Harper Yan V01036841
Bi-Weekly Update: Analysis and Implementation of Scalable Video Streaming over P2P Network
Date: March 21, 2025

Overview

Over the past two weeks, he focus was on resolving bugs in the bandwidth sampling and monitoring logic, followed by scaling up the experiment to include more peers and varied churn rates. This update details the fixes implemented, the expanded experiment setup, and initial observations. Updated scripts are available at: https://github.com/Harper-Yan/CSC-579-P2P-Videostreaming-simulation/master.

Progress

1. **Bug Fixes in Bandwidth Sampling and Monitoring**
    - **Issue Identified**: In the previous update, the bandwidth data for Peer 4 showed repetitive and inconsistent values (e.g., oscillating between without clear correlation to system dynamics). This was traced to an error in the P2P bandwidth sampling function within the cdnbye-dashjs-p2p-engine integration, where the WebRTC data channel was not properly aggregating bandwidth usage across all active peers.
    - **Resolution**: Modified the sampling logic to ensure accurate tracking of per-peer bandwidth by implementing a rolling average over a 5-second window and filtering out duplicate or stale data points. The fix also addressed a race condition in Puppeteer's peer simulation script that caused bandwidth logs to be overwritten.
    - **Outcome**: Post-fix, bandwidth metrics are now consistent and reflect actual P2P offloading activity, validated through manual inspection of logs and real-time stats from dash.js.
2. **Scaling the Experiment**
    - **Setup**: Increased the simulation scale from 5 peers to 50 peers using Puppeteer to launch multiple Chrome instances. Adjusted the peer-churn rate range from a single 30% to a stepped range (10%, 30%, 50%, 70%) to explore performance under varying network stability conditions. The video source was upgraded to a larger file (10-minute MPEG-DASH stream, ~50 MB) to stress-test buffering and bandwidth utilization.
    - **Execution**: Ran the simulation for 30-minute intervals per churn rate, with real-time logging of startup delay, buffering events, and P2P bandwidth usage. Each run included 10% of peers joining/leaving every 5 minutes to simulate realistic churn.

Next Steps

- Refine WebRTC stability for high-churn scenarios by tweaking Swarm Cloud token settings or exploring alternative trackers.
- Develop scripts for automated data filtering and visualization to interpret the results more effectively.
- Test with even larger peer counts (e.g., 100+) to assess scalability limits.