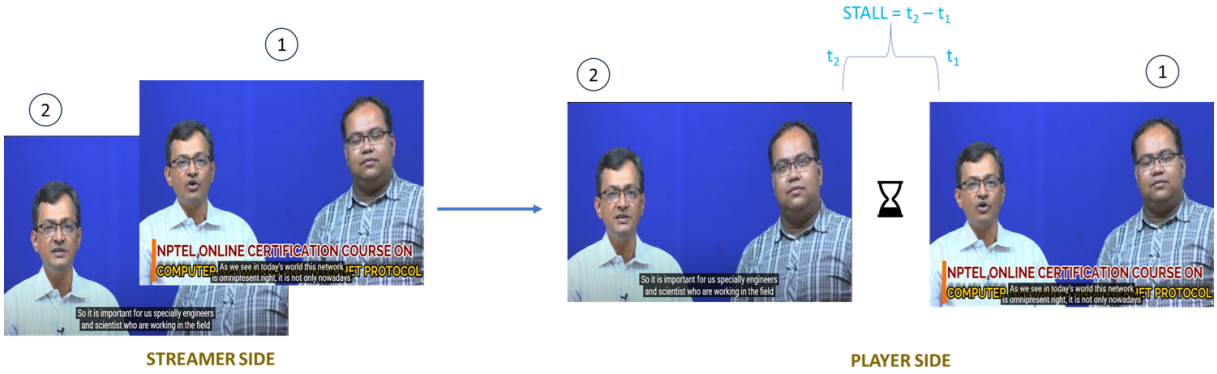


Figure 6.4: Stalling in Video Streaming

where:

- t_1: Time when playback of segment P_i completes.
- t_2: Time when playback of segment P_{i+1} starts.

Impact: High stall durations lead to user frustration and degraded perceived quality, especially in live applications such as online classrooms or real-time events.

6.1.4.2 End-to-End (E2E) Lag

End-to-End Lag is defined as the delay between when a video frame is generated (captured or rendered) on the **Streamer PC** and when it is displayed on the **Player PC** as elaborated in Fig. 6.5.

This metric captures the total transmission, scheduling, and rendering delay introduced in the streaming pipeline. It reflects the **true responsiveness** of the system, which is particularly important for **real-time interactive applications**.

$$\text{E2E Lag} = t_2 - t_1 \tag{6.2}$$

where:

- t_1: Time at which the frame is streamed or appears on the streamer screen.

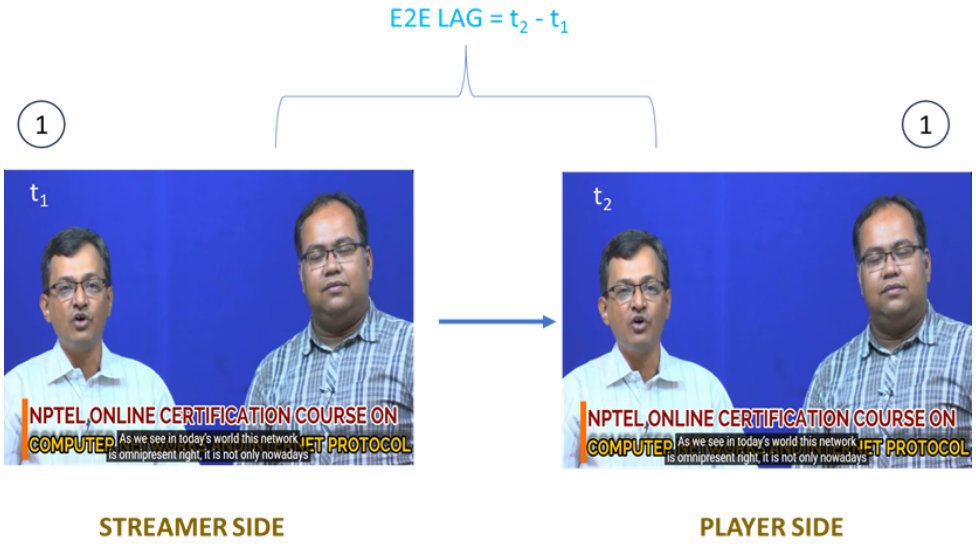


Figure 6.5: End to End Lag in Video Streaming

- t_2: Time at which the same frame is rendered on the viewer's screen.

Measurement Method: We use **OCR-based timestamp detection** on both the streamer and player video feeds to compute this value precisely and non-intrusively.

Impact: Lower E2E lag ensures that users experience video with minimal delay, which is essential for applications such as live teaching, news broadcasting, or remote collaboration.

Together, these metrics provide a comprehensive understanding of how the system handles **latency and playback continuity** under real-world multipath network conditions.

6.1.5 Baseline for Evaluation

To meaningfully assess the performance of COMPACT using multipath over QUIC, we compare it against a baseline scheduler that does not use content-aware prioritization. The baseline chosen is the widely used MinRTT scheduler, which is representative of traditional multipath scheduling strategies.

MinRTT Scheduler The MinRTT scheduler[39] selects the path with the minimum Round-Trip Time (RTT) and pushes the majority of the data through that path. It is content-agnostic, meaning it treats all tiles equally regardless of their visual importance in the video. Infact, in this

baseline, there is no concept of tiles.

For every video segment, the path with the lower RTT at that moment is assigned the majority of the tiles (both FG and BG), aiming to minimize latency through low-delay routing. This scheduler reflects the behavior of standard Multipath-QUIC implementations, where the objective is to maximize throughput and minimize delay, without consideration of application-layer content or user QoE.

For a Sanity Check, we also compare the performance of COMPACT over QUIC with its original implementation over SCTP (Stream Control Transmission Protocol). This comparison helps validate that the migration to QUIC preserves the core benefits of COMPACT.

6.2 Results

6.2.1 COMPACT over SCTP v/s COMPACT over QUIC

- **STALL**

  Fig. 6.6 shows that the median stall values for SCTP is a little lower as compared to QUIC, whereas, the 99th percentile for stall is better for QUIC than SCTP. The reason for this kind of observation can be contributed by the Congestion Control Algorithms(CCA) employed by both of these protocols.

  SCTP is accustomed to employ a TCP Reno[59] styled CCA, which tends to increase the congestion window aggressively, resulting in lower median for stalls. Whereas, QUIC employs BBR[60] CCA, which tends to operate on BDP (Buffer Delay Product) and does not unnecessarily increase the congestion Window to incur loss. But SCTP incur loss due to aggressive Cwnd increase. This reason can account for the better 99th percentile in case of QUIC.

- **E2E Lag**

  The E2E lag values, both median and 99th percentile for SCTP is are a little higher for than for QUIC as can be seen in Fig. 6.7. The above reasoning of Congestion Control

Algorithms can be suited here. BBR helps to maintain the liveliness of the video without incurring much losses as compared to TCP Reno styled CCA for SCTP.

Also, for CAR trace, we can observe that the 99th percentile went quite up as compared to other traces. The reasoning has been duly pointed out the description of the trace which is that this trace is much fluctuating as compared to other traces.

Figure 6.6: Stalls for COMPACT over SCTP vs QUIC

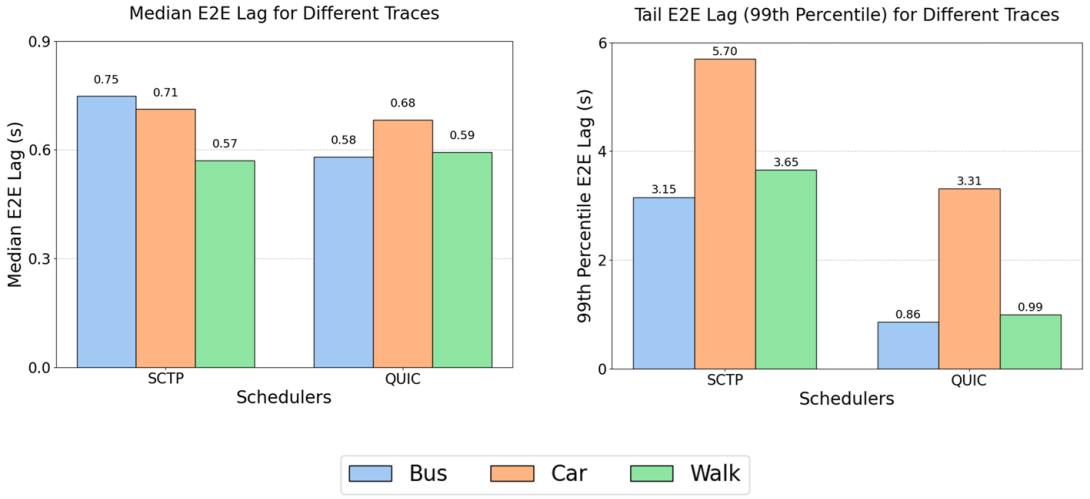


Figure 6.7: E2E lag for COMPACT over SCTP vs QUIC

6.2.2 MinRTT over QUIC v/s COMPACT over QUIC

- **STALL**

  We can clearly see in Fig. 6.8 that both median and 99th percentile values for MinRTT over QUIC, clearly surpasses the respective values for COMPACT over QUIC. Even the best case for minRTT on walk trace is far behind COMPACT. This can be accounted by the classic HOL observed in schedulers like minRTT. Particularly, for the 99th percentile the results for walk trace goes exceptionally higher, due to the fact that schedulers with no capability to handle situations where all the paths go down at once.

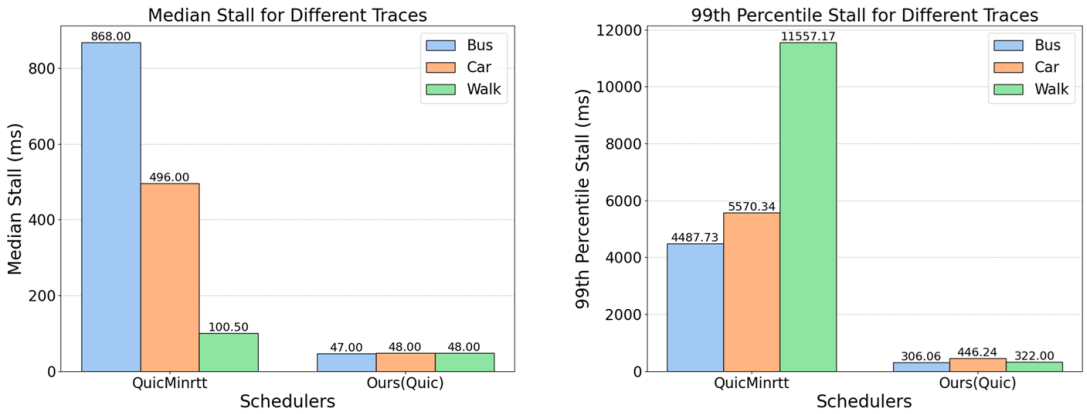


Figure 6.8: Stalls for MinRTT over QUIC vs COMPACT over QUIC

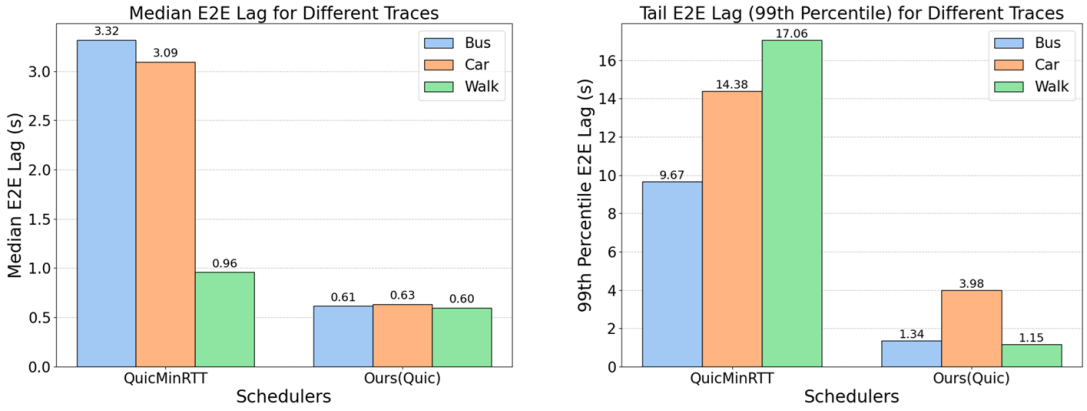


Figure 6.9: E2E lag for MinRTT over QUIC vs COMPACT over QUIC

- **E2E LAG** The very same observation can be seen in Fig. 6.9 for E2E values for MinRTT and COMPACT. MinRTT tens to schedule its maximum traffic on path with minimum RTT.

But if the network goes down, spontaneously, minRTT has no intelligence to shift that traffic to other path. The traffic keeps getting queued, till the correct estimated of the RTT values is received by the scheduler.

Chapter 7

Conclusion and Future Work

This work presents the implementation and evaluation of COMPACT —a content-aware multipath video streaming framework—over QUIC, a modern, UDP-based transport protocol. COMPACT intelligently separates video frames into foreground (FG) and background (BG) tiles and schedules them across heterogeneous paths based on visual importance and path characteristics. By leveraging PicoQuic's features such as multiplexed streams, user-space deployment, and 0-RTT connection establishment, the extended framework achieves improved Quality of Experience (QoE) for live video applications, especially in bandwidth-constrained and high-latency environments.

Through extensive testing using real-world network traces and diverse video scenarios, we demonstrate that COMPACT over QUIC significantly outperforms traditional schedulers like MinRTT in reducing stall duration, end-to-end lag, and quality fluctuations, while maintaining compatibility with modern Internet protocols. Additionally, our comparison with the SCTP-based version of COMPACT confirms that the transition to QUIC preserves its performance advantages and improves deployability.

The Future Work will extend to utilize COMPACT to cater to other live streaming applications like live sports, news broadcasting and some other real - time applications where where foreground-background dynamics and low latency are equally critical.

Bibliography

[1] X. Chen, S. Chen, X. Wang, and Y. Huang, “"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.*, vol. 4, no. CSCW3, pp. 237:1–237:32, January 2021.

[2] “Online Learning Statistics: The Ultimate List in 2024 | Devlin Peck,” 2024-11-14.

[3] G. C. Ph.D, “2024 Online Learning Statistics,” June 2024, section: Online Colleges.

[4] T. Faas, L. Dombrowski, A. Young, and A. D. Miller, “Watch Me Code: Programming Mentorship Communities on Twitch.tv,” *Proceedings of the ACM on Human-Computer Interaction*, vol. 2, no. CSCW, pp. 1–18, November 2018.

[5] D. L. Chen, D. Freeman, and R. Balakrishnan, “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*, ser. CHI ’19. New York, NY, USA: Association for Computing Machinery, May 2019, pp. 1–14.

[6] M. Wu and Q. Gao, “Using live video streaming in online tutoring: Exploring factors affecting social interaction,” *International Journal of Human-Computer Interaction*, vol. 36, no. 10, pp. 964–977, 2020.

[7] P. Sarda, “Why youtube still rules the edtech roost,” https://www.forbesindia.com/article/edtech-special/why-youtube-still-rules-the-edtech-roost/57761/1, 2020, published on Feb 18, 2020; Accessed on Apr 12, 2024.

[8] R. Article, A. Khan, M. Saeed, M. Anwar, and L. Kanwal, "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*, vol. 13:05, 2023, September 2023.

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

[10] S. Aggarwal, M. Ghoshal, P. Banerjee, D. Koutsonikolas, and J. Widmer, "802.11ad in Smartphones: Energy Efficiency, Spatial Reuse, and Impact on Applications," in *IEEE INFOCOM 2021 - IEEE Conference on Computer Communications*. Vancouver, BC, Canada: IEEE, May 2021, pp. 1–10.

[11] D. Xu, A. Zhou, X. Zhang, G. Wang, X. Liu, C. An, Y. Shi, L. Liu, and H. Ma, "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*. Virtual Event USA: ACM, July 2020, pp. 479–494.

[12] X. Yuan, M. Wu, Z. Wang, Y. Zhu, M. Ma, J. Guo, Z.-L. Zhang, and W. Zhu, "Understanding 5G performance for real-world services: a content provider's perspective," in *Proceedings of the ACM SIGCOMM 2022 Conference*. Amsterdam Netherlands: ACM, August 2022, pp. 101–113.

[13] Y. Zeng, P. H. Pathak, and P. Mohapatra, "A first look at 802.11ac in action: Energy efficiency and interference characterization," in *2014 IFIP Networking Conference*. Trondheim, Norway: IEEE, June 2014, pp. 1–9.

[14] J. He, M. Ammar, and E. Zegura, "A Measurement-Derived Functional Model for the Interaction Between Congestion Control and QoE in Video Conferencing," in *Passive and Active Measurement*, A. Brunstrom, M. Flores, and M. Fiore, Eds. Cham: Springer Nature Switzerland, 2023, pp. 129–159.

[15] K. MacMillan, T. Mangla, J. Saxon, and N. Feamster, "Measuring the performance and network utilization of popular video conferencing applications," in *Proceedings of the 21st ACM Internet Measurement Conference*. Virtual Event: ACM, November 2021, pp. 229–244.

[16] H. Lim, J. Lee, J. Lee, S. D. Sathyanarayana, J. Kim, A. Nguyen, K. T. Kim, Y. Im, M. Chiang, D. Grunwald, K. Lee, and S. Ha, "An Empirical Study of 5G: Effect of Edge on Transport Protocol and Application Performance," *IEEE Transactions on Mobile Computing*, vol. 23, no. 4, pp. 3172–3186, April 2024.

[17] A. Hassan, A. Narayanan, A. Zhang, W. Ye, R. Zhu, S. Jin, J. Carpenter, Z. M. Mao, F. Qian, and Z.-L. Zhang, "Vivisecting mobility management in 5G cellular networks," in *Proceedings of the ACM SIGCOMM 2022 Conference*. Amsterdam Netherlands: ACM, August 2022, pp. 86–100.

[18] S. Dhawaskar Sathyanarayana, K. Lee, D. Grunwald, and S. Ha, "Converge: QoE-driven Multipath Video Conferencing over WebRTC," in *Proceedings of the ACM SIGCOMM 2023 Conference*. New York NY USA: ACM, September 2023, pp. 637–653.

[19] H. Wang, Z. Yu, R. Zhang, S. Tao, H. Yu, and S. Shi, "TwinStar: A Practical Multi-path Transmission Framework for Ultra-Low Latency Video Delivery," in *Proceedings of the 31st ACM International Conference on Multimedia*. Ottawa ON Canada: ACM, October 2023, pp. 9234–9242.

[20] J. E, L. He, Z. Zhao, Y. Wang, G. Chen, and W. Chen, "AggDeliv: Aggregating Multiple Wireless Links for Efficient Mobile Live Video Delivery," in *IEEE INFOCOM 2024 - IEEE Conference on Computer Communications*, May 2024, pp. 1173–1180.

[21] J. Wu, C. Yuen, M. Wang, and J. Chen, "Content-Aware Concurrent Multipath Transfer for High-Definition Video Streaming over Heterogeneous Wireless Networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 27, no. 3, pp. 710–723, March 2016.

[22] J. Wu, B. Cheng, C. Yuen, Y. Shang, and J. Chen, "Distortion-Aware Concurrent Multipath Transfer for Mobile Video Streaming in Heterogeneous Wireless Networks," *IEEE Transactions on Mobile Computing*, vol. 14, no. 4, pp. 688–701, April 2015.

[23] J. Wu, B. Cheng, and M. Wang, "Improving Multipath Video Transmission With Raptor Codes in Heterogeneous Wireless Networks," *IEEE Transactions on Multimedia*, vol. 20, no. 2, pp. 457–472, February 2018.

[24] C. Xu, Z. Li, J. Li, H. Zhang, and G.-M. Muntean, "Cross-Layer Fairness-Driven Concurrent Multipath Video Delivery Over Heterogeneous Wireless Networks," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 25, no. 7, pp. 1175–1189, July 2015.

[25] C. Wang, H. Wang, F. Qian, K. Zheng, C. Wang, F. Mao, X. Guo, and C. Xu, "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*. Madrid Spain: ACM, July 2023, pp. 1–15.

[26] Z. Zheng, Y. Ma, Y. Liu, F. Yang, Z. Li, Y. Zhang, J. Zhang, W. Shi, W. Chen, D. Li, Q. An, H. Hong, H. H. Liu, and M. Zhang, "XLINK: QoE-driven multi-path QUIC transport in large-scale video services," in *Proceedings of the 2021 ACM SIGCOMM 2021 Conference*. Virtual Event USA: ACM, August 2021, pp. 418–432.

[27] X. Zhu, J. Sun, X. Zhang, Y. E. Guo, F. Qian, and Z. M. Mao, "Mpbond: efficient network-level collaboration among personal mobile devices," in *Proceedings of the 18th International Conference on Mobile Systems, Applications, and Services*, 2020, pp. 364–376.

[28] L. Keller, A. Le, B. Cici, H. Seferoglu, C. Fragouli, and A. Markopoulou, "MicroCast: cooperative video streaming on smartphones," in *Proceedings of the 10th international conference on Mobile systems, applications, and services*. Low Wood Bay Lake District UK: ACM, June 2012, pp. 57–70.

[29] S. Jin, R. Zhu, A. Hassan, X. Zhu, X. Zhang, Z. M. Mao, F. Qian, and Z.-L. Zhang, "Oasis: Collaborative neural-enhanced mobile video streaming," in *Proceedings of the 15th ACM*

Multimedia Systems Conference, ser. MMSys '24. New York, NY, USA: Association for Computing Machinery, 2024, p. 45–55.

[30] C. Raiciu, M. Handley, and D. Wischik, "Coupled congestion control for multipath transport protocols," Tech. Rep., 2011.

[31] D. Wischik, C. Raiciu, A. Greenhalgh, and M. Handley, "Design, implementation and evaluation of congestion control for multipath TCP," Mar. 2011. [Online]. Available: https://www.usenix.org/conference/nsdi11/design-implementation-and-evaluation-congestion-control-multipath-tcp

[32] C. Raiciu, C. Paasch, S. Barre, A. Ford, M. Honda, F. Duchene, O. Bonaventure, and M. Handley, "How Hard Can It Be? Designing and Implementing a Deployable Multipath TCP."

[33] Q. De Coninck and O. Bonaventure, "Multipath QUIC: Design and Evaluation," in *Proceedings of the 13th International Conference on emerging Networking EXperiments and Technologies*. Incheon Republic of Korea: ACM, November 2017, pp. 160–166.

[34] Q. De Coninck, "The packet number space debate in multipath QUIC," *ACM SIGCOMM Computer Communication Review*, vol. 52, no. 3, pp. 2–9, July 2022.

[35] Q. De Coninck and O. Bonaventure, "Multipath QUIC: Design and Evaluation," in *Proceedings of the 13th International Conference on emerging Networking EXperiments and Technologies*. Incheon Republic of Korea: ACM, November 2017, pp. 160–166.

[36] V. Singh, S. Ahsan, and J. Ott, "MPRTP: multipath considerations for real-time media," in *Proceedings of the 4th ACM Multimedia Systems Conference*. Oslo Norway: ACM, February 2013, pp. 190–201.

[37] S. Chaudhary, N. Mishra, K. Gambhir, T. Rajore, A. Bhattacharya, and M. Maity, "Compact: Content-aware multipath live video streaming for online classes using video tiles," in *Proceedings of the 16th ACM Multimedia Systems Conference*, ser. MMSys '25. New York, NY, USA: Association for Computing Machinery, 2025, p. 201–213. [Online]. Available: https://doi.org/10.1145/3712676.3714451

[38] [Online]. Available: https://github.com/private-octopus/picoquic

[39] P. Hurtig, K.-J. Grinnemo, A. Brunstrom, S. Ferlin, Alay, and N. Kuhn, "Low-latency scheduling in mptcp," *IEEE/ACM Transactions on Networking*, vol. 27, no. 1, pp. 302–315, 2019.

[40] S. K. Saha, S. Aggarwal, R. Pathak, D. Koutsonikolas, and J. Widmer, "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*. Los Cabos Mexico: ACM, October 2019, pp. 1–16.

[41] G. Lv, Q. Wu, Y. Liu, Z. Li, Q. Tan, F. Yang, W. Chen, Y. Ma, H. Guo, Y. Chen, and G. Xie, "Chorus: Coordinating mobile multipath scheduling and adaptive video streaming," in *Proceedings of the 30th Annual International Conference on Mobile Computing and Networking*, ser. ACM MobiCom '24. New York, NY, USA: Association for Computing Machinery, 2024, p. 246–262.

[42] T. D. Wallace and A. Shami, "A Review of Multihoming Issues Using the Stream Control Transmission Protocol," *IEEE Communications Surveys & Tutorials*, vol. 14, no. 2, pp. 565–578, 2012.

[43] R. R. Stewart, "Stream Control Transmission Protocol," Internet Engineering Task Force, Request for Comments RFC 4960, September 2007, num Pages: 152.

[44] P. Natarajan, P. P. D. Amer, J. Leighton, and F. Baker, "Using SCTP as a Transport Layer Protocol for HTTP," Internet Engineering Task Force, Internet Draft draft-natarajan-http-over-sctp-02, July 2009, num Pages: 13.

[45] A. Langley, A. Riddoch, A. Wilk, A. Vicente, C. Krasic, D. Zhang, F. Yang, F. Kouranov, I. Swett, J. Iyengar, J. Bailey, J. Dorfman, J. Roskind, J. Kulik, P. Westin, R. Tenneti, R. Shade, R. Hamilton, V. Vasiliev, W.-T. Chang, and Z. Shi, "The quic transport protocol: Design and internet-scale deployment," in *Proceedings of the Conference of the ACM Special Interest Group on Data Communication*, ser. SIGCOMM '17. New York, NY, USA: Association for Computing Machinery, 2017, p. 183–196.

[46] G. J. Sullivan, J.-R. Ohm, W.-J. Han, and T. Wiegand, "Overview of the high efficiency video coding (hevc) standard," *IEEE Transactions on circuits and systems for video technology*, vol. 22, no. 12, pp. 1649–1668, 2012.

[47] M. Viitanen, A. Koivula, A. Lemmetti, A. Ylä-Outinen, J. Vanne, and T. D. Hämäläinen, "Kvazaar: open-source hevc/h. 265 encoder," in *Proceedings of the 24th ACM international conference on Multimedia*, 2016, pp. 1179–1182.

[48] H. Guo, S. Yao, Z. Yang, Q. Zhou, and K. Nahrstedt, "Crossroi: cross-camera region of interest optimization for efficient real time video analytics at scale," in *Proceedings of the 12th ACM Multimedia Systems Conference*, 2021, pp. 186–199.

[49] S. Chaudhary, A. Taneja, A. Singh, P. Roy, S. Sikdar, M. Maity, and A. Bhattacharya, "Tileclipper: Lightweight selection of regions of interest from videos for traffic surveillance," in *2024 USENIX Annual Technical Conference (USENIX ATC 24)*, 2024, pp. 967–984.

[50] "Ffmpeg," https://www.ffmpeg.org/, 2024.

[51] J. Le Feuvre, C. Concolato, and J.-C. Moissinac, "Gpac: open source multimedia framework," in *Proceedings of the 15th ACM international conference on Multimedia*, 2007, pp. 1009–1012.

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

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

[54] "microsoft/msquic," May 2025, original-date: 2019-10-26T04:10:24Z. [Online]. Available: https://github.com/microsoft/msquic

[55] "cloudflare/quiche," May 2025, original-date: 2018-09-29T18:22:05Z. [Online]. Available: https://github.com/cloudflare/quiche

[56] "facebook/mvfst," May 2025, original-date: 2018-04-09T22:49:15Z. [Online]. Available: https://github.com/facebook/mvfst

[57] [Online]. Available: https://github.com/quicwg/base-drafts/wiki/Implementations

[58] P. Bi, Y. Zou, M. Xiao, D. Yu, Y. Li, Z. Liu, and Q. Xie, "Litequic: Improving qoe of video streams by reducing cpu overhead of quic," in *Proceedings of the 32nd ACM International Conference on Multimedia*, ser. MM '24. New York, NY, USA: Association for Computing Machinery, 2024, p. 7918–7927. [Online]. Available: https://doi.org/10.1145/3664647.3681670

[59] J. Padhye, V. Firoiu, and D. Towsley, "A stochastic model of tcp reno congestion avoidence and control," USA, Tech. Rep., 1999.

[60] N. Cardwell, Y. Cheng, C. S. Gunn, S. H. Yeganeh, and V. Jacobson, "Bbr: Congestion-based congestion control: Measuring bottleneck bandwidth and round-trip propagation time," *Queue*, vol. 14, no. 5, p. 20–53, Oct. 2016. [Online]. Available: https://doi.org/10.1145/3012426.3022184