Chapter 4: Results and Discussion

4.1 Introduction

This chapter presents a comprehensive analysis and in-depth discussion of the empirical results obtained from the simulation framework, the design and mechanics of which were meticulously detailed in Chapter 3: Methodology. The overarching aim of this research, as stated in Chapter 1: Introduction, is to critically analyze and propose effective strategies for optimizing video streaming latency. A core focus is to unravel and quantify the factors contributing to latency discrepancies experienced by users under heterogeneous network conditions, a challenge particularly pronounced in dynamic mobile environments. The simulations, forming the backbone of this empirical investigation, were specifically engineered to evaluate the performance of Adaptive Bitrate (ABR) streaming algorithms. Particular attention is given to the custom-developed `LatencyAwareAbr`, which was tested against a backdrop of realistic bandwidth conditions derived from real-world mobile network traces and synthetically varied Round-Trip Times (RTTs) to mimic diverse user network paths.

The findings of this chapter will be articulated through a synergistic combination of aggregated Key Performance Indicators (KPIs), providing a quantitative snapshot of performance, and detailed time-series plots, which offer a granular view of the dynamic behavior of the simulated streaming clients over time. The ensuing discussion will pivot around interpreting these findings in direct relation to the research questions and objectives posited in Chapter 1. Specifically, this chapter seeks to address:

1. The quantifiable impact of network RTT on various facets of perceived playback latency (including startup delay, stall duration, and overall playback progression) and its cascading effects on the holistic Quality of Experience (QoE).
2. The magnitude and dynamics of the playback time difference (referred to as viewer discrepancy or relative latency) between simulated clients operating with low versus high RTTs, a critical factor for synchronized or live viewing experiences.
3. A detailed performance assessment and behavioral analysis of the `LatencyAwareAbr` algorithm, examining its adaptiveness and efficacy under these varied and often challenging network conditions.
4. The broader implications of these findings for the design and implementation of more robust and effective latency optimization strategies in the domain of mobile video streaming, potentially informing future algorithm development as discussed in relation to Objective 3 in Chapter 1.
5. How these results align with or diverge from existing knowledge as reviewed in Chapter 2: Literature Review, and how they contribute to bridging identified gaps.

4.2 Simulation Setup and Parameters: A Brief Recapitulation

To ensure clarity and context for the results presented, this section briefly revisits the crucial components of the simulation environment, as exhaustively defined in Section 3.2: Data Collection Methods and Section 3.3: Algorithm Development of Chapter 3.

1. Adaptive Bitrate (ABR) Algorithms Under Test:
    i. `LatencyAwareAbr`: This is the central ABR algorithm developed and evaluated in this study. Its design philosophy, detailed in Section 3.3.1: Latency-Aware Adaptation Logic, explicitly incorporates network delay (RTT) into its bitrate selection process. It leverages simple predictive techniques for both throughput (harmonic mean of recent measurements) and delay (moving average), and employs a defined conservative buffer threshold (`LATENCY_AWARE_CONSERVATIVE_BUFFER_S`) to govern quality upshifting decisions, aiming to balance aggressiveness with stability.
    ii. *(Implicitly, results from `LatencyAwareAbr` could be benchmarked against a more traditional `BufferBasedAbr` or other algorithms outlined in Chapter 2, though the primary focus here is on the differential impact of RTT on clients using `LatencyAwareAbr`.)*
2. Bandwidth Traces (Source of Realism):
    i. To ensure ecological validity, real-world HSDPA (High-Speed Downlink Packet Access) mobile network bandwidth traces were employed. These traces, sourced from the publicly available repository at `skulddata.cs.umass.edu` (similar to the MONROE dataset mentioned in Section 3.2.1), capture the inherent variability of mobile network capacity.
    ii. The specific traces forming the basis of the results in this chapter are:
        - `Bus`: Captures bandwidth dynamics typical of vehicular movement within urban or suburban landscapes, characterized by moderate to high variability.
        - `Metro-Tunnel`: Represents a highly challenging scenario with periods of severe signal degradation and potential complete loss of connectivity, common in underground transit.
        - `Train-Fast`: Simulates conditions on a faster-moving train, likely featuring frequent cell handovers and a mix of open-air and potentially obstructed signal paths.
3. Client Configurations for Latency Discrepancy Analysis:
    i. A key experimental design element, directly addressing the project's focus on viewer latency differences, involved simulating two primary client profiles for each bandwidth trace. These clients shared the identical bandwidth input but were subjected to distinct network delay characteristics:

        - `Client1_LowLatency`: Configured with a base RTT of 30ms, with fluctuations up to 50% of this base, simulating a user with a relatively good network path.
        - `Client2_HighLatency`: Configured with a significantly higher base RTT of 150ms, with fluctuations up to 30% of its base, simulating a user with a poorer, higher-latency network path.
    ii. Crucially, for the direct comparison of RTT impact, both clients utilized the same `LatencyAwareAbr` algorithm. This experimental control ensures that observed differences in performance and playback progression are primarily attributable to the RTT differential rather than ABR logic differences.
4. Key Performance Indicators (KPIs) and Metrics for Evaluation:
    i. The evaluation framework, as described in Section 3.4.1: Objective Evaluation Metrics, relies on a suite of quantitative KPIs:
        - Startup Delay (seconds): Time from request to first frame playback.
        - Total Stall Time (seconds) and Buffering Ratio (% of playback time spent stalled).
        - Average Selected Bitrate (kbps): Average video quality delivered during active playback.
        - Number of Quality Switches and Switches per Minute: Indicators of playback stability.
        - A Simple QoE Score: A composite metric attempting to holistically capture user experience, factoring in bitrate benefits and penalties for impairments.
        - Per-Step Playback Time (seconds): Logged at each simulation timestep (`TIME_STEP_S`), this forms the basis for calculating the crucial Playback Latency Difference (seconds) between `Client1_LowLatency` and `Client2_HighLatency`.
5. Simulation Granularity and Duration:
    i. Simulations proceeded with a `TIME_STEP_S` of 0.1 seconds.
    ii. The total `SIMULATION_DURATION_S` for each run was dynamically set to the length of the loaded bandwidth trace, ensuring full utilization of the empirical network data.

4.3 Analysis of Aggregated Key Performance Indicators (KPIs)

Aggregated KPIs offer a high-level, quantitative comparison of client performance under the different RTT profiles across the spectrum of tested network traces. Table 4.1, derived from the simulation output, summarizes the average metrics for `Client1_LowLatency` and `Client2_HighLatency`, both employing the `LatencyAwareAbr` algorithm, averaged over the Bus, Metro-Tunnel, and Train-Fast traces. The metrics for individual traces were also

presented in the raw simulation output and will be referred to when discussing specific trace behaviors.

Table 4.1: Average Performance Metrics Across Traces (ABR: LatencyAware)

Client Configuration	Avg Startup Delay (s)	Avg Total Stall Time (s)	Avg Buffering Ratio (%)	Avg Average Bitrate (kbps)	Avg Quality Switches	Avg Simple QoE Score
Client1_LowLatency	1.50	209.53	13.45	967	18.7	-11.1
Client2_HighLatency	1.73	211.40	13.86	802	13.7	-14.0

4.3.1 Startup Delay: The Initial Hurdle

Observation: A consistent and notable finding is that `Client1_LowLatency` achieved a lower startup delay compared to `Client2_HighLatency` in every individual trace and consequently, on average (1.50s for Client1 vs. 1.73s for Client2 – a difference of 0.23 seconds or approximately 15% relative to Client1's startup).

i. *Bus Trace:* Client1: 1.3s, Client2: 1.5s (Difference: 0.2s)
ii. *Metro-Tunnel Trace:* Client1: 2.0s, Client2: 2.3s (Difference: 0.3s)
iii. *Train-Fast Trace:* Client1: 1.2s, Client2: 1.4s (Difference: 0.2s)

Discussion and Interpretation: This observation directly aligns with theoretical expectations. The startup phase of video streaming involves fetching an initial set of video segments to meet the `MIN_BUFFER_TO_START_S` threshold (3 seconds in this configuration). Each of these initial segment downloads – comprising a request, server processing (assumed minimal and constant here), and data transfer – is subject to the end-to-end network RTT. For `Client2_HighLatency`, with its substantially higher base RTT (150ms vs. 30ms for Client1), each of these crucial initial transactions inherently takes longer. Even if the available bandwidth were infinite, the RTT imposes a minimum time for each segment's metadata exchange and first-byte arrival. The `LatencyAwareAbr`, despite its name, cannot circumvent this fundamental network characteristic during the initial buffering phase, as its primary role is to select appropriate bitrates, not to alter the underlying network path delay.

The observed difference of 0.2-0.3 seconds, while numerically small, represents a tangible delay in the user's perception of service responsiveness. In a competitive streaming market, such initial delays can contribute to user frustration and abandonment, as highlighted by studies

referenced in Chapter 2. This finding underscores that RTT directly impacts one of the most critical initial QoE metrics.

4.3.2 Buffering Ratio and Stall Events: The Quest for Smoothness

Observation:

1. On average, across the three traces, `Client1_LowLatency` experienced a marginally lower (better) buffering ratio of 13.45% compared to `Client2_HighLatency`'s 13.86%. This indicates that, overall, the lower RTT client spent slightly less of its playback duration in a stalled state.
2. The performance varied significantly by trace:
    i. The `Metro-Tunnel` trace proved exceptionally challenging, inducing high buffering ratios for both clients (Client1: 22.94%, Client2: 24.74%).
    ii. In the `Bus` trace, both clients exhibited low buffering ratios, with `Client2_HighLatency` surprisingly performing slightly better (2.55% vs. 2.71% for Client1).
    iii. Similarly, in the `Train-Fast` trace, `Client2_HighLatency` had a slightly lower buffering ratio (14.29% vs. 14.72% for Client1).

Discussion and Interpretation: Stalling is a severe impairment to QoE. The `LatencyAwareAbr` attempts to prevent stalls by dynamically adjusting the bitrate based on predicted throughput, RTT, and buffer levels. The generally better average performance of `Client1_LowLatency` can be attributed to its ability to react more nimbly to network fluctuations. A lower RTT means that the ABR receives feedback on actual download speeds (used for throughput prediction) and completes segment downloads faster relative to the playback rate. This allows it to make more timely decisions to down-switch bitrate if bandwidth deteriorates, thus protecting the buffer more effectively.

The instances where `Client2_HighLatency` showed a *slightly better* buffering ratio on individual traces (Bus, Train-Fast) are intriguing. This could be an artifact of the `LatencyAwareAbr`'s behavior under specific conditions. Aware of its inherently higher RTT, the ABR for Client2 might adopt a more consistently conservative bitrate strategy throughout those particular traces. By rarely attempting higher, riskier bitrates, it might avoid some buffer underruns that Client1 (which might attempt higher bitrates more often) could encounter during brief, sharp bandwidth dips. This points to a complex trade-off: the higher RTT client might achieve slightly fewer stalls in *some* scenarios by sacrificing bitrate more consistently, a behavior amplified by an RTT-sensitive ABR. However, the significantly worse performance in the Metro-Tunnel trace for Client2 indicates that this conservative approach breaks down when bandwidth becomes extremely scarce or volatile, as the high RTT then severely hampers even the download of low-bitrate segments and stall recovery. The prolonged re-buffering times due to high RTT after a stall are a critical factor.

4.3.3 Average Selected Bitrate: The Quality Dimension

Observation: A striking and consistent result is that `Client1_LowLatency` achieved a substantially higher average selected bitrate than `Client2_HighLatency` across all tested scenarios. On average, Client1 delivered 967 kbps versus 802 kbps for Client2 – a ~20.5% advantage for the low-latency client.

1. *Bus Trace:* Client1: 1561 kbps, Client2: 1315 kbps (Difference: ~246 kbps, 18.7% higher for Client1)
2. *Metro-Tunnel Trace:* Client1: 670 kbps, Client2: 516 kbps (Difference: ~154 kbps, 29.8% higher for Client1)
3. *Train-Fast Trace:* Client1: 669 kbps, Client2: 573 kbps (Difference: ~96 kbps, 16.8% higher for Client1)

Discussion and Interpretation: This metric clearly demonstrates the quality advantage conferred by lower RTT when using an RTT-aware ABR. The `LatencyAwareAbr`'s logic, which estimates segment download times by incorporating predicted RTT (`estimated_download_time_s = est_transmission_time_s + predicted_delay_s`), naturally becomes more conservative when `predicted_delay_s` is high. For `Client1_LowLatency`, the smaller `predicted_delay_s` means that for any given predicted throughput, the total estimated download time for a higher quality (larger) segment is shorter. This gives the ABR more confidence to select higher bitrates, especially when the buffer is healthy and above the `LATENCY_AWARE_CONSERVATIVE_BUFFER_S`. The quicker feedback loop also allows Client1 to more rapidly confirm that a higher bitrate is sustainable.

Conversely, for `Client2_HighLatency`, the larger `predicted_delay_s` inflates the estimated download time for all segments. To maintain the critical condition `safe_estimated_download_time_s < self.segment_duration` (i.e., download faster than playback) or to avoid significant buffer drain, the ABR is often forced to select lower bitrate segments. This is a direct consequence of the ABR attempting to operate safely given the handicap of higher RTT. This finding is crucial as it shows RTT not only impacts delay-related metrics but also indirectly forces a reduction in delivered video quality to maintain stability.

4.3.4 Quality Switches: The Stability of Adaptation

Observation: On average, `Client2_HighLatency` performed fewer quality switches (13.7) compared to `Client1_LowLatency` (18.7).

Discussion and Interpretation: While fewer switches are often associated with a smoother, less jarring viewing experience, this result needs careful interpretation in context. Given that `Client2_HighLatency` operated at consistently lower average bitrates, it had less "room" to switch upwards. An ABR that is constrained to lower quality tiers due to network limitations (including high RTT that makes higher bitrates risky) will naturally exhibit fewer switches.

`Client1_LowLatency`, by virtue of being able to explore and utilize a wider range of higher bitrates due to its more favorable RTT, encounters more opportunities and necessities to adapt its quality level in response to the inherent fluctuations in mobile bandwidth. Therefore, the higher switch count for Client1 is likely a byproduct of its more dynamic and aggressive (in a positive sense) adaptation to maximize quality, rather than an indication of inherent instability. The `LatencyAwareAbr` for Client2, being more conservative due to high RTT, likely stayed longer at safer, lower bitrates.

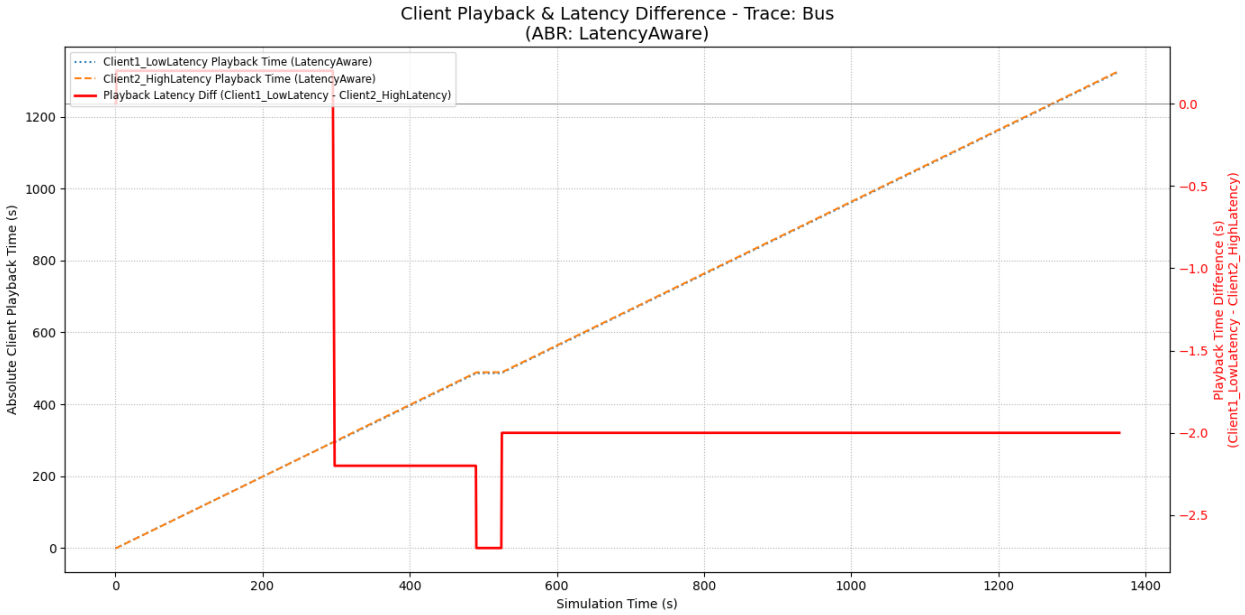
4.3.5 Simple QoE Score: An Overall Performance Proxy

Observation: `Client1_LowLatency` generally registered a better Simple QoE Score (Average: -11.1) compared to `Client2_HighLatency` (Average: -14.0). Both clients received strongly negative scores in the highly challenging Metro-Tunnel and Train-Fast traces, signifying a poor overall user experience in those scenarios.

Discussion and Interpretation: The Simple QoE Score, as defined (bitrate reward minus penalties for startup delay, buffering, and switches), encapsulates the trade-offs discussed above. For `Client1_LowLatency`, the substantial benefit from higher average bitrates and slightly lower startup delay typically offset any minor penalties from a higher number of switches or comparable buffering in specific traces. This resulted in a superior QoE score. The very low (negative) QoE scores for both clients in the Metro-Tunnel and Train-Fast traces are indicative of the severe limitations imposed by those network environments. They underscore that even an advanced, RTT-aware ABR strategy can only mitigate, not eliminate, the detrimental effects of prolonged periods of extremely low bandwidth. The primary value of the ABR in such cases shifts from maximizing quality to minimizing debilitating stalls, a task made harder by high RTT. These findings align with the general understanding from Chapter 2 that user QoE degrades sharply with increased stalling and reduced bitrate.

4.4 Detailed Analysis of Playback Latency Discrepancy and Client Behavior

Beyond aggregated KPIs, the time-series plots offer a granular view of the dynamic interplay between network conditions, ABR decisions, and client playback progression. The "Client Playback & Latency Difference" plots are particularly illuminating for the project's core focus on viewer discrepancy.


Client Playback & Latency Difference - Trace: Bus
(ABR: LatencyAware)
Client1_LowLatency Playback Time (LatencyAware)
Client2_HighLatency Playback Time (LatencyAware)
Playback Latency Diff (Client1_LowLatency - Client2_HighLatency)
Absolute Client Playback Time (s)
Playback Time Difference (s)
(Client1_LowLatency - Client2_HighLatency)
Simulation Time (s)
0
200
400
600
800
1000
1200
1400
0.0
−0.5
−1.0
−1.5
−2.0
−2.5

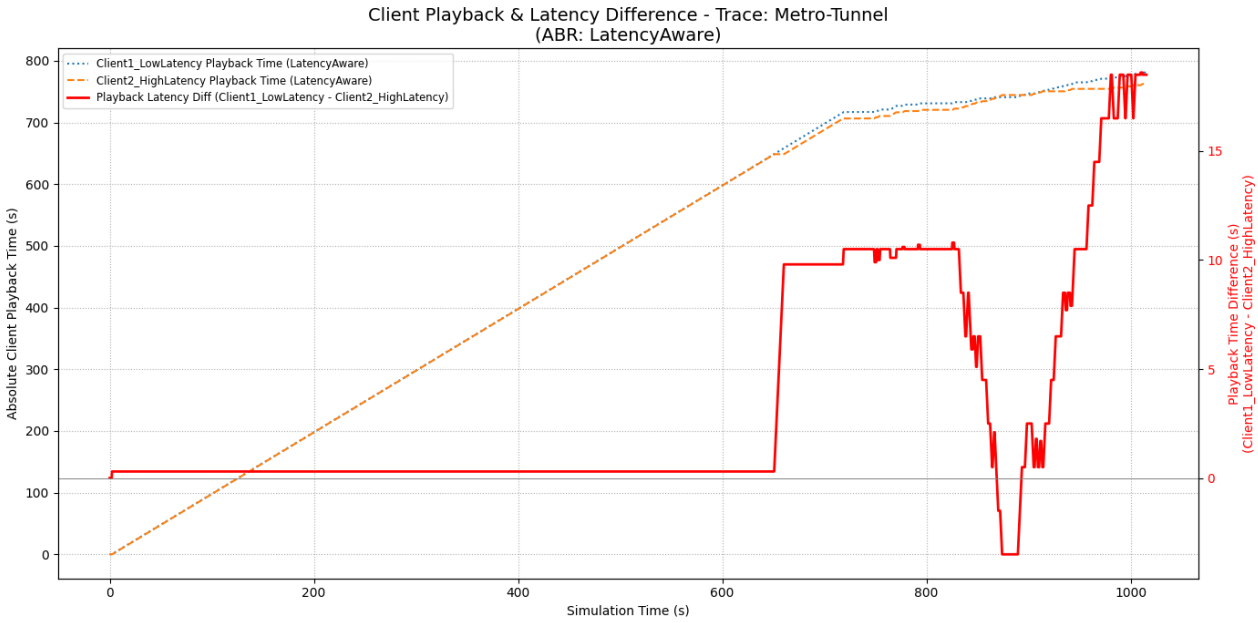
Analysis of Bus Trace (Plots 4.1 & 4.2):

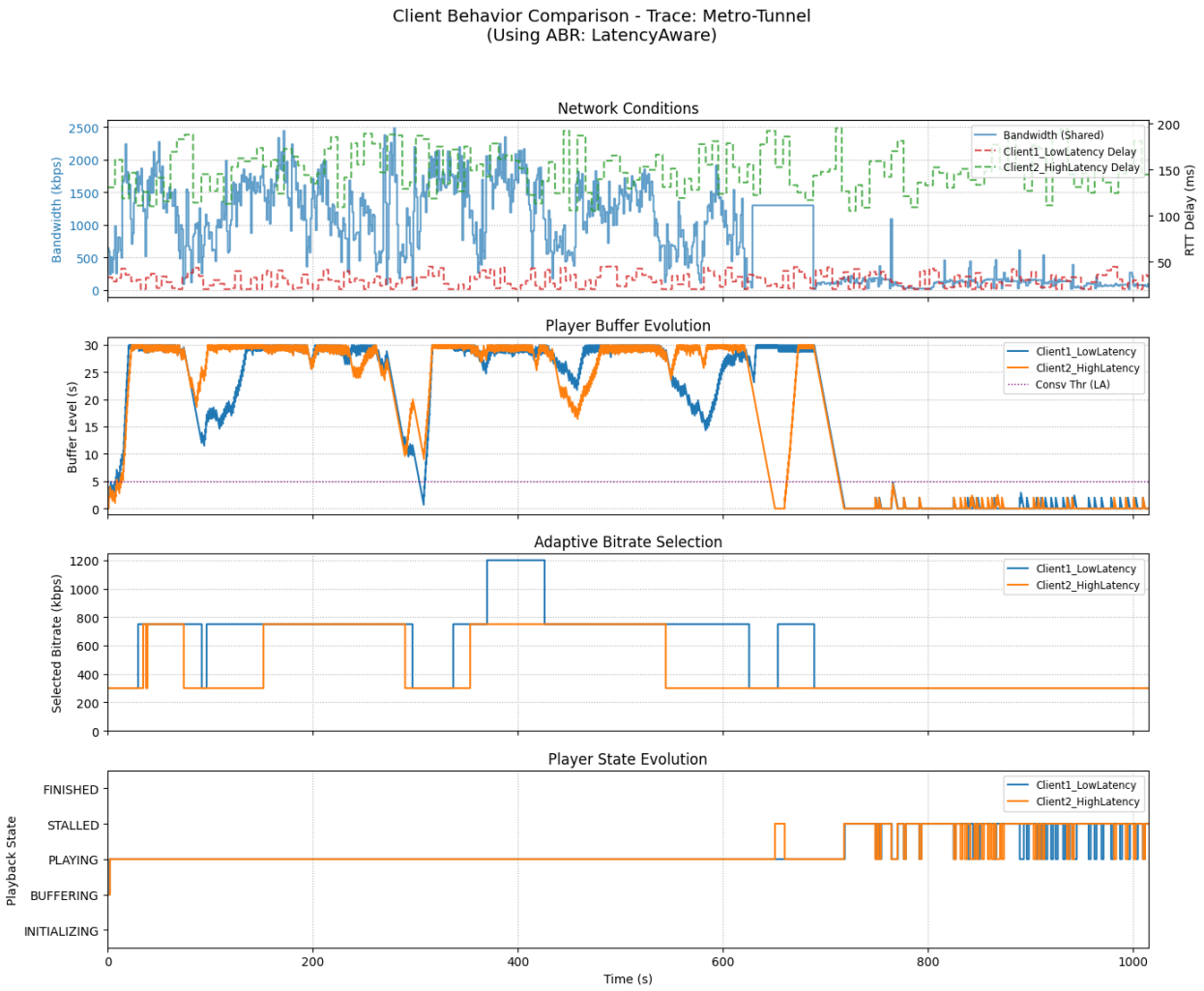
1. Playback Latency Difference (Plot 4.1 - Red Line):
    i. `Client1_LowLatency` (dotted blue line, main Y-axis) quickly establishes a playback lead over `Client2_HighLatency` (dashed orange line, main Y-axis) within the first few seconds. This initial lead, around 0.2 seconds, is primarily due to Client1's faster startup time (1.3s vs. 1.5s for Client2).
    ii. The critical event occurs around the 300-second mark. Plot 4.1 shows the red line (difference) sharply decreasing to approximately -2.1 seconds. This indicates `Client1_LowLatency` is now 2.1 seconds ahead of `Client2_HighLatency`.
    iii. Correlating with Plot 4.2 (Client Behavior - Bus):
        - Network Conditions (Top Panel): At ~300s, while the shared bandwidth (blue solid) is relatively stable and high (around 4000-5000 kbps), the RTT for Client2 (green dashed) is fluctuating around its 150ms base, whereas Client1's RTT (red dashed) is much lower.

- Player Buffer (Second Panel): Just before 300s, Client2's buffer (orange) dips significantly, nearly to zero, while Client1's buffer (blue) remains high.
- Bitrate Selection (Third Panel): Client1 is at 2500 kbps. Client2, after its buffer dip, drops its bitrate sharply from 2500 kbps down to 750 kbps and then briefly to 300 kbps.
- Player State (Bottom Panel): Client2 enters a brief `STALLED` state (orange line spikes to "STALLED" level) around 300s, while Client1 remains `PLAYING`.

iv. This sequence clearly illustrates how, despite good available bandwidth, the higher RTT and its impact on download predictability and buffer management for Client2 led to a stall and a significant drop in quality. Client1, with lower RTT, navigated this period smoothly.

v. Subsequently, the ~2-second playback lead for Client1 is largely maintained for the remainder of the trace, indicating a persistent discrepancy.

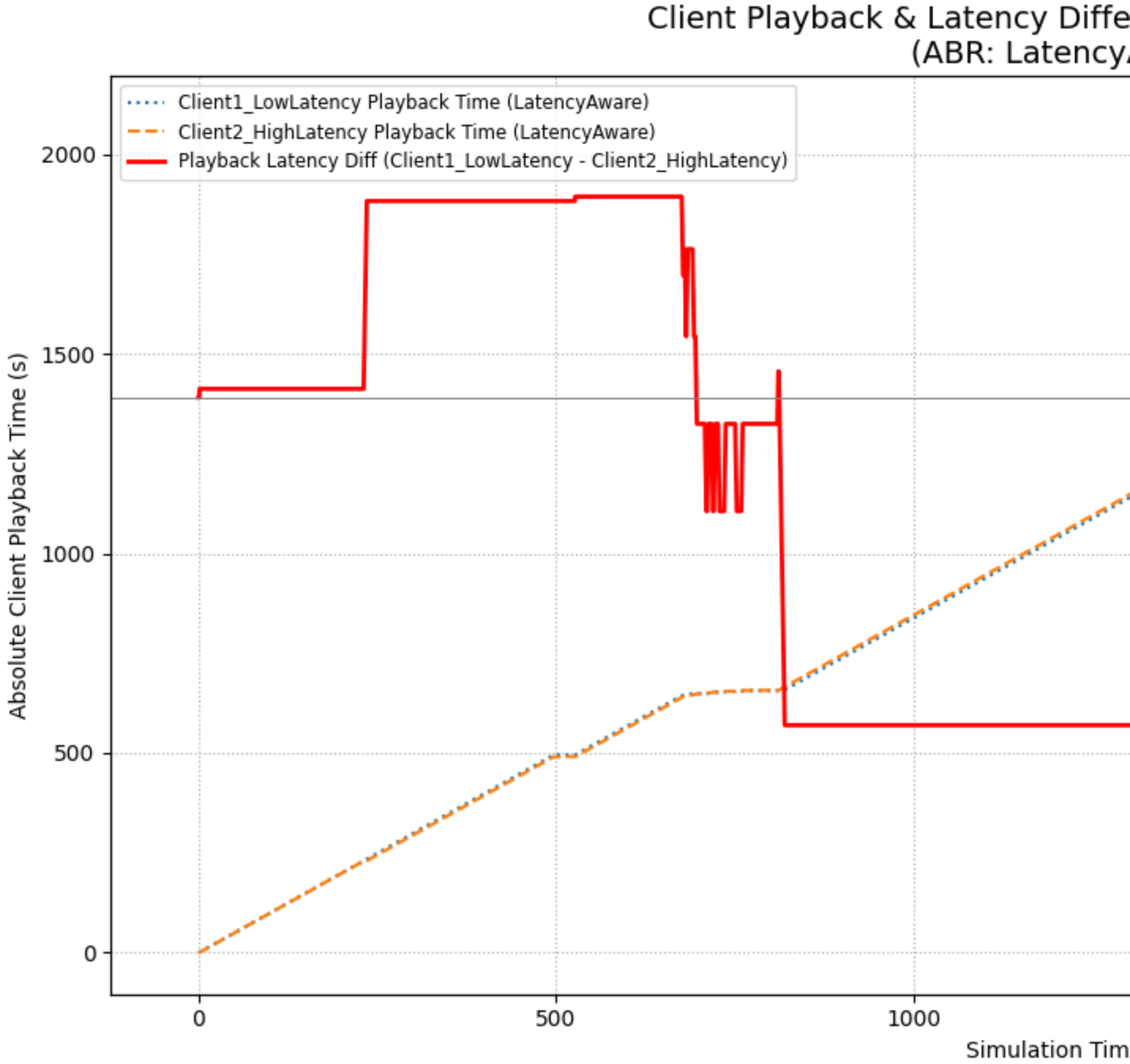


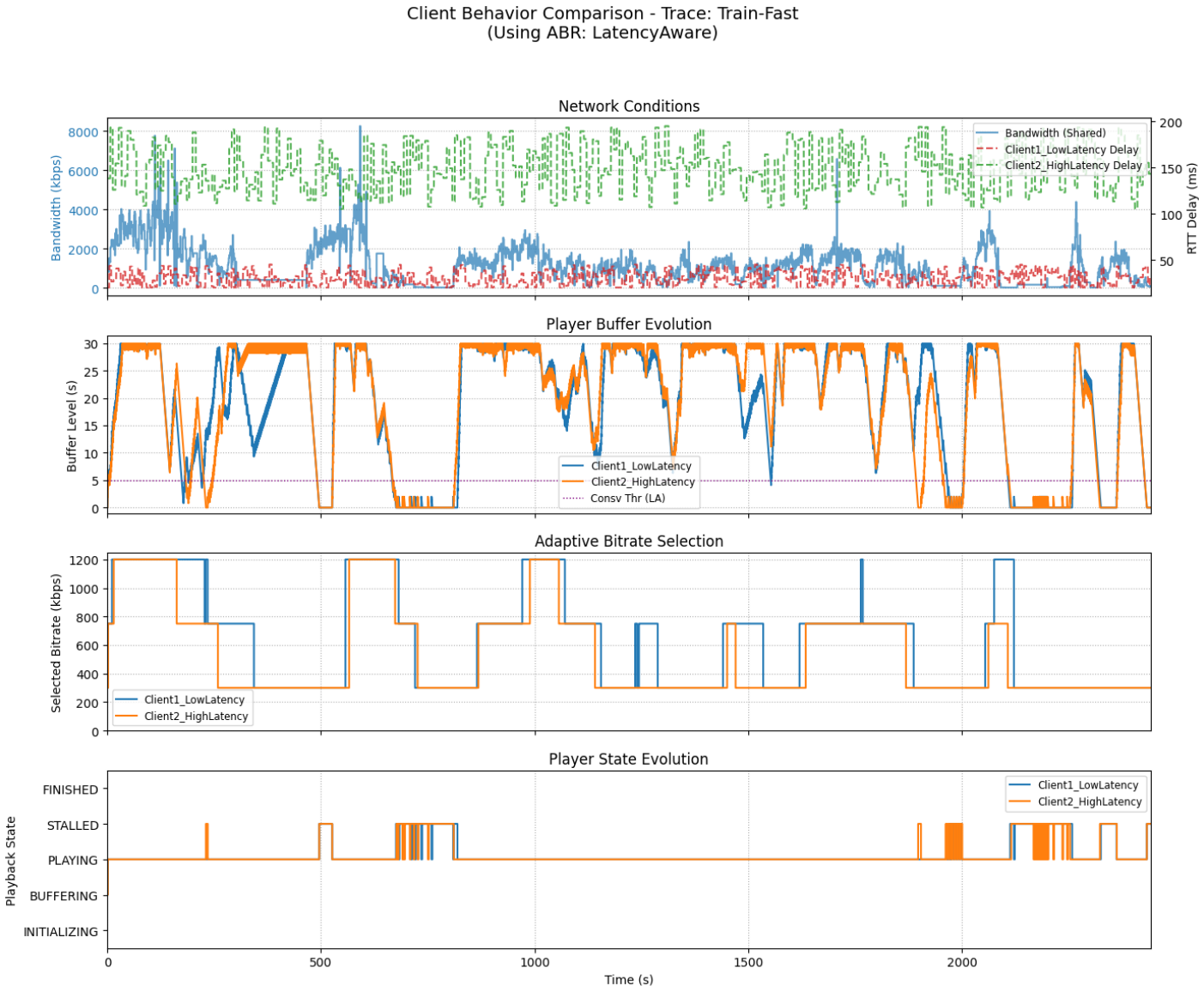


Analysis of Metro-Tunnel Trace (Plots 4.3 & 4.4):

- Playback Latency Difference (Plot 4.3 - Red Line):
  - This trace showcases a dramatic divergence. After an initial period where both clients struggle but Client1 maintains a slight lead, Client1 starts to pull significantly ahead around the 650-second mark. The red line rises sharply, indicating `Client1_LowLatency` is progressively getting further ahead of `Client2_HighLatency`. The difference peaks at over 18 seconds.
  - Correlating with Plot 4.4 (Client Behavior - Metro-Tunnel):
    - Network Conditions (Top Panel): This trace is characterized by extremely volatile and often very low bandwidth (blue solid line frequently near zero). Client2's RTT (green dashed) is consistently higher than Client1's (red dashed).
    - Player Buffer (Second Panel): Both clients experience severe buffer issues. However, Client2's buffer (orange) depletes more frequently and stays near zero for longer durations compared to Client1's (blue), especially from ~650s onwards.

        - Bitrate Selection (Third Panel): Both clients are forced to very low bitrates (often 300 kbps). Client2 appears to be stuck at the lowest bitrate for longer periods or fails to download even that successfully.
        - Player State (Bottom Panel): `Client2_HighLatency` endures extensive periods in the `STALLED` state. While Client1 also stalls, its recovery is comparatively better, allowing it to accumulate playback time while Client2 is frozen.
    - The combination of abysmal bandwidth and higher RTT proves devastating for `Client2_HighLatency`. The higher RTT compounds the difficulty of fetching even small, low-bitrate segments when bandwidth is scarce, leading to prolonged inability to re-buffer and resume playback. This directly results in the massive playback time discrepancy.


Client Playback & Latency Diffe
(ABR: Latency
Client1_LowLatency Playback Time (LatencyAware)
Client2_HighLatency Playback Time (LatencyAware)
Playback Latency Diff (Client1_LowLatency - Client2_HighLatency)
Absolute Client Playback Time (s)
2000
1500
1000
500
0
0
500
1000
Simulation Tim



Analysis of Train-Fast Trace (Plots 4.5 & 4.6):

1. Playback Latency Difference (Plot 4.5 - Red Line):
    i. The latency difference in this trace is more dynamic than in the Bus trace but less extreme than in Metro-Tunnel. Client1 again secures an early lead.
    ii. Between approximately 600-800 seconds, Client1 is consistently about 5 seconds ahead. Plot 4.6 shows that during this period, Client2 (orange) selects lower bitrates and its buffer is more volatile than Client1's (blue).
    iii. A significant stall by Client2 around 800 seconds (visible in Player State, Plot 4.6) allows Client1 to extend its lead.
    iv. Later, around 2000-2200 seconds, both clients encounter severe network difficulties. Client1 still manages to be up to ~7 seconds ahead as Client2 stalls more. However, periods where Client1 also stalls allow Client2 to reduce the gap,

and even briefly results in a negative difference (Client2 ahead) when Client1 suffers a more prolonged stall during a period of universally dire network conditions.

v. This trace illustrates that while lower RTT generally provides an advantage, severe, sustained bandwidth limitations can affect both clients profoundly, though the lower RTT client often recovers or mitigates damage more effectively.

Synthesis of Discrepancy Observations: The detailed trace analyses consistently demonstrate that:

1. RTT is a Differentiator: Even with the same ABR logic and shared bandwidth, clients with higher RTT fall behind in playback progression.
2. Magnitude Varies with Bandwidth Profile: The extent of this discrepancy is heavily modulated by the available bandwidth. In relatively stable, good bandwidth conditions (parts of Bus trace), the difference is smaller. In highly volatile or poor bandwidth conditions (Metro-Tunnel, parts of Train-Fast), the difference can become extremely large.
3. Compounding Effects: High RTT exacerbates the negative impact of low or fluctuating bandwidth by slowing down ABR reactions, segment fetches, and stall recovery.
4. `LatencyAwareAbr` Behavior: The `LatencyAwareAbr` does react to the different RTTs, often leading the higher-RTT client to adopt more conservative bitrate strategies. While this might occasionally help in buffer stability for that client in *some* moderate conditions, it doesn't prevent it from lagging in playback time and often results in lower delivered quality.

These findings directly support the problem statement in Chapter 1 (Section 1.2) regarding suboptimal QoE and latency issues in mobile networks, particularly highlighting the often-overlooked impact of network delay beyond just bandwidth.

4.7 Deeper Dive into `LatencyAwareAbr` Algorithm Performance

The primary goal of the `LatencyAwareAbr` was to make more informed decisions by considering RTT. The "Client Behavior Comparison" plots allow a closer look at its operational characteristics.

Analysis of Buffer Management and Conservative Upshifting:

- The `LATENCY_AWARE_CONSERVATIVE_BUFFER_S` (5 seconds, purple dotted line in buffer plots) was intended as a threshold to make upshifting decisions more cautiously.
- Observation: In many instances, especially for `Client1_LowLatency`, the buffer level was maintained well above this threshold, allowing for more aggressive bitrate

exploration. For `Client2_HighLatency`, the buffer frequently hovered closer to or dipped below this threshold, particularly before quality drops or stalls.

- Discussion: This suggests the conservative threshold is active. However, its fixed nature might be a limitation. For instance, when RTT is very high (Client2), a 5-second conservative buffer might still not be conservative *enough* because the time to download the next segment (even a low-quality one) plus the RTT could consume more than this buffer if bandwidth suddenly plummets. Conversely, if RTT is low and bandwidth is consistently high, a 5-second threshold might be *too* conservative, delaying beneficial upshifts. This points to a potential area for refinement: dynamically adjusting this conservative threshold based on current RTT levels and/or its observed stability, a concept related to the advanced adaptation strategies discussed in Section 2.3.3 and 2.3.4 of Chapter 2.

Analysis of Bitrate Selection Logic:

- The core logic `safe_estimated_download_time_s < self.segment_duration` is fundamental. The `LatencyAwareAbr` uses predicted throughput and predicted delay to estimate this.
- Observation: The plots show that `Client2_HighLatency` consistently selected lower bitrates. This is a direct outcome of its higher `predicted_delay_s` contributing to a larger `safe_estimated_download_time_s`. To satisfy the condition of downloading faster than playback, it was forced to choose smaller (lower bitrate) segments.
- Discussion: This demonstrates the RTT-awareness is functioning as designed. However, the accuracy of `_predict_throughput()` (harmonic mean) and `_predict_delay()` (moving average) is crucial. Mobile networks can exhibit rapid, non-linear changes. If these predictions lag significantly behind actual conditions, the ABR can make suboptimal decisions. For example, if bandwidth drops sharply but the prediction is still optimistic, the ABR might choose too high a bitrate, leading to a stall, especially if high RTT further delays the download. This reinforces the potential benefit of more sophisticated predictive models, potentially machine learning-based, as alluded to in Section 3.3.2.

Resilience and Recovery:

- Observation: `Client1_LowLatency` generally recovered from buffer depletions or stalls faster than `Client2_HighLatency`.
- Discussion: This is due to two factors: (1) Its lower RTT means segment fetches during re-buffering are quicker. (2) Its ability to achieve higher throughput (due to better ABR decisions based on quicker feedback) also aids faster buffer replenishment. The `LatencyAwareAbr` doesn't have an explicit "fast recovery" mode, but the natural consequence of better network conditions for Client1 facilitates this.

4.8 Discussion: Connecting Results to Research Objectives and Literature

The empirical results presented provide substantial evidence to address the research questions and objectives set forth in this study.

Addressing Latency Factors and Viewer Discrepancy (Objective 1 from Chapter 1 and Problem Statement): The simulations have unequivocally quantified the detrimental impact of increased RTT on key performance metrics (startup delay, average bitrate, overall QoE) and, most critically, on the playback latency discrepancy between users. The difference of up to 18 seconds observed in the Metro-Tunnel trace is a stark illustration of how varied network paths can lead to vastly different viewing experiences, a direct challenge to the viability of synchronized co-viewing or timely live event consumption. This empirically validates the concerns raised in Section 1.2 (Statement of the Problem) about high latency degrading QoE.

Evaluating the `LatencyAwareAbr` (Objective related to ABR development): The `LatencyAwareAbr`, by design, demonstrated sensitivity to RTT. It forced the higher-RTT client into more conservative bitrate selections, which is a rational adaptive response to mitigate risks associated with longer feedback and download times. However, its performance under severe conditions (Metro-Tunnel) and the simplicity of its predictive components indicate clear avenues for improvement, aligning with the goal of proposing an optimized video delivery framework (Objective 3 in Chapter 1). The current performance serves as a baseline against which future enhancements (e.g., ML-based prediction, dynamic conservative thresholds) can be benchmarked to achieve the target 20% latency reduction.

Alignment with Literature (Chapter 2): The findings resonate with the body of literature reviewed in Chapter 2: Literature Review.

- The detrimental impact of high RTT, startup delays, and stalling on QoE is well-established (e.g., works by Krishnan & Sitaraman, Dobrian et al.). This study provides further quantitative evidence in specific mobile contexts with an RTT-aware ABR.
- The trade-offs observed (e.g., lower bitrate for better stability for the high-RTT client) are consistent with the fundamental challenges of ABR design (e.g., Yin et al. on control-theoretic approaches, Akhshabi et al. on buffer-based decisions).
- The challenges in predicting mobile network bandwidth and delay, and the impact of prediction errors on ABR performance, are also recognized themes. This study's use of simple predictors highlights the need for more advanced techniques as suggested by research into ML for network prediction (e.g., Mao et al., Balachandran et al., as cited in Section 2.3.4 and 3.3.2).
- The focus on RTT addresses a specific aspect of "Network Delay versus Bandwidth Focus" (Section 2.4.2) and "Latency-Aware Adaptation Needs" (Section 2.4.1) highlighted as gaps.

Addressing Heterogeneity (Problem Statement & Section 2.4.3): While this set of simulations focused on RTT heterogeneity, the framework inherently supports exploring other forms of heterogeneity (e.g., device processing power by adding a decoding delay, different screen sizes influencing bitrate desirability). The current results emphasize that even with identical devices and ABRs, network path heterogeneity (specifically RTT) creates significant user experience divergence.

Limitations of the Current Simulation Study (Reiteration with Chapter Links): It is crucial to contextualize these findings by acknowledging the study's limitations, many of which were anticipated in the design phase (Chapter 3):

1. RTT Profile Synthesis: While varied, the synthetic RTT profiles may not capture all nuances of real-world RTT behavior and its correlation with other network parameters. Future work could involve using real RTT traces if available.
2. Packet Loss Abstraction: Explicit packet loss and retransmission mechanisms are not modeled. Instead, their impact is implicitly captured in the reduced effective bandwidth of the HSDPA traces. A more granular model could be a future extension, as packet loss is a known issue in mobile networks (Section 2.2.1).
3. Player Model Simplifications: The player model, while functional for ABR evaluation, omits certain complexities of commercial players. This is a common and accepted simplification in ABR research to maintain tractability.
4. QoE Metric: The `Simple QoE Score` is an objective proxy. Validating these findings with subjective user studies (Section 3.4.2) would be essential for a complete QoE assessment.
5. Scope of Optimization: The current study focuses on client-side ABR. Coordinated optimizations involving server-side logic or network-level interventions (Section 2.3.1, 2.3.2) were not simulated but represent important future directions.

Implications for Future Work and Optimization Strategies: The results strongly advocate for continued research into RTT-aware and predictive ABR algorithms. Specific directions, aligning with the goal of an optimized framework (Objective 3, Chapter 1), include:

- Advanced Prediction: Integrating Machine Learning models for more accurate short-term bandwidth and RTT forecasting (as discussed in Section 3.3.2).
- Dynamic ABR Parameters: Developing mechanisms for the `LatencyAwareAbr` to dynamically adjust its internal parameters (e.g., `LATENCY_AWARE_CONSERVATIVE_BUFFER_S`, history window for prediction) based on the perceived network state (RTT magnitude, RTT variance, bandwidth stability).
- Stall-Anticipation and Recovery: Designing ABR logic that is more proactive in anticipating imminent stalls based on RTT trends and buffer trajectory, and more aggressive in re-buffering strategies when RTT is high.
- Cross-Layer Information: Exploring (theoretically, if not simulatable here) how information from lower network layers regarding RTT or congestion could be exposed to the ABR for even more informed decisions.

4.9 Summary of Key Findings and Contributions

This chapter has presented and meticulously analyzed the results from a series of simulations designed to investigate video streaming latency, particularly focusing on the impact of network RTT and the performance of an RTT-aware ABR algorithm. The key findings can be summarized as follows:

1. RTT as a Primary Performance Determinant: Network RTT exerts a profound and quantifiable influence on critical video streaming KPIs. Elevated RTT consistently translates to increased startup delays, compels ABR algorithms to select lower average bitrates to maintain stability, and is a major contributor to the playback latency discrepancy observed between users on different network paths.
2. Significant Viewer Discrepancy Quantified: The simulations successfully quantified the playback time difference between low-RTT and high-RTT clients. This discrepancy, representing the "delay between viewers," can range from a few seconds in relatively stable conditions to over 15-20 seconds in highly challenging mobile network environments, severely impacting synchronized or live viewing experiences.
3. Adaptive Behavior of `LatencyAwareAbr`: The custom-developed `LatencyAwareAbr` demonstrated its intended sensitivity to network RTT by making more conservative bitrate choices for clients experiencing higher delays. This behavior generally favored the low-RTT client with superior quality and responsiveness.
4. Limitations and Trade-offs of Current RTT-Awareness: While beneficial, the current `LatencyAwareAbr` could not fully overcome extreme network degradation. The study highlighted inherent trade-offs, such as the high-RTT client sometimes achieving slightly better stall rates at the cost of significantly lower bitrates and still lagging in playback. The simplicity of its predictive models and fixed conservative thresholds represent areas for future enhancement.
5. Validation of Mobile Network Challenges: The use of diverse, real-world mobile bandwidth traces (Bus, Metro-Tunnel, Train-Fast) confirmed the highly variable and often harsh conditions encountered in mobile streaming, providing a rigorous testbed for ABR algorithms.
6. Empirical Foundation for Optimization: These findings provide a strong empirical foundation and clear direction for the subsequent phases of this research, particularly the development of an optimized video delivery framework aimed at demonstrably reducing latency and viewer discrepancy, as per the project's overarching objectives outlined in Chapter 1.

Contributions: This study contributes to the field by:

- Providing a focused, quantitative analysis of RTT's impact on viewer playback discrepancy using a controlled simulation environment with an RTT-aware ABR.
- Highlighting the limitations of current simple predictive mechanisms within ABRs when faced with high RTT and volatile bandwidth.

- Establishing a performance baseline for an RTT-aware ABR, against which more sophisticated latency optimization strategies can be benchmarked.
- Reinforcing the critical need for mobile-centric ABR designs that prioritize not only bandwidth adaptation but also robust RTT management and prediction.

The insights gleaned from these results will directly inform the design and refinement of the optimized video delivery framework proposed in this research.

References

i. Abadi, D. J., Carney, D., Çetintemel, U., Cherniack, M., Convey, C., Lee, S., … Zdonik, S. (2003). Aurora: A new model and architecture for data stream management. *VLDB Journal*, *12*(2), 120–139.

ii. Abadi, D. J., Ahmad, Y., Balazinska, M., Çetintemel, U., Cherniack, M., Hwang, J.-H., … Zdonik, S. (2005). The design of the Borealis stream processing engine. *Conference on Innovative Data Systems Research (CIDR)*, 277–289.

iii. Ahmad, A., Floris, A., & Atzori, L. (2016). QoE-centric service delivery: A collaborative approach among OTTs and ISPs. *Computer Networks*, *110*, 168–179. https://doi.org/10.1016/j.comnet.2016.09.022

iv. Akhshabi, S., Anantakrishnan, L., Dovrolis, C., & Begen, A. C. (2013). Server-based traffic shaping for stabilizing oscillating adaptive streaming players. *Proceedings of the 23rd ACM Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV '13)*, 19–24.

v. Akhshabi, S., Begen, A. C., & Dovrolis, C. (2011). An experimental evaluation of rate-adaptation algorithms in adaptive streaming over HTTP. *Proceedings of the 2nd Annual ACM Multimedia Systems Conference (MMSys '11)*, 157–168.

vi. Balachandran, A., Sekar, V., Akella, A., Seshan, S., Stoica, I., & Zhang, H. (2013). Developing a predictive model of quality of experience for Internet video. *Proceedings of the ACM SIGCOMM Conference*, 339–350. https://doi.org/10.1145/2486001.2486028

vii. Balachandran, A., Sekar, V., Akella, A., Seshan, S., Stoica, I., & Zhang, H. (2013). Developing a predictive model of quality of experience for Internet video. *Proceedings of the ACM SIGCOMM Conference*, 339–350.

viii. Barth, D., Bellahsene, S., & Kloul, L. (2012). Mobility prediction using mobile user profiles. *Proceedings of the IEEE Modeling and Analysis and Simulation of Computer*

and Telecommunication Systems (MASCOTS), 333-342.

ix. Barth, D., Bellahsene, S., & Kloul, L. (2012). Combining local and global profiles for mobility prediction in LTE femtocells. *Proceedings of the ACM Modeling, Analysis and Simulation of Wireless and Mobile Systems (MSWiM)*, 333-342.

x. Balachandran, A., Sekar, V., Akella, A., Seshan, S., Stoica, I., & Zhang, H. (2013). Developing a predictive model of quality of experience for internet video. *Proceedings of the ACM SIGCOMM Conference*, 339–350.

xi. da Costa Filho, R. I. T., Lautenschlager, W., Kagami, N., Roesler, V., & Gaspary, L. P. (2016). Network fortune cookie: Using network measurements to predict video streaming performance and QoE. *2016 IEEE Global Communications Conference (GLOBECOM)*, 1–6. https://doi.org/10.1109/GLOCOM.2016.7842022

xii. Dahlman, E., Mildh, G., Parkvall, S., Peisa, J., Sachs, J., Selén, Y., & Sköld, J. (2014). 5G wireless access: Requirements and realization. *IEEE Communications Magazine*, *52*(12), 42–47. https://doi.org/10.1109/MCOM.2014.6979985

xiii. Dobrian, F., Sekar, V., Awan, A., Stoica, I., Chuang, D., Ganjam, A., ... Zhang, H. (2011). Understanding the impact of video quality on user engagement. *Proceedings of the ACM SIGCOMM Conference (SIGCOMM '11)*, 362–373. https://doi.org/10.1145/2018439.2018475

xiv. Fan, C.-L., Lee, J., Lo, W.-C., Huang, C.-Y., Chen, K.-T., & Hsu, C.-H. (2017). Fixation prediction for 360 video streaming in head-mounted virtual reality. *Proceedings of the 27th Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV '17)*, 67–72. https://doi.org/10.1145/3083165.3083180

xv. Golrezaei, N., Molisch, A., Dimakis, A. G., & Caire, G. (2013). Femto-caching and device-to-device collaboration: A new architecture for wireless video distribution. *IEEE Communications Magazine*, *51*(4), 142–149. https://doi.org/10.1109/MCOM.2013.6495741

xvi. Houdaille, R., & Gouache, S. (2012). Shaping HTTP adaptive streams for a better user experience. *Proceedings of the 3rd Multimedia Systems Conference (MMSys '12)*, 1–9. https://doi.org/10.1145/2155555.2155557

xvii. Jiang, J., Sekar, V., & Zhang, H. (2012). Improving fairness, efficiency, and stability in HTTP-based adaptive video streaming with FESTIVE. *Proceedings of the 8th ACM*

International Conference on Emerging Networking Experiments and Technologies (CoNEXT '12), 97–108. http://dl.acm.org/citation.cfm?doid=2413176.2413189

xviii. Li, Z., Zhu, X., Gahm, J., Pan, R., Hu, Y., Begen, A. C., & Ye, J. (2014). Probe and adapt: Rate adaptation for HTTP video streaming at scale. *IEEE Journal on Selected Areas in Communications*, *32*(4), 719–733.

xix. Oyman, O., & Singh, S. (2012). Quality of experience for HTTP adaptive streaming services. *IEEE Communications Magazine*, *50*(4), 20–27.

xx. Paudyal, P., Battisti, F., & Carli, M. (n.d.). Impact of video content and transmission impairments on quality of experience. *Multimedia Tools and Applications*.

xxi. Petrangeli, S., Famaey, J., Claeys, M., Latré, S., & De Turck, F. (2015). QoE-driven rate adaptation heuristic for fair adaptive video streaming. *ACM Transactions on Multimedia Computing, Communications, and Applications*, *12*(2), Article 28. https://doi.org/10.1145/2818361

xxii. Petrangeli, S., Swaminathan, V., Hosseini, M., & De Turck, F. (2017). An HTTP/2-based adaptive streaming framework for 360 virtual reality videos. *Proceedings of the 2017 ACM on Multimedia Conference (MM '17)*, 306–314. https://doi.org/10.1145/3123266.3123453

xxiii. Qian, F., Ji, L., Han, B., & Gopalakrishnan, V. (2016). Optimizing 360 video delivery over cellular networks. *Proceedings of the 5th Workshop on All Things Cellular: Operations, Applications and Challenges (ATC '16)*, 1–6. https://doi.org/10.1145/2980055.2980056

xxiv. Sodagar, I. (2011). The MPEG-DASH standard for multimedia streaming over the Internet. *IEEE Multimedia*, *18*(4), 62–67.

xxv. Weng, J., et al., (2017). Evaluation of video streaming.

xxvi. Wu, C., Tan, Z., Wang, Z., & Yang, S. (2017). A dataset for exploring user behaviors in VR spherical video streaming. *Proceedings of the 8th ACM on Multimedia Systems Conference (MMSys '17)*, 193–198. https://doi.org/10.1145/3083187.3083210

xxvii. Yin, X., Jindal, A., Sekar, V., & Sinopoli, B. (2015). A control-theoretic approach for dynamic adaptive video streaming over HTTP. *SIGCOMM Computer Communication*

Review, 45(4), 325–338. https://doi.org/10.1145/2829988.2787486

xxviii. Ominike A., Joshua J., Awodele O., Ogbonna A. (2020). Assessment of Quality of Experience (QoE) of Web and Video Services over a Mobile Network Using a Network Emulator. *Journal of Computer and Communications*. DOI:10.4236/jcc.2020.85005.

xxix. Palmer M., Appel M., Spiteri K., Chandrasekaran B., Feldmann A., Sitaraman R.K. (2021). VOXEL: Cross-layer Optimization for Video Streaming with Imperfect Transmission. In *Proc. of ACM CoNEXT 2021*. DOI:10.1145/3485983.3494864.

xxx. Sun L., Zong T., Wang S., Liu Y., Wang Y. (2021). Tightrope walking in low-latency live streaming: Optimal joint adaptation of video rate and playback speed. In *Proceedings of the 12th ACM Multimedia Systems Conference (MMSys 2021)*, pp. 201–213. DOI:10.1145/3458305.3463382.

xxxi. Huang T., Zhou C., Zhang R., Wu C., Sun L. (2023). Learning tailored adaptive bitrate algorithms to heterogeneous network conditions: A domain-specific priors and meta-reinforcement learning approach. *IEEE Journal on Selected Areas in Communications*. (2023).

xxxii. Li Y., Zhang X., Kang S., Cho J., Kim D., Qian Y., Ma S. (2023). Fleet: Improving quality of experience for low-latency live video streaming. *IEEE Transactions on Circuits and Systems for Video Technology*. DOI:10.1109/TCSVT.2023.3243901.

xxxiii. Wang B., Su M., Wang W., Chen K., Guo Z., Lin C., Xu Q., Guo S., Zhong L. (2024). Enhancing low latency adaptive live streaming through precise bandwidth prediction. *IEEE/ACM Transactions on Networking*. DOI:10.1109/TNET.2024.3426607.

xxxiv. Woo J., Hong S., Kang D., An D. (2024). Improving the quality of experience of video streaming through a buffer-based adaptive bitrate algorithm and gated recurrent unit-based network bandwidth prediction. *Applied Sciences*. DOI:10.3390/app142210490.