Project StreamSight
Hackenza: Characterization of Delays in Packet Transmission

Team Rocket

Jayant Choudhary [2023A7PS0404G]
Swayam Lakhotia [2023A7PS0368G]
Siddhant Kedia [2023A7PS0375G]
Pratham Chheda [2023AAPS0138G]

Team Skills

Pratham Chheda

- Proficiency in Machine Learning
- GitHub for SAIDL: SAIDL Assignment
- Experience with Text-to-SQL AI startups

Swayam Lakhotia

- Proficient in C++ and Python
- Worked with large datasets (ICICI Bank)
- Developed a Valorant bot for voice chat music

Jayant Choudhary

- Proficient in Python and C++
- Experience in computer vision, SLAM, and industrial projects
- Worked for a Text-to-SQL AI startup

Siddhant Kedia

- Proficient in C, C++ and various Python libraries

1. Why We Chose This Project

With a newfound interest in networking after our Network Programming course, we wanted to broaden our horizons. This project provides us with the necessary challenge to explore networking and focus on subfields like IoT.

2. Our Approach and Current Progress

- We began by examining the `.pcapng` file in Wireshark to inspect its structure and understand key packet attributes (timestamps, source/destination addresses, protocol details).
- Next, we used PyShark to extract meaningful data from the file. We identified various transmission delays (processing, latency, and retransmissions) and encountered negative time differences indicating out-of-order arrivals.
- Our focus is on identifying the protocols used, packet sizes, and the frequency of delays. We are specifically analyzing TCP, UDP, and MQTT.

3. Implementation Goals

- List all protocols used, packet sizes, and delay frequencies.
- Analyze TCP, UDP, and MQTT to understand unique delay characteristics.
- Measure key network performance metrics.
- Identify MQTT client-broker interactions using CONNECT and CONNACK messages.
- Map cloud interactions for MQTT.
- Correlate TCP, UDP, and MQTT data to detect anomalies and inefficiencies.

4. Implementation Strategy

4.1. TCP Analysis

1. **Inter-Packet Delay (IPD):** Time between consecutive TCP packets.
2. **Retransmission Delay:** Time between resending the same sequence number.
3. **RTT Delay:** Time taken for a SYN-ACK handshake or data-ACK response.
4. **ACK Delay:** Time between sending a data packet and receiving an acknowledgment.
5. **Jitter, Packet Loss & Congestion:** Tracking delay variations and dropped packets.

4.2. UDP Analysis

1. **Inter-Packet Delay (IPD):** Gaps between consecutive UDP packets.
2. **Jitter Calculation:** Tracking delay variations per IP.
3. **Packet Loss Detection:** Using RTP sequence numbers or timestamp gaps.
4. **Congestion Estimation:** Based on high jitter and packet loss.

4.3. MQTT Analysis

- Clients initiate communication with a `CONNECT` packet; brokers respond with `CONNACK`.
- Cloud services are identified by filtering packets forwarded beyond the broker (typically with higher latencies and external IPs).
- Using `msg_type`, we'll track communication flow and delays at different levels:
  - **Broker Processing Delay:** Time taken by the broker to process and forward messages. We will check for the same by:
    1. **Pinpoint broker "receive" event:** This is typically the arrival of the MQTT `PUBLISH` at the broker.
    2. **Pinpoint broker "send" event:** This could be the broker's next outbound message related to the same publish.
    3. **Calculate time difference:** Broker Processing Delay = Timestamp(broker outbound event) – Timestamp(broker inbound event).
  - **Broker-Client Delay:** To check for Time between broker transmission and client reception:
    1. Track the MQTT `PUBLISH` packet originating from the client.
    2. Look for the corresponding MQTT acknowledgment.
    3. **Calculate time difference:** Device-to-Broker Delay = Timestamp(broker receives or responds) – Timestamp(device sends PUBLISH).
  - **Cloud Delay:** To check for Time taken for messages sent from the broker to reach cloud services:
    1. **Identify the broker-to-cloud flow:** Filter out this connection by checking for a new TCP connection (or TLS connection) from broker IP/port to the cloud IP/port.
    2. **Mark the "start":** When the broker sends the first packet to the cloud.
    3. **Mark the "end":** When the cloud service responds with an ACK, or a final MQTT acknowledgment/HTTP 200, etc.
    4. **Calculate:** Cloud Upload Delay = Timestamp(cloud ACK) – Timestamp(broker first send to cloud).

5. Integration

- Correlate TCP, UDP, and MQTT delay patterns to detect unusual network behavior.
- Identify issues such as excessive delay, retransmission loops, packet loss spikes, and congestion.
- Aim to understand overall packet efficiency and pinpoint bottlenecks.

6. Visualization and GUI

Dashboard Overview
We have chosen a combination of tools for our visualization layer:

- **Plotly:** For interactive, dynamic visualizations.
- **Streamlit:** For developing a clean and responsive UI with minimal code overhead.

Our current dashboard prototype includes:

- Anomaly detection table for delays exceeding thresholds.
- Display of average latency.
- Distribution of protocols by packet.
- Delay analysis by protocol.
- Delay timeseries graphs.

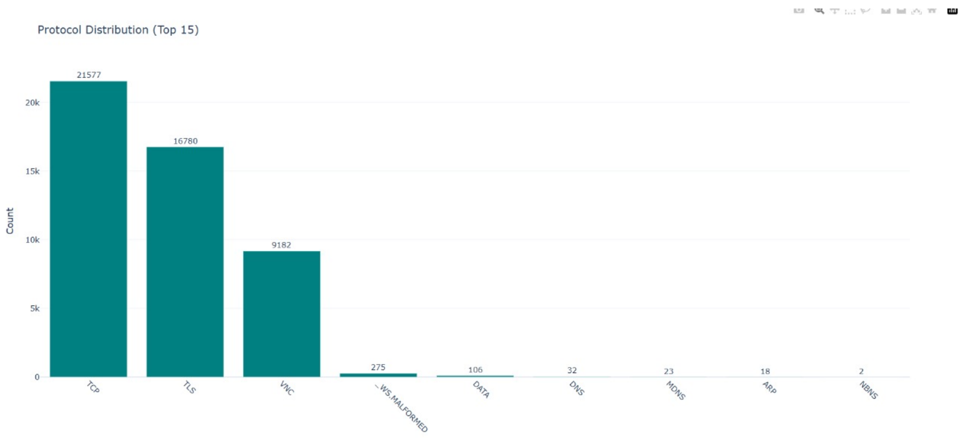


Figure 1: Graph showing protocol distribution of a pcapng file analyzed through PySharK

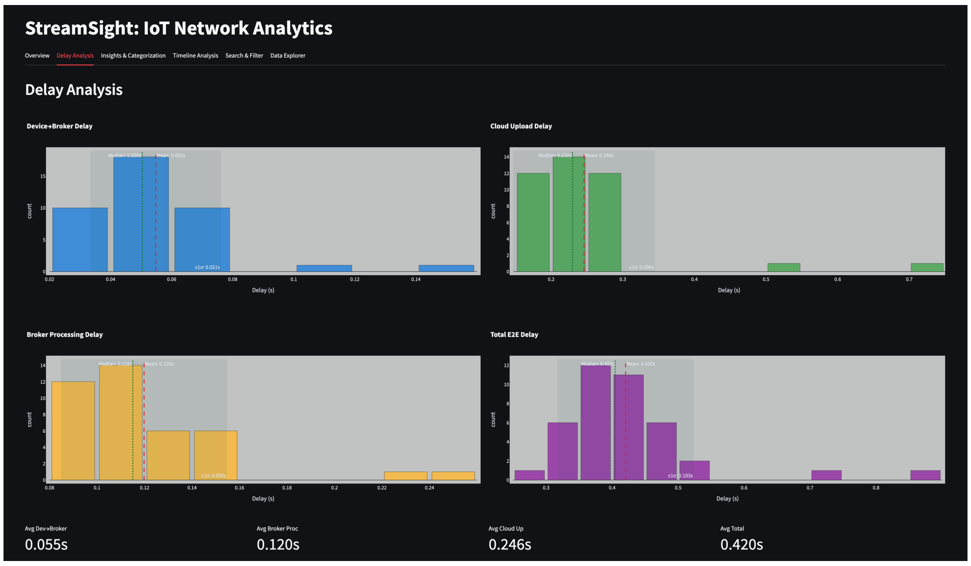
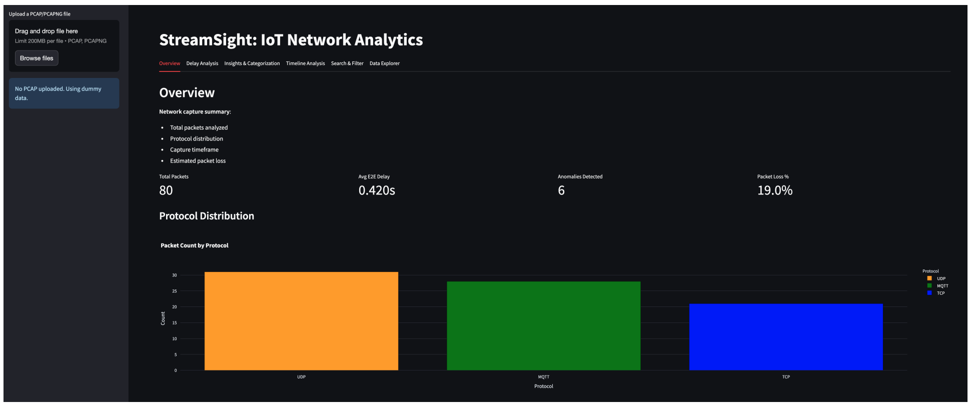


Figure 2: Interactive display showing packet information details

7. User Interface Design

The interface includes:

1. **Navigation Sidebar**
   - File upload component
   - Analysis settings
   - Filter controls
2. **Overview Dashboard**
3. **Delay Analysis Workspace**
   - Interactive timeline with color-coded delay types
   - Packet flow visualization
4. **Insights & Categorization**
   - Anomaly Detection Thresholds
   - Delay Type
5. **Search and Filter**
6. **Data explorer**



No PCAP uploaded. Using dummy data.

StreamSight: IoT Network Analytics

Overview

Network capture summary:

- Total packets analyzed
- Protocol distribution
- Capture timeframe
- Estimated packet loss

Total Packets 80
Avg E2E Delay 0.420s
Anomalies Detected 6
Packet Loss % 19.0%

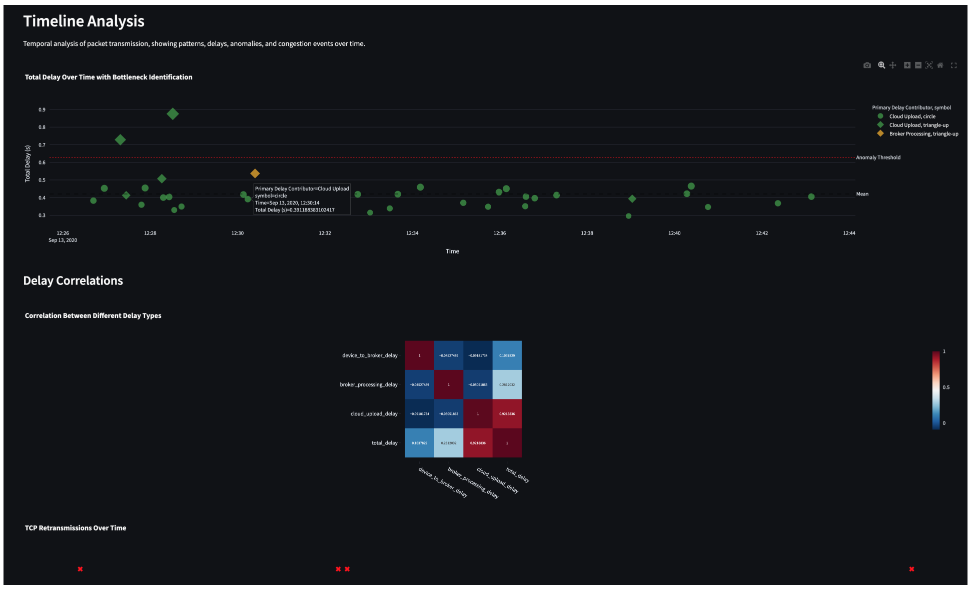
Protocol Distribution

Figure 3: User interface showing protocol distribution and filtering controls

7.1. Interactive Timeline

We plan to implement a zoomable, interactive timeline that shows:

- Packet flow with color-coded delay types.
- Anomaly markers for significant delay events.
- Protocol-specific indicators.
- Hoverable tooltips with packet details.


Timeline Analysis

Temporal analysis of packet transmission, showing patterns, delays, anomalies, and congestion events over time.

Total Delay Over Time with Bottleneck Identification

Primary Delay Contributor=Cloud Upload
symbol=circle
Time=Sep 13, 2020, 12:30:14
Total Delay (s)=0.391188383102417

Delay Correlations

Correlation Between Different Delay Types

	device_to_broker_delay	broker_processing_delay	cloud_upload_delay	total_delay
device_to_broker_delay	1	−0.04527489	−0.09181734	0.1037829
broker_processing_delay	−0.04527489	1	−0.05051863	0.2812032
cloud_upload_delay	−0.09181734	−0.05051863	1	0.9218836
total_delay	0.1037829	0.2812032	0.9218836	1

TCP Retransmissions Over Time

Figure 4: Timeline analysis and correlation matrix

8. Root Cause Identification

Our system will attempt to identify the fundamental causes of delays. When anomalies are detected:

- The system flags the primary bottleneck.
- Visualizes delay spikes alongside network conditions like packet loss, protocol patterns, and throughput rates.
- Uses a correlation heatmap to compare normal operations against anomalies.

9. Logic Behind Different Graphs

We employ several targeted visualizations:

- **Protocol Distribution Bar Chart:** Breaks down the protocols in the network capture.
- **Delay Histograms:** Show the distribution of delays for:
  - Device-to-Broker delay.
  - Broker Processing delay.
  - Cloud Upload delay.
  - Total End-to-End delay.
- **Timeline Analysis:** Plots delays over time with highlighted anomalies.
- **Retransmission Timeline:** Visualizes instances of packet retransmission.
- **Correlation Heatmap:** Displays relationships between different delay types.
- **Bottleneck Analysis:** Identifies which communication stage contributes most to overall delays.

10. Limitations

- PCAPNG files provide limited information for network analysis.
- Only major protocols (TCP, UDP, MQTT) are analyzed; additional protocols (e.g., TLS) are omitted to maintain clarity.
- Specific data for IoT MQTT systems is not provided, which may limit testing of our analysis.

11. Future Scope

- Improve parsing speed by using JSON parsing or preprocessing to filter only the relevant protocols.
- Incorporate API calls to offer AI suggestions for network issues.
- Implement advanced statistical and correlation analyses, such as:
  - Packet size vs. delay time.
  - Impact of protocol overhead.
  - Time-of-day patterns.
  - Network congestion predictions.