Assignment 1

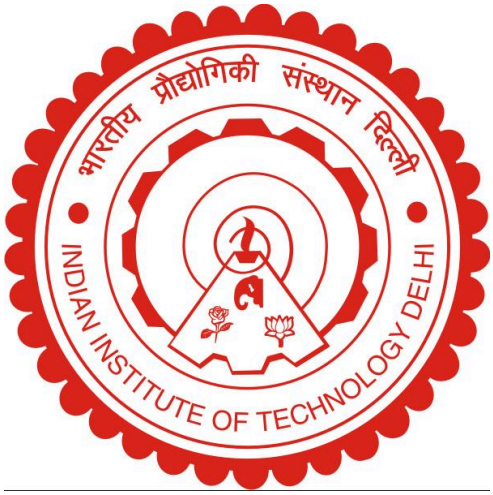
Data Collection for Application Performance Monitoring

Submitted By :

Manish Kumar (2023MCS2497)

Pratham Agrawal (2023MCS2486)

Submitted On : **28th February, 2025**



DEPARTMENT OF COMPUTER SCIENCE & TECHNOLOGY
Indian Institute of Technology, Delhi
Hauz Khas, New Delhi, 110016

Academic Year: 2024-25

Analysis:

1. Introduction

The goal of this project is to automate the collection of application performance metrics during video streaming sessions on YouTube under varying network conditions. This report documents the implementation of the automation framework, the data collection process, and the preliminary data analysis.

2. Automation Framework

2.1 Task Details

The automation framework is implemented in Python using Selenium and BrowserMob Proxy to collect network traffic and application performance metrics. The framework performs the following tasks:

- Opens YouTube video URLs using the Chrome browser.
- Played each video for 3 minutes while attempting to skip ads.
- Captures HTTP transactions using HAR logs.
- Captures packet-level network traffic using `pyshark`.
- Collects video resolution and buffer occupancy metrics every second.
- Logs the startup latency, re-buffering ratio, average resolution, and network bandwidth statistics.

2.2 Implementation

The framework is composed of two primary scripts:

- `main.py`: This script implements the core functionality of the automation process.
- `automate.sh`: This bash script reads video URLs from a file and runs the Python script sequentially for each URL.

2.3 Network Shaping

Network shaping is emulated using Linux's traffic controller (`tc`). The network bandwidth is varied every 5 seconds, randomly selected between 50 kbps and 5 Mbps. The shaping functions are implemented in:

- `start_shaping()`: Initializes the network shaping.
- `update_shaping()`: Updates the bandwidth periodically.
- `stop_shaping()`: Stops network shaping.

3. Data Collection

3.1 Process

The framework collects data for multiple video sessions. Each session:

- Streams a randomly selected video for 3 minutes.
- Varies the network bandwidth at 5-second intervals.
- Captures packet traffic into `.pcap` files.
- Logs HTTP transactions into `.har` files.
- Records video metrics into `.csv` files.
- Summarizes session metrics into `.log` files.

3.2 Output Files

Each session generates the following files:

- `486497_<timestamp>.pcap`: Packet capture file.
- `486497_<timestamp>.har`: HAR log file.
- `486497_<timestamp>.csv`: Video metrics (timestamp, resolution, buffer occupancy).
- `486497_<timestamp>.log`: Summary of startup time, re-buffering ratio, average resolution, bandwidth statistics.

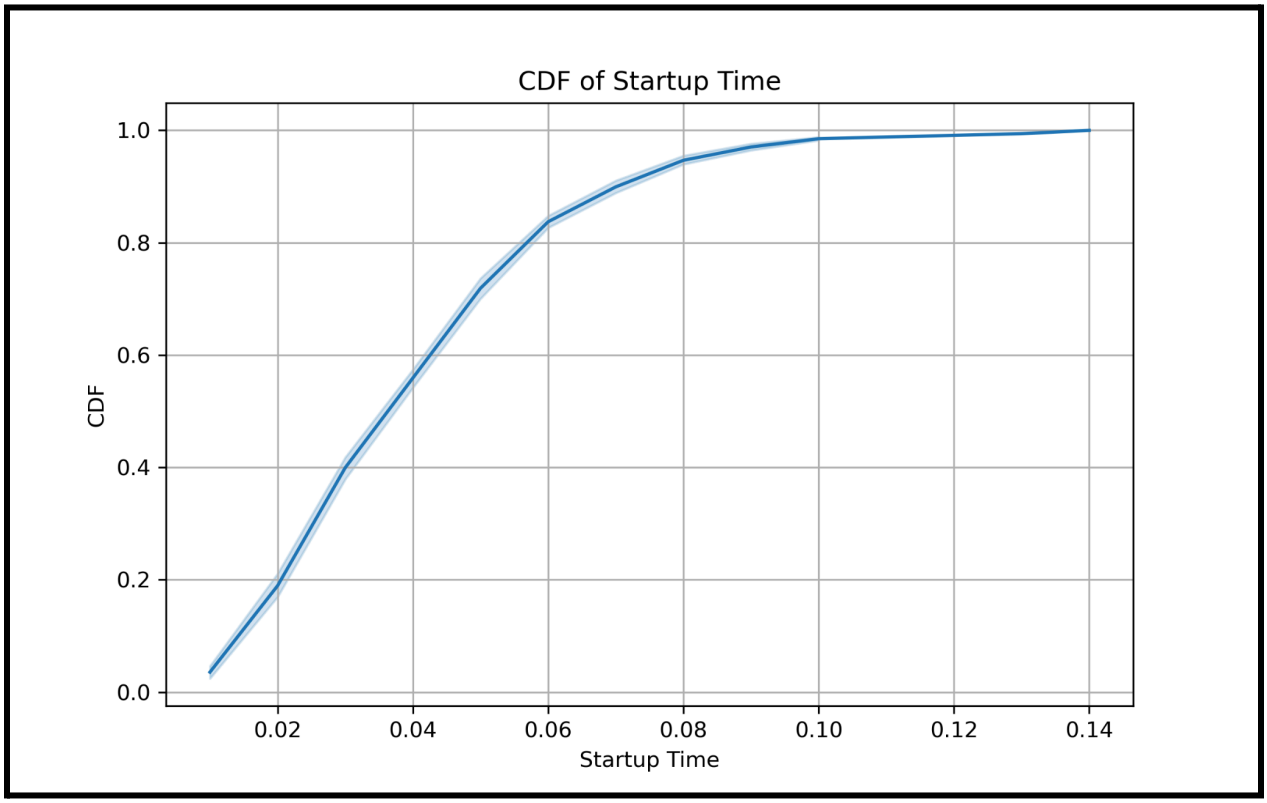
4. Data Analysis

We collected data for 169 separate youtube links

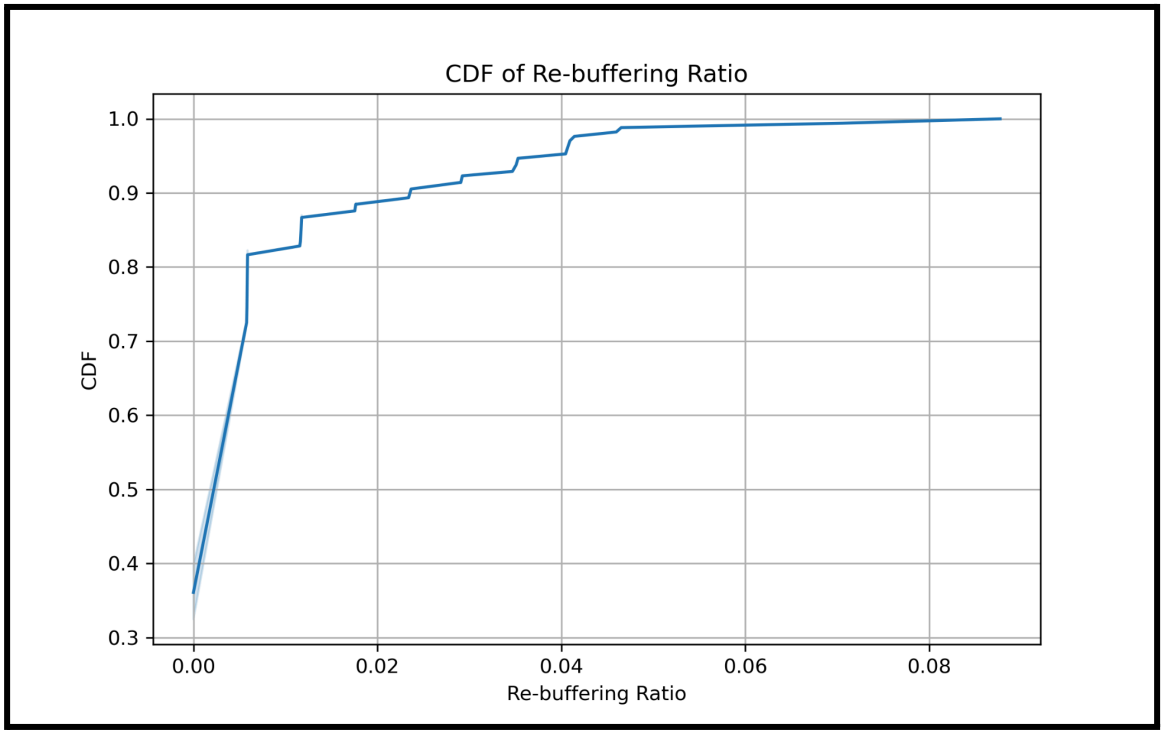
4.1 Dataset Characterization

Preliminary dataset characterization includes:

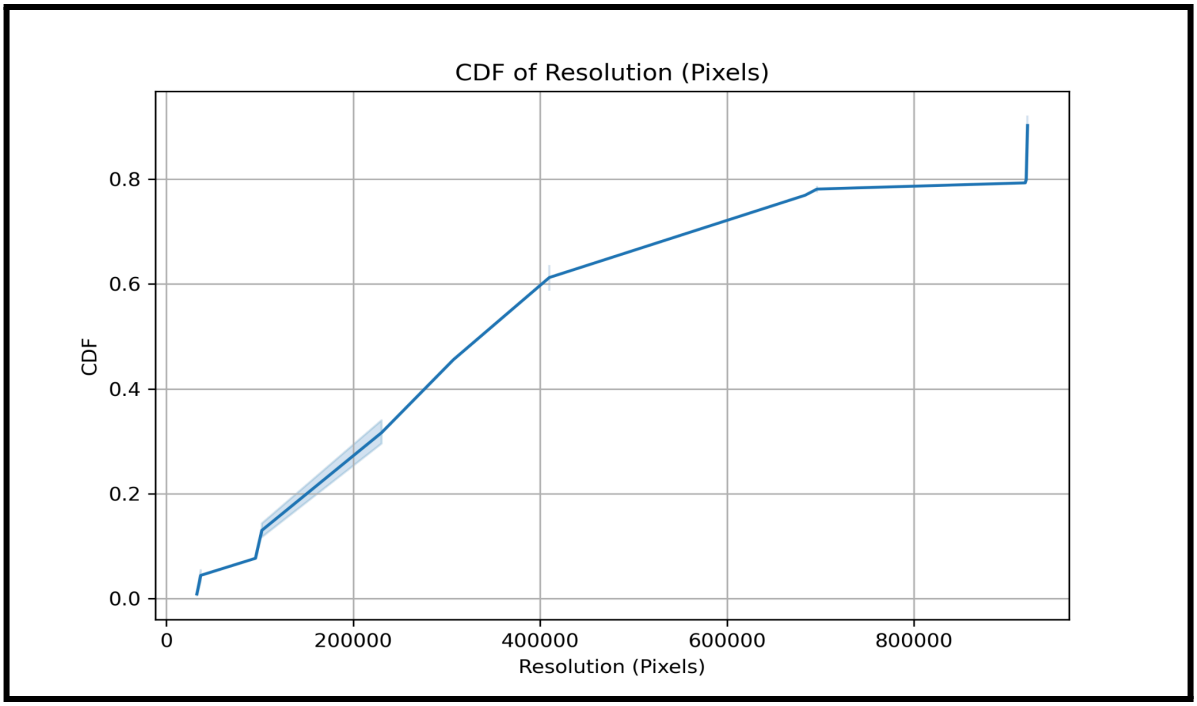
- Cumulative Distribution Function (CDF) of:
    - **Startup latency.**



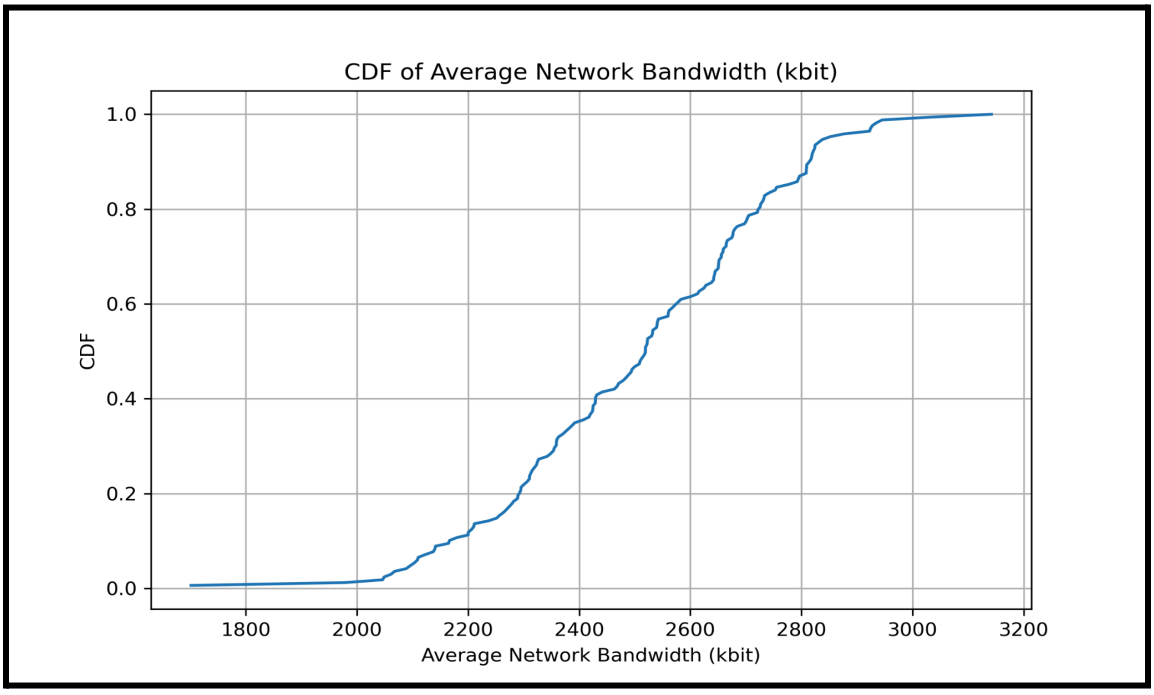
- **Re-buffering ratio.**



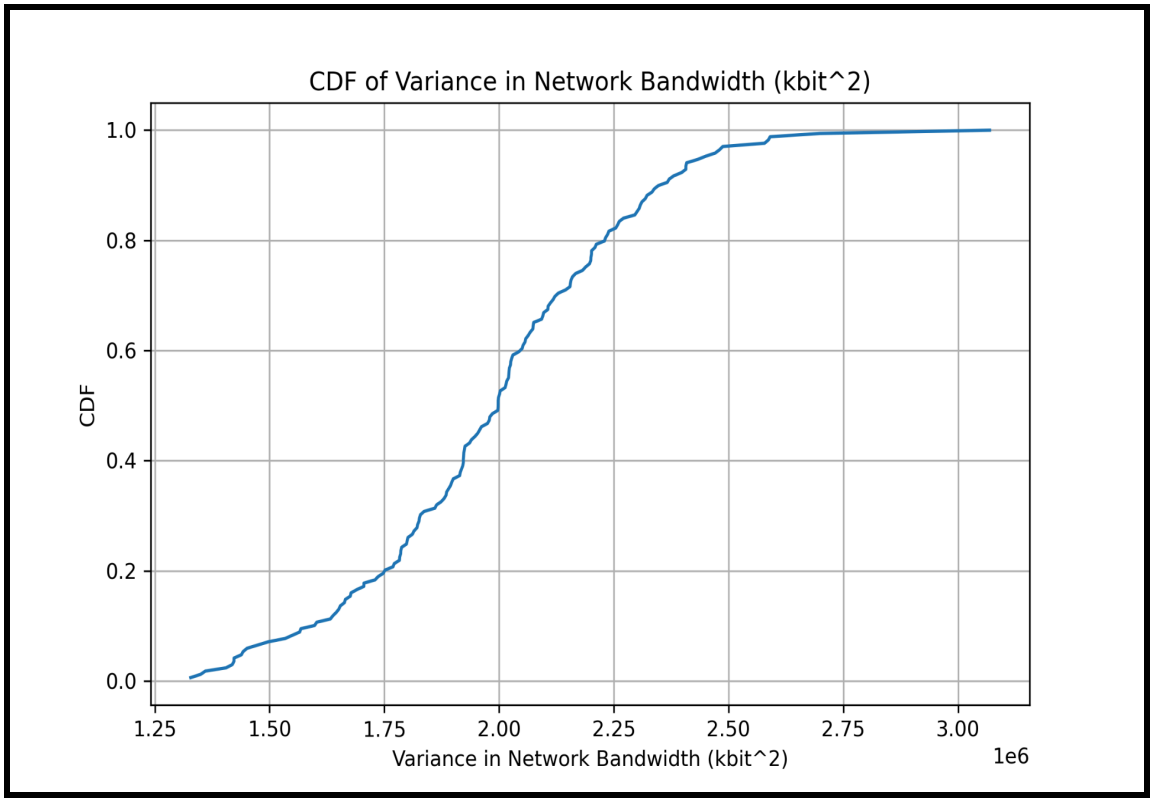
- **Average resolution**



- **<u>Average network bandwidth.</u>**



- **<u>Variance in network bandwidth.</u>**



4.2 HTTP Logs

The HAR logs provide insights into HTTP transactions for video and audio data. We looked into the i-tag values and found that almost every file had around ***12 chunks with an i-tag value of 18 which is equivalent to that of 360p for video playback.*** We believe that only chunks with lower resolution we represent with an i-tag field & the rest of chunks were encoded with some other tag in the HAR capture file. A finer study would particularly involve:

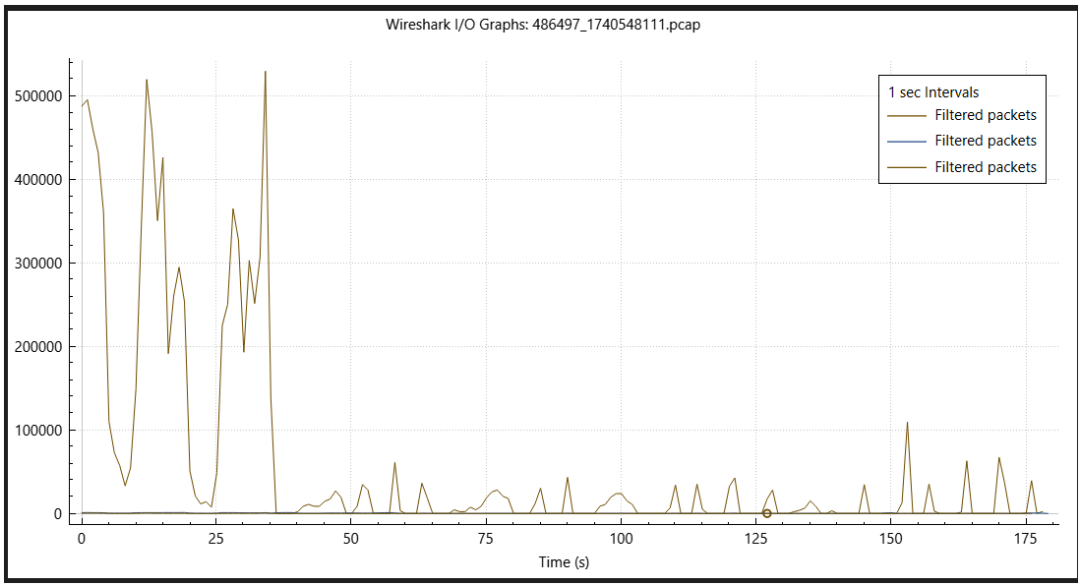
- Parsing HAR logs to extract **video/audio transactions**.
- Identifying **requested video resolution**.
- Comparing **requested vs. player-reported resolution**.

4.3 Network Logs

The packet capture files (`.pcap`) are collected, but **we were not able to extract HTTP transactions & useful metrics from those**.

We noticed that almost all the HTTP GET requests are actually encrypted under TLS which makes it harder to identify directly.

However, from the below plot it is evident that the client downloaded video chunks at regular intervals in order to keep video playback smooth. (However, we could evaluate the exact count of these chunks & their individual sizes).



Future improvements should include:

- Parsing `.pcap` files to identify **HTTP transactions**.
- Comparing **network logs with HAR logs** to validate video/audio transactions.

5. Conclusion

This report presents the implementation and preliminary analysis of an automation framework for application performance monitoring during video streaming sessions. The framework successfully captures HAR and PCAP logs.

Data Link : DATA_HSN_A1