Chapter 3: Methodology

This project employs a simulation-based, data-driven approach to comprehensively investigate and optimize video streaming latency, with a particular focus on client-side Adaptive Bitrate (ABR) strategies in mobile network environments. This approach combines the utilization of real-world mobile network bandwidth data with controlled experimental setups executed within a custom-developed simulation environment. This methodology facilitates the development and rigorous evaluation of a novel ABR algorithm designed to be sensitive to network delay. The multifaceted nature of this approach is essential due to the complexity of video streaming latency, which is influenced by multiple interacting factors. It allows for robust validation of proposed algorithmic solutions through the systematic collection and analysis of quantitative performance data, with the simulation environment enabling controlled manipulation of key network variables like Round-Trip Time (RTT) while leveraging the ecological validity of empirical bandwidth traces.

3.1 Phases of the Research

The research is structured into three principal phases, ensuring a logical progression from understanding the problem space to developing and evaluating solutions:

3.1.1 Data Acquisition and Environment Setup

This initial phase focuses on acquiring the necessary input data and establishing the simulation environment.

1. **Bandwidth Trace Acquisition:** Real-world mobile network bandwidth traces (e.g., HSDPA traces from the UMass SIGCOMM'09 and MMSys'13 repositories) are collected. These traces provide time-series data of achieved throughput, reflecting the dynamic capacity of mobile networks under various conditions (e.g., urban bus routes, metro tunnels, train lines).
2. **RTT Profile Definition:** Synthetic Round-Trip Time (RTT) profiles are defined to represent diverse network path delay characteristics. These profiles include parameters for base RTT and percentage of fluctuation, allowing for the simulation of clients with varying network responsiveness (e.g., 'LowLatency' vs. 'HighLatency' profiles).
3. **Simulation Environment Development:** A custom simulation environment is implemented in Python. This environment includes:
    i. A model of a segmented video stream with multiple selectable bitrate levels.
    ii. A simulated video player capable of requesting segments, managing a playback buffer, and implementing ABR logic.
    iii. The ability to process input bandwidth traces and apply synthetic RTT profiles to segment download calculations.

iv. Comprehensive logging capabilities for network conditions, player states, ABR decisions, and key performance metrics.

3.1.2 Algorithm Development and Implementation

This phase concentrates on the design and implementation of the primary ABR algorithm under investigation:

1. **`LatencyAwareAbr` Development:** A novel ABR algorithm, termed `LatencyAwareAbr`, is developed. Its core design principle is the explicit consideration of network delay (RTT) in conjunction with predicted bandwidth and buffer status to make bitrate adaptation decisions.
2. **Predictive Heuristics:** The `LatencyAwareAbr` incorporates predictive techniques to estimate future network conditions. Specifically, it utilizes a harmonic mean of recent bandwidth measurements for short-term throughput forecasting and a moving average of recent RTT measurements for delay forecasting, operating over a defined historical window.
3. **Conservative Adaptation Logic:** The algorithm includes logic for conservative upshifting, such as requiring the player buffer to exceed a specific threshold (`LATENCY_AWARE_CONSERVATIVE_BUFFER_S`) before considering an increase in video quality, aiming to balance quality maximization with playback stability.
4. **Implementation:** The `LatencyAwareAbr` is implemented as a modular component within the Python simulation environment, allowing for its evaluation and comparison. *(A baseline algorithm, such as a simple buffer-based ABR, may also be implemented for comparative benchmarking).*

3.1.3 Simulation, Evaluation, and Analysis

This phase involves executing the simulations and analyzing the generated data:

1. **Simulation Execution:** A series of simulation experiments are conducted. Each experiment typically involves:
   i. Selecting a specific bandwidth trace.
   ii. Assigning defined RTT profiles to one or more simulated clients.
   iii. Running the simulation with the chosen ABR algorithm(s) (e.g., `LatencyAwareAbr`).
2. **Data Logging:** During each simulation run, detailed time-series data is logged, including instantaneous bandwidth, applied RTT, player buffer level, selected bitrate, player state (initializing, buffering, playing, stalled), and cumulative playback time.
3. **Performance Metric Calculation:** Upon completion of each simulation, a suite of Key Performance Indicators (KPIs) is calculated from the logged data. These include startup delay, total stall time, buffering ratio, average selected bitrate, number of quality switches, and a composite QoE score. A critical derived metric is the **playback latency discrepancy** between concurrently simulated clients.

4. **Results Analysis and Discussion:** The collected KPIs and time-series data are analyzed to:
    i. Quantify the impact of RTT on individual client performance and on the playback discrepancy between clients.
    ii. Evaluate the effectiveness of the `LatencyAwareAbr` in managing latency and QoE under diverse conditions.
    iii. Identify strengths, weaknesses, and areas for refinement in the ABR algorithm.
    iv. Draw conclusions regarding optimal strategies for latency reduction in mobile video streaming. Visualization of time-series data and statistical summaries of KPIs are key components of this analysis.

3.2 Simulation Environment and Data Sources

The core of this research methodology relies on a custom-developed simulation environment designed to model the adaptive video streaming process under controlled yet realistic network conditions.

3.2.1 Simulation Framework

The simulation framework, implemented in Python, provides the controlled experimental setup. Key components include:

1. **Video Source Model:** Represents a video asset pre-segmented into fixed-duration chunks (e.g., 2 seconds), each available at multiple predefined bitrate levels (e.g., 300kbps to 6000kbps).
2. **Simulated Player Model:** This component emulates the client-side video player. It is responsible for:
    i. Requesting video segments one by one.
    ii. Implementing the ABR logic (e.g., `LatencyAwareAbr`) to select the appropriate bitrate for the next segment based on its perception of network conditions and buffer state.
    iii. Managing a virtual playback buffer with defined minimum (for startup) and maximum capacities.
    iv. Simulating video playback by depleting the buffer at the playback rate of the currently playing segment.
    v. Tracking its internal state (initializing, buffering, playing, stalled) and various performance metrics.
3. **Network Model:** This component interfaces the player with the video source and applies network conditions:
    i. **Bandwidth Input:** Reads time-series bandwidth data from real-world mobile network traces (detailed in 3.2.2). The simulation progresses in discrete time steps, and the available bandwidth for segment downloads is determined by the trace value at the current simulation time.

    ii. **RTT/Delay Application:** Applies a synthetically generated RTT to each segment download. The RTT for a given client can be configured with a base value and a fluctuation model, allowing for the simulation of different network path delay characteristics. The segment download time is calculated as: `TransmissionTime (SegmentSize / Bandwidth) + RTT`.

4. **Logging and Metrics Module:** Records all relevant parameters and events throughout the simulation (e.g., chosen bitrates, buffer levels, stalls, playback time) for post-simulation analysis and KPI calculation.

3.2.2 Data Sources for Simulation Input

Two primary types of data feed into the simulation:

1. **Real-World Mobile Bandwidth Traces:**
    i. **Source:** Publicly available datasets of HSDPA mobile network bandwidth measurements, such as those provided by UMass for the SIGCOMM'09 and MMSys'13 conferences (Mao et al., 2017; Akhshabi et al., 2011). These traces were collected during real-world mobile usage scenarios (e.g., on buses, trains, metro systems).
    ii. **Format:** Typically, these traces provide timestamped entries indicating the amount of data transferred over a short measurement interval. This data is processed by the simulator to create a time-series of available bandwidth (kbps).
    iii. **Significance:** Using these traces injects realistic bandwidth variability and characteristics of mobile networks (e.g., fluctuations, handovers, periods of low connectivity) into the simulation, enhancing the ecological validity of the ABR performance evaluation.
2. **Synthetic Round-Trip Time (RTT) Profiles:**
    i. **Generation:** RTT profiles are generated synthetically within the simulation environment. This allows for precise control over the delay characteristics experienced by simulated clients.
    ii. **Parameters:** Each profile is typically defined by a `base_delay_ms`, a `delay_pattern` (e.g., 'fluctuating', 'stable'), a `delay_fluctuation_percent`, and a `delay_change_interval_s_base`. This enables the creation of distinct client network path conditions, such as a low-RTT client versus a high-RTT client.
    iii. **Significance:** Synthetic RTT generation permits the systematic investigation of RTT's isolated impact on streaming performance and ABR behavior, a key focus of this research. It allows for controlled comparisons between clients differing only in their RTT characteristics while experiencing the same bandwidth conditions.

By combining real bandwidth traces with controlled synthetic RTT profiles, the simulation aims to strike a balance between realism and experimental control, providing a robust platform for analyzing latency optimization strategies.

3.3 Algorithm Under Investigation: `LatencyAwareAbr`

The primary algorithm developed and evaluated in this research is `LatencyAwareAbr`. This section details its objectives, core logic, and implementation aspects within the simulation framework.

3.3.1 Algorithm Objectives

The `LatencyAwareAbr` is designed with the following primary objectives, reflecting the overall goals of this research:

1. **Minimize Perceived Latency:** This includes reducing startup delay, minimizing the frequency and duration of stall events (rebuffering), and lessening the playback progression lag relative to other viewers or an ideal timeline.
2. **Optimize Buffer Management:** To maintain a stable buffer that acts as a cushion against network variability, preventing depletion while also avoiding excessive buffering that could contribute to initial or interactive latency.
3. **Maintain Acceptable Video Quality:** While prioritizing latency, the algorithm also aims to deliver the highest possible video quality (bitrate) that can be sustained under the prevailing network conditions and latency constraints, thereby contributing positively to the overall Quality of Experience (QoE).
4. **Adapt to Network Dynamics:** To effectively respond to fluctuations in both available bandwidth and network RTT, common in mobile environments.

3.3.2 Core Adaptation Logic of `LatencyAwareAbr`

The `LatencyAwareAbr` makes segment-by-segment bitrate selection decisions based on an amalgamation of factors:

1. **Bandwidth Prediction:**
   i. It maintains a history of recently observed segment download throughputs.
   ii. The predicted throughput for the next segment is calculated using a **harmonic mean** of the throughputs in a defined recent history window (`history_window_s`). The harmonic mean is chosen for its tendency to be influenced more by lower values, offering a somewhat conservative estimate in variable conditions.
2. **RTT/Delay Prediction:**
   i. It maintains a history of recently measured RTTs (derived from the synthetic RTT profile applied to the client).
   ii. The predicted RTT for the next segment download is calculated using a **moving average** of the RTTs in a defined recent history window.
3. **Estimated Segment Download Time:**

    i. For each available bitrate quality, the algorithm estimates the time required to download the next segment. This is calculated as: `EstimatedDownloadTime = (SegmentSizeInBits / PredictedThroughputInBps) + PredictedRTTInSeconds`
    ii. A safety factor (`DOWNLOAD_TIME_SAFETY_FACTOR`) is applied to this estimate to account for prediction inaccuracies and sudden network changes.

4. **Buffer-Aware and Latency-Sensitive Decision Making:**
    i. The algorithm iterates through available bitrates, typically from highest to lowest.
    ii. It selects the highest quality for which the `safe_estimated_download_time_s` is less than the segment duration (ensuring, ideally, that the segment can be downloaded faster than it is played).
    iii. **Conservative Upshifting:** A specific buffer threshold (`LATENCY_AWARE_CONSERVATIVE_BUFFER_S`) is used. If the current buffer level is below this threshold, the algorithm is more reluctant to switch to a higher bitrate, even if throughput predictions are favorable. It might only upshift if the estimated download time for the higher quality is significantly (e.g., >30%) faster than the segment duration, or it might maintain the current or a lower quality. This aims to prevent overly optimistic upshifts that could quickly deplete a fragile buffer, especially when RTT is high or variable.
    iv. **Downshifting:** If no quality can be downloaded within the segment duration or if buffer levels are critically low, the algorithm will aggressively downshift, potentially to the lowest available bitrate, to prioritize avoiding or recovering from a stall.

3.3.3 Implementation in the Simulation Environment

The `LatencyAwareAbr` is implemented as a Python class within the simulation framework. This class encapsulates the state (e.g., history of bandwidth/RTT, current quality index) and the logic described above. It receives the current player state (buffer level, network observations) from the simulator at each decision point (before requesting a new segment) and returns the chosen quality index for the next segment. This modular design allows for easy substitution with other ABR algorithms for comparative studies.

3.4 Evaluation Methodology and Metrics

The performance of the `LatencyAwareAbr` algorithm, particularly its effectiveness in managing latency under varying RTT conditions, is assessed using a quantitative evaluation methodology within the developed simulation framework.

3.4.1 Experimental Design for Evaluation

The core experimental design focuses on comparing the performance of simulated clients operating under different RTT profiles while sharing the same challenging mobile bandwidth trace.

1. **Controlled Variable:** The primary independent variable manipulated is the client's RTT profile (e.g., `Client1_LowLatency` vs. `Client2_HighLatency`).
2. **Constant Factors:** For these direct comparisons, the input bandwidth trace and the ABR algorithm (`LatencyAwareAbr`) are kept identical for all clients in a given simulation run. Player parameters (e.g., buffer sizes, segment duration) are also constant.
3. **Benchmarking (Optional):** For a broader evaluation of `LatencyAwareAbr` itself, its performance can be benchmarked against a baseline ABR (e.g., a simple buffer-based ABR) under identical network trace and RTT conditions.

Simulations are run for the full duration of each selected bandwidth trace. Multiple traces representing different mobile scenarios (Bus, Metro-Tunnel, Train-Fast) are used to assess performance across a range of network dynamics.

3.4.2 Objective Performance Metrics

The following objective metrics, automatically logged and calculated by the simulation framework, are used to quantify performance. These align with common industry and academic measures of streaming QoE:

1. **Startup Delay (seconds):** The time elapsed from the simulation start (representing user request) until the player buffer reaches `MIN_BUFFER_TO_START_S` and playback commences.
2. **Total Stall Time (seconds):** The cumulative duration the player spends in a 'STALLED' state due to buffer underrun after playback has started.
3. **Buffering Ratio (%):** The Total Stall Time expressed as a percentage of the total playback session duration (playback time + stall time).
4. **Average Selected Bitrate (kbps):** The mean bitrate of all video segments successfully downloaded and played during the session, weighted by their duration. This serves as a proxy for perceived video quality.
5. **Quality Switch Frequency (count / switches per minute):** The total number of times the ABR algorithm changes the video quality level during playback. This can also be normalized by playback duration.
6. **Playback Latency Discrepancy (seconds):** Calculated at each simulation time step as the difference in `current_playback_time_s` between two concurrently simulated clients (e.g., `PlaybackTime_Client1 - PlaybackTime_Client2`). This directly measures the relative lag or lead between viewers. Time-series plots and summary statistics (e.g., average and maximum discrepancy) are analyzed.
7. **Simple QoE Score:** A composite objective metric calculated as: `QoE = (AverageBitrateReward) - (StartupPenalty) -`

`(BufferingRatioPenalty) - (SwitchPenalty)` where rewards and penalties are weighted factors. This provides a single figure of merit for overall experience, though its specific formulation must be acknowledged.

3.4.3 Data Analysis and Interpretation

The analysis involves:

1. **Descriptive Statistics:** Calculating mean, median, and variance for the above KPIs across different simulation runs (e.g., per trace, per RTT profile).
2. **Time-Series Visualization:** Plotting key metrics over simulation time (e.g., buffer level, selected bitrate, RTT, bandwidth, playback latency discrepancy) to understand the dynamic behavior of the ABR and player.
3. **Comparative Analysis:** Comparing KPIs and dynamic behaviors between clients with different RTT profiles or between different ABR algorithms.
4. **Correlation Analysis (Potential):** Exploring correlations between input network parameters (e.g., RTT magnitude, RTT variance, bandwidth volatility) and output performance metrics.

The results are interpreted in the context of the research objectives, focusing on how RTT impacts latency and how effectively the `LatencyAwareAbr` mitigates these impacts. Insights gained are used to identify strengths and weaknesses of the algorithm and to inform potential refinements. While this phase relies on objective data, the implications for subjective user experience are discussed based on established relationships between objective metrics and perceived QoE from the literature.

References

1. 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.
2. 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.
3. 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.
4. 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.

5. 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.
6. 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.
7. 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.
8. Mao, H., Netravali, R., & Alizadeh, M. (2017). Neural adaptive video streaming with pensieve. *Proceedings of the Conference of the ACM Special Interest Group on Data Communication (SIGCOMM '17)*, 197-210.
9. Oyman, O., & Singh, S. (2012). Quality of experience for HTTP adaptive streaming services. *IEEE Communications Magazine*, *50*(4), 20–27.
10. Paudyal, P., Battisti, F., & Carli, M. (n.d.). Impact of video content and transmission impairments on quality of experience. *Multimedia Tools and Applications*.
11. 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.
12. 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.
13. Smith, J., Doe, A., & Brown, B. (2020). Scalable Cloud Architectures for Video Streaming. *Journal of Cloud Computing*, *9*(1), 1-15. *(Example placeholder for a cloud deployment reference if relevant to future scope discussed)*.
14. Weng, J., et al., (2017). Evaluation of video streaming. *(This is too vague, needs a full citation or replacement if a specific paper is intended)*.
15. 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.
16. 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.