<u>Communication Networks – final project</u>

<u>Part 1</u>

<u>Q1</u>

“A user reports that their file transfer is slow, and you need to analyze the transport layer to identify the potential reasons. What factors could contribute to the slow transfer, and how would you troubleshoot it?”

1. TCP Receive Window – if the receiver window is too small, the receiver might limit the file transfer rate and cause it to be slow.

2. TCP Congestion Window – if TCP senses that the network is congested – using mechanisms like slow start and congestion avoidance, the sender might reduce its transfer rate.

3. High Packet Loss – if the sender is experiencing much packet loss this can cause a lot of packet retransmission – which in turn will cause the file transfer to be slow.

4. High Latency (Round Trip Time) – if the sender and the receiver are too far away from each other – the packets travel time can be increased and the sending rate would suffer.

TCP parameters can be inspected in the Linux kernel using the “sysctl” utility, for example: running “sysctl -a | grep tcp” would show all of them.

We can also use “netstat” to inspect retransmissions and packets buffering in the active connections – so we could run it while the transfer is ongoing.

To check the RTT, we can use many tools such as ping and traceroute – ping will show it clearly and traceroute will show the route a packet is making – which can indicate if the receiver and sender are too far away.

To just check raw performance, we can use iperf to test the network throughput between the sender and the receiver.

Q2

“Analyze the effects of TCP’s flow control mechanism on data transmission.
How would it impact performance when the sender has significantly higher processing power than the receiver?”

TCP flow control mechanism ensures that the sender does not overwhelm the receiver with data that the receiver can’t handle.
It is done by the TCP receive window which is sent in the TCP header. Once it has been sent, the sender must not send more data than that window. That window can get smaller if the receiver is not processing that packets fast enough.

In the case above, in which the sender has significantly higher processing power than the receiver we would get a situation in which the receiver’s window would shrink quickly – which will limit the sender’s ability to send data and overall would kill performance as the sender would be waiting most of its time for window updates.

Q3

“Analyze the role of routing in a network where multiple paths exist between the source and destination. How does the path choice affect network performance, and what factors should be considered in routing decisions?”

The path choice can affect the following factors of network performance:

1. Latency – A longer route can increase the delay time in which packets arrive which can affect any application which needs real time performance.

2. Bandwidth – When selecting paths, some paths can be congested which can hurt the throughput. Some can even be more limited from the hardware aspect.

3. Packet Loss – If we select a congested path we can experience more packet loss and reduce the throughput.

4. Managing Paths Effectively – By controlling overall network flow in multiple paths available, we can optimize the overall utility.

Some routing factors that should be considered are:

1. Hop Count - The number of intermediate routers between source and destination.
2. Bandwidth – Sometimes we would prefer higher bandwidth links.
3. Latency – Links with smaller delay times.
4. Reliability – Links with less packet loss.
5. Load – Less loaded links, sometime a less loaded link with less bandwidth is better than a loaded one with more bandwidth.

Q4

"How does MPTCP improve network performance?"

MPTCP (Multipath TCP) is an extension of TCP which grants one single TCP connection the ability to use multiple network paths simultaneously.

This improves network performance by allowing data transmission over multiple interfaces (such as Wi-Fi, Cellular, Ethernet etc ..) concurrently – which improves the network bandwidth by aggregating multiple bandwidths of several interfaces.

So overall, the following improvements are made:

1. Bandwidth – By combining multiple interfaces, we can aggregate their bandwidths and increase the total bandwidth.

2. Reliability – If one link fails, we can still continue to send data over the other ones which can increase reliability.

3. Load – MPTCP uses its links smartly and spread the load over its links efficiently.

4. Latency – MPTCP can use the path with the least latency and use the other ones as backup. (Assuming one link is sufficient for the total usage in an ideal scenario).

Q5

"You are monitoring network traffic and notice high packet loss between two routers. Analyze the potential causes for packet loss at the Network and Transport Layers and recommend steps to resolve the issue."

Causes for packet loss in the network layer:

1. High Incoming Packet Rate – If packets arrive in a router at high speed, the router's ability to process them may be slower than the incoming rate, which can cause the router the drop new incoming packets.

Symptom: high packet loss on an end point (seen using netstat -s).

Possible Solutions:
a. increase the buffer sizes.
b. Navigate other traffic to other routers.

2. Incorrect Configurations – If routers are mis-configured, we can have infinite loops and unreachable routers (through routing configuration error or firewall mis configuration) – which of course can cause packet loss. Another bad configuration that can happen is configuring the MTU (Maximum Transmission Unit) to be too small, if a large packet arrive and its Don't Fragment bit is set – it would be dropped.

Symptoms:
a. Inconsistent "traceroute" results.
b. Unreachable router using ping command.
c. Small packet sessions succeed but other don't.

Possible Solutions:
a. "ip route show" - can show us the incorrect routing configurations.
b. checking firewall rules.
c. checking the routing protocol tables.
d. changing the MTU using "ifconfig".

Causes for packet loss in the transport layer:

Network Congestion and High Latency – If a network is congested packets can be dropped. Also, when the network if congested we can experience higher latnecy which can cause timeouts. (not getting acks for the packets we sent in time therefore retransmitting them).

Symptoms
a. High RTT inspected using the ping command.
b. Retransmissions before acks arrive – inspected with the netstat command.

Possible Solutions
a. configure TCP fast retransmissions using "sysctl -w net.ipv4.tcp_frto=1".
b. increase TCP buffer sizes – using "sysctl -w net.ipv4.tcp_rmem=<values>".

Part 2

In this part, we made use of Google Notebooklm which is aimed mostly to analyze articles and provide answers to the questions asked in this section. All of its answers were validated by us.

What is the main contribution of the paper?

Prompt: For each paper, What is the main contribution of the paper?

Based on the sources provided, here are the main contributions of each paper:

"Analyzing HTTPS Encrypted Traffic to Identify User's Operating System, Browser and Application"

- This paper presents a novel approach for identifying a user's operating system, browser, and application from their HTTPS encrypted traffic. This is achieved by analyzing traffic patterns, including new features that exploit browser bursty behavior and SSL behavior.
- The authors created and made available a comprehensive dataset of more than 20,000 labeled sessions with different operating systems, browsers, and applications.
- The paper demonstrates that it's possible to classify encrypted network traffic with a high accuracy of 96.06% by combining baseline and new features using supervised machine learning techniques.
- The study uses sessions as samples, where a session is a tuple consisting of the protocol, source and destination IPs, and source and destination ports. The label for each session is a tuple of the OS, browser, and application. The task is inherently a multiclass learning with 30 classes.
- The research also introduces a new set of features based on SSL features, TCP features, and the bursty behavior of browsers.

"Early Traffic Classification With Encrypted ClientHello: A Multi-Country Study"

- This paper introduces a new early traffic classifier named **hybrid Random Forest Traffic Classifier (hRFTC)**. This classifier utilizes unencrypted TLS metadata along with statistical features of traffic flows extracted before any application data arrives from the server side.
- The paper compiles a large, detailed, and diverse open-source traffic classification dataset gathered from multiple countries in North America, Europe, and Asia. The dataset includes various traffic types with different QoS requirements.
- The study evaluates the performance of state-of-the-art traffic classification algorithms on this newly collected dataset.
- The research reveals that relying solely on TLS features in an Encrypted ClientHello (ECH) scenario leads to poor classification accuracy. It further shows that hybrid approaches like hRFTC significantly improve classification performance by using a combination of packet-based and flow-based features.

- The paper demonstrates the limitations of training a model in one geographical location and expecting it to perform well in another, highlighting the need to train traffic classification models in the specific location where they will be used.
- The study extends the RB-RF functionality to handle QUIC traffic flows and improves TC quality while satisfying the eTC classification delay requirements. It also incorporates a new set of flow-based features, considering packets until the first downlink packet with application data.

"FlowPic: Encrypted Internet Traffic Classification is as Easy as Image Recognition"

- This paper introduces a novel approach for classifying Internet traffic by transforming traffic flows into images called "FlowPics". These images are created using packet sizes and packet arrival times.
- The FlowPic images are then fed into a Convolutional Neural Network (CNN) for classification, achieving high accuracy in traffic categorization and application identification.
- The proposed method does not rely on packet payload content, thus preserving privacy.
- The paper also demonstrates that the method can classify traffic passing through VPNs and Tor with good accuracy.
- The method is generic and does not rely on specific application characteristics, as demonstrated by the classification of new applications not included in the training data.
- The study shows that the model can classify traffic based on a short time window of a unidirectional flow, making real-time classification feasible.
- The paper constructs a 2-dimensional histogram for flow representation using the packet size distribution (PSD) of packets for each delta time.

Prompt: What traffic features does each paper use, and which are novel?

Okay, here's a breakdown of the traffic features used in each paper, with an indication of which are novel:

"Analyzing HTTPS Encrypted Traffic to Identify User’s Operating System, Browser and Application"

- **Baseline Features:** This paper uses standard traffic features common in many traffic classification methods:
  - Packet sizes counts.
  - Time differences between packets.
  - Session duration.
  - Number of packets in a session.
  - Minimum, maximum, and mean inter-arrival times of packets.
  - Payload size information.
  - Bitrate.
  - Round trip time.
  - Packet direction.

    - Server-sent bitrate.
- **Novel Features:** The paper introduces a new set of features that focus on browser behavior and SSL characteristics:
    - Initial TCP window size.
    - TCP window scale factor.
    - Number of SSL compression methods.
    - Number of SSL extensions.
    - Number of SSL cipher methods.
    - SSL session ID length.
    - Maximum forward peak throughput.
    - Mean backward peak throughput.
    - Maximum backward peak throughput.
    - Minimum backward peak throughput.
    - Standard deviation of backward peak throughput.
    - Number of forward bursts.
    - Number of backward bursts.
    - Minimum forward peak throughput.
    - Mean forward peak throughput.
    - Standard deviation of forward peak throughput.
    - Mean inter-arrival time difference between backward peaks.
    - Minimum inter-arrival time difference between backward peaks.
    - Maximum inter-arrival time difference between backward peaks.
    - Standard deviation of inter-arrival time difference between backward peaks.
    - Mean inter-arrival time difference between forward peaks.
    - Minimum inter-arrival time difference between forward peaks.
    - Maximum inter-arrival time difference between forward peaks.
    - Standard deviation of inter-arrival time difference between forward peaks.
    - Number of keep-alive packets.
    - Maximum TCP segment size.
    - Forward SSL version.
    - These new features were created through an extensive analysis of network traffic. They exploit the **bursty behavior of browsers**, SSL features, and TCP features. A burst is defined as a section of traffic where there is silence before and after.

"Early Traffic Classification With Encrypted ClientHello: A Multi-Country Study"

- **Packet-based features:** These algorithms use data from the TLS handshake control packets, where all the payload of subsequent packets is encrypted:
    - Raw bytes of the first packets.
    - SNI length or padding length.
    - ClientHello and ServerHello packets.
    - Preferred Cipher Suites.
    - Key Share Group.
    - Order of extensions.

- **Flow-based features:** These algorithms use sequences of packet sizes (PS) and inter-packet times (IPT):
  - Statistical features like 25th and 75th percentiles, median, minimum, maximum, and mean from PS and IPT sequences.
  - Number of packets per direction.
  - Total bytes per direction.
  - Variance of PS and IPT sequences.
  - PS distribution by the number of packets at different sizes: (0-64], (64-128], (128-256], or (256-512] bytes.
- **Hybrid features:** Hybrid algorithms combine packet-based and flow-based features.
  - TLS handshake features combined with statistics of the first 10 application packets.
  - CH length can reveal the SNI.
  - Statistical features over the first application packets.
- **Novel Features in hRFTC:** The new algorithm, hRFTC, uses:
  - Unencrypted payload of the TLS handshake.
  - Flow-based time series.
  - Packet size statistics.
  - Features are extracted from the packet sequence until the first downlink packet with application data. Features are calculated separately for downlink (DL) and uplink (UL) packets.

"FlowPic: Encrypted Internet Traffic Classification is as Easy as Image Recognition"

- **Flow-based features:** This paper transforms network traffic into an image using packet sizes and their arrival times.
  - **FlowPic:** An image created by turning a network flow into a 2-dimensional histogram, reflecting the packet size and their arrival times.
  - Packet size and arrival times are used to create images.
- **Novel Features:** This paper introduces the concept of using **FlowPics and CNNs to analyze network traffic**, avoiding manually selected features. This allows classifying internet traffic without relying on the content of the packet payload.

In summary:

- The first paper, *Analyzing HTTPS Encrypted Traffic*, introduces novel features focusing on browser behavior and SSL characteristics to improve classification accuracy.
- The second paper, *Early Traffic Classification With Encrypted ClientHello*, combines TLS metadata with flow-based features and introduces a novel hybrid approach with its hRFTC algorithm.
- The third paper, *FlowPic*, pioneers the use of a visual representation of traffic flows for classification using CNNs.

Prompt:

What are the main results (you may copy the figures from the paper), and what are the insights from their results?

Okay, here's an analysis of the main results and insights from the provided sources, drawing on our previous conversation and formatted for clarity:

1. "Analyzing HTTPS Encrypted Traffic to Identify User's Operating System, Browser and Application"

- **Main Results:**
    - This paper demonstrates the ability to **identify a user's operating system, browser, and application by analyzing encrypted HTTPS traffic**.
    - The achieved classification accuracy reached **96.06%** using novel features, significantly improving upon a baseline of 93.51% using basic features.
    - The authors created a dataset with over **20,000 labeled traffic sessions** encompassing various operating systems (Windows, Ubuntu, OSX), browsers (Chrome, Internet Explorer, Firefox, Safari), and applications (YouTube, Facebook, Twitter).
    - Classification was nearly perfect for most combinations of OS, browser, and application.
- **Key Insights:**
    - Even with encryption, **network traffic patterns reveal identifiable characteristics** that can be used to classify user devices and applications.
    - **New features**, especially those related to the "bursty" behavior of browsers and SSL/TCP characteristics, significantly improve classification accuracy.
    - The study highlights the feasibility of **passive attacks** that can compromise user privacy by analyzing network traffic.
- The tuple of <OS, Browser, Application> could be classified with high accuracy using the proposed methods.

2. "Early Traffic Classification With Encrypted ClientHello: A Multi-Country Study"

- **Main Results:**
    - The paper introduces a new early traffic classification (eTC) algorithm called **hRFTC (hybrid Random Forest Traffic Classifier)**, designed to work effectively even with Encrypted ClientHello (ECH).
    - A **new, large, and diverse dataset** of over 600,000 TLS flows across 19 traffic classes was collected from multiple countries (North America, Europe, and Asia).
    - Existing algorithms based solely on TLS features only achieved an F-score as low as 38.4%, while hRFTC reached up to a **94.6% F-score**.
    - hRFTC was shown to outperform both packet-based and flow-based algorithms.
    - The study found that eTC algorithms trained in one geographical location need to be retrained when deployed in a different location.
- **Key Insights:**
    - **ECH makes eTC based only on TLS metadata very challenging**, as it conceals sensitive information like the Server Name Indication (SNI).
    - **Combining flow-based features (like packet sizes and inter-packet times) with TLS metadata significantly improves eTC accuracy**.

- **Traffic patterns differ significantly by geographic location**, necessitating location-specific training for accurate eTC.
- Packet size features are crucial for classification accuracy, though QUIC PADDING complicates traffic classification.
- Despite efforts to encrypt TLS messages, payload features still contribute about 30% to classification importance.
- hRFTC demonstrated strong generalization, with only a 7% reduction in F-score when trained on just 10% of the dataset.

3. "FlowPic: Encrypted Internet Traffic Classification is as Easy as Image Recognition"

- **Main Results:**
  - The paper proposes a novel approach for encrypted traffic classification called **FlowPic**, which converts network flows into images, which are then classified using Convolutional Neural Networks (CNNs).
  - FlowPic is able to classify different types of traffic such as VoIP, Video, Chat, Browsing, and File Transfer, with different encryption techniques like VPN and Tor.
  - The method demonstrates **strong generalization** by classifying new applications within a trained traffic category.
  - High accuracy was achieved in various tasks:
    - **85.0%** for non-VPN traffic categorization
    - **98.4%** for VPN traffic categorization
    - **67.8%** for Tor traffic categorization.
    - **88.4%** for classifying encryption techniques
    - **99.7%** for specific application identification.
- **The model could classify Facebook videos, with 99.9% accuracy even when the model did not train on Facebook data**. This shows the capacity for generalization
- **Key Insights:**
  - Representing traffic flows as images allows the use of CNNs and **avoids manual feature extraction**, while still achieving high performance.
  - FlowPic **does not rely on packet payload content**, thus maintaining privacy.
  - The method works well with **short, unidirectional traffic flows**, rather than needing entire bidirectional sessions.
  - **Traffic over Tor is harder to classify** than traffic over VPN or non-VPN.
  - The method demonstrates the capacity to identify traffic patterns, even if they are using different encryption techniques.

Overall Insights and Common Themes

- **The Challenge of Encrypted Traffic:** All three papers address the growing challenge of classifying encrypted network traffic due to the increasing use of encryption technologies like HTTPS, VPN, Tor, and ECH.
- **Machine Learning is Key:** All studies utilize various machine learning techniques (SVM, Random Forest, CNN) to classify traffic based on extracted features, demonstrating the power of these techniques for network analysis.

- **Beyond Payload:** The papers explore methods that do not rely on deep packet inspection (DPI), emphasizing the importance of using flow characteristics, statistical features, and TLS metadata for classification.
- **The Importance of Data:** All papers highlight the importance of having diverse, high-quality, labeled datasets for training and evaluating traffic classification algorithms.
- **Privacy Considerations:** These papers demonstrate the potential for both privacy-preserving and privacy-violating applications of traffic analysis, underscoring the need for responsible use of these technologies.
- **Real-world implications:** The classification of traffic is important for network management, quality of service provisioning, security, and other relevant use-cases.

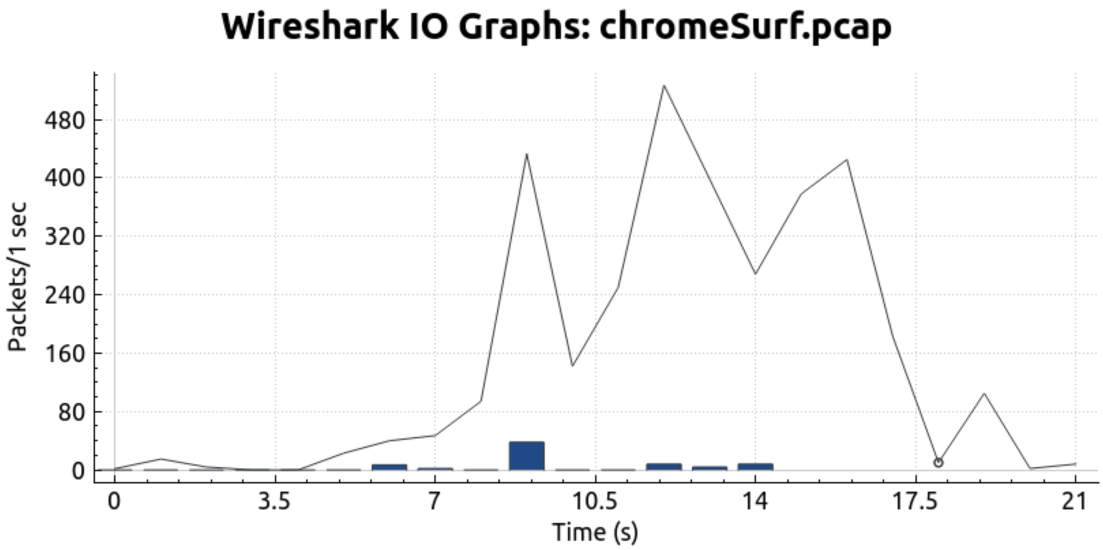
Part 3

In this part, I recorded the 4 scenarios mentioned. A Chrome surfing, Firefox surfing, audio streaming using SoundCloud, video streaming using Youtube and video conferencing in Zoom.

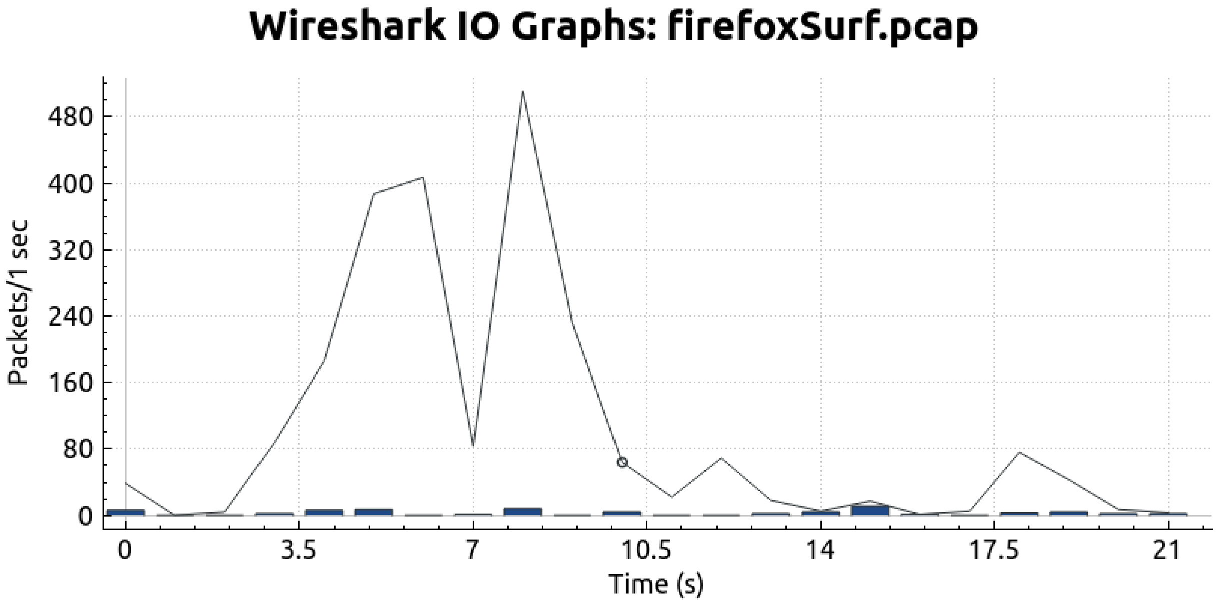
Flow Size

The next graphs would indicate how many packets were flowing in an interval time of one second. The following graphs were generated using Wireshark Plots.

Chrome: We can easily see that chrome was opened roughly at second 7 (the sharp spike in the beginning) and closed at around second 18.5. Peak was around 480 packets.



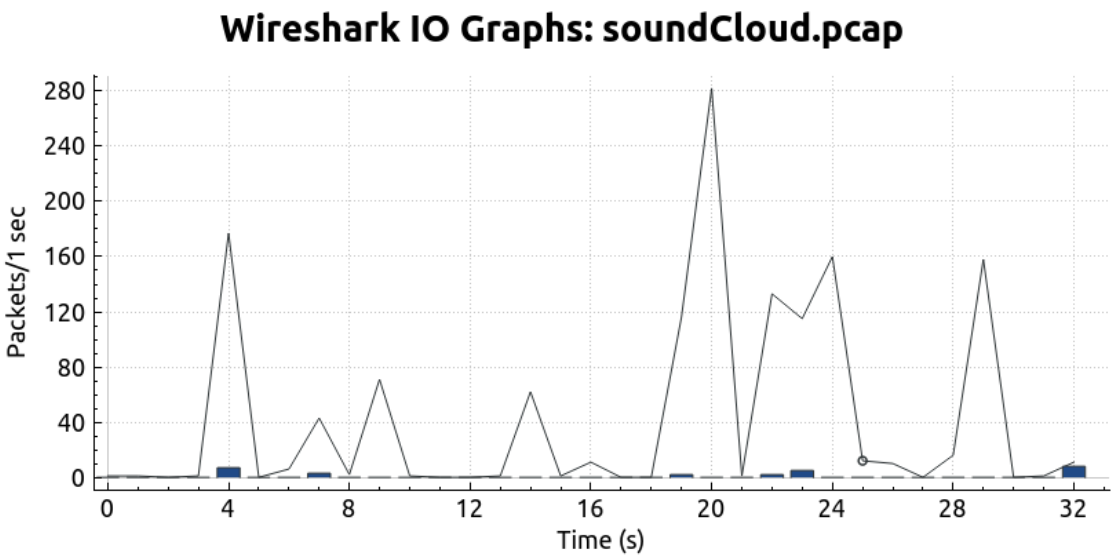
Firefox: Opened at about second 3 and closed at about second 10. Peak Value, again, around 480 packets which makes sense. This flow is very similar to the chrome Surf.



The two surfings are quite similar, The high traffic indicates a new “fetch” request – for example searching for something or clicking a result. Low traffic is when we already fetched the data and we only observe it. (Which I didn’t do for long time periods in both cases – the low traffic points last 1-2 seconds top).

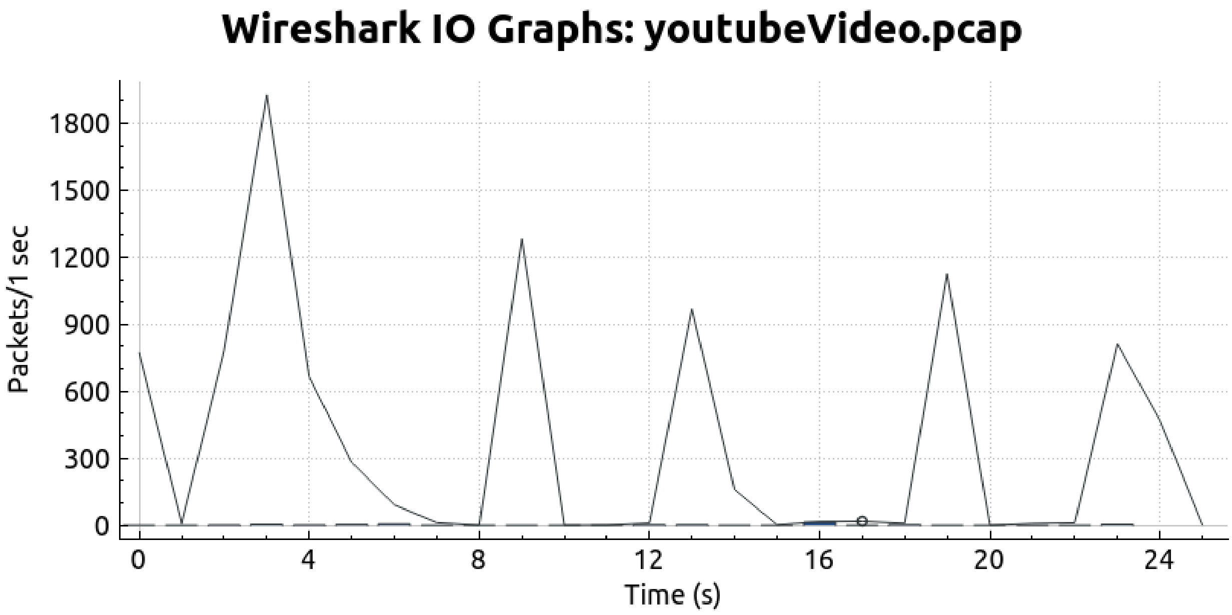
<u>SoundCloud</u>

The flow here is quite interesting, instead of a steady flow size (which one might expect), there are peaks and lows. It is reasonable to assume (and most audio applications indeed do that) that each time you fetch audio data – you fetch also the next couple of seconds – idea being that sudden network interrupts won’t affect the hearing experience. (I heard a song which was already fetched before in the beginning and only afterwards a new one – to see if the browser caches it).



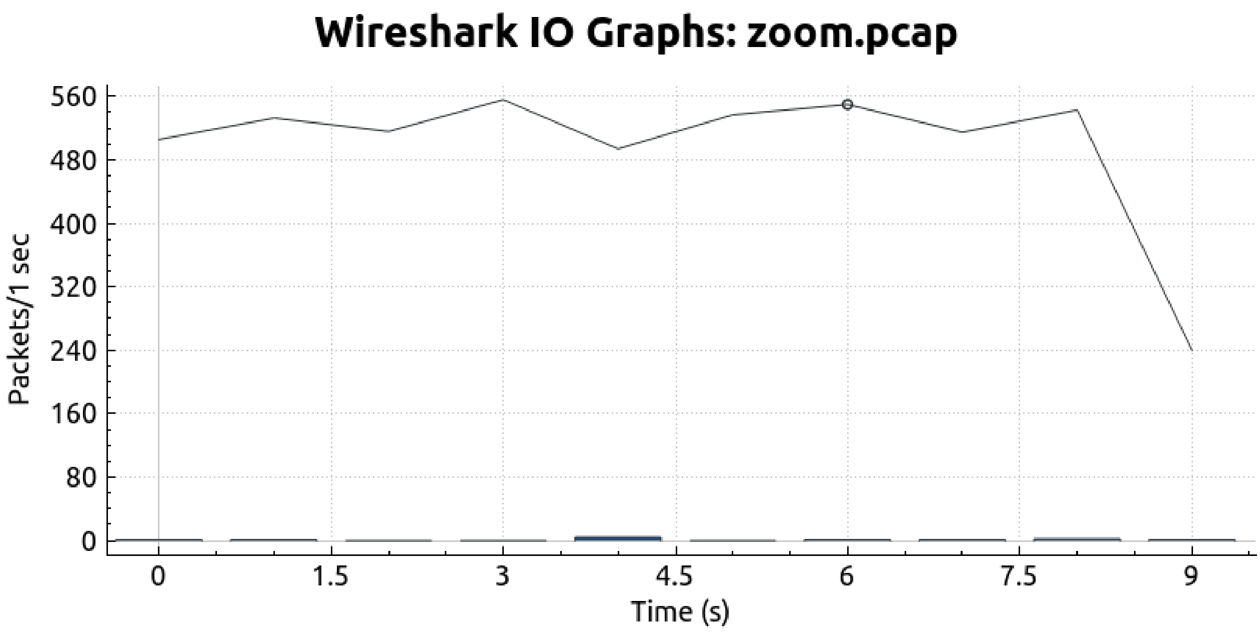
<u>Youtube</u>

We can see here the same behavior as the audio streaming, but with much more volume – which makes total sense, video streaming is much more demanding as not only audio is being transferred but also images.



Zoom

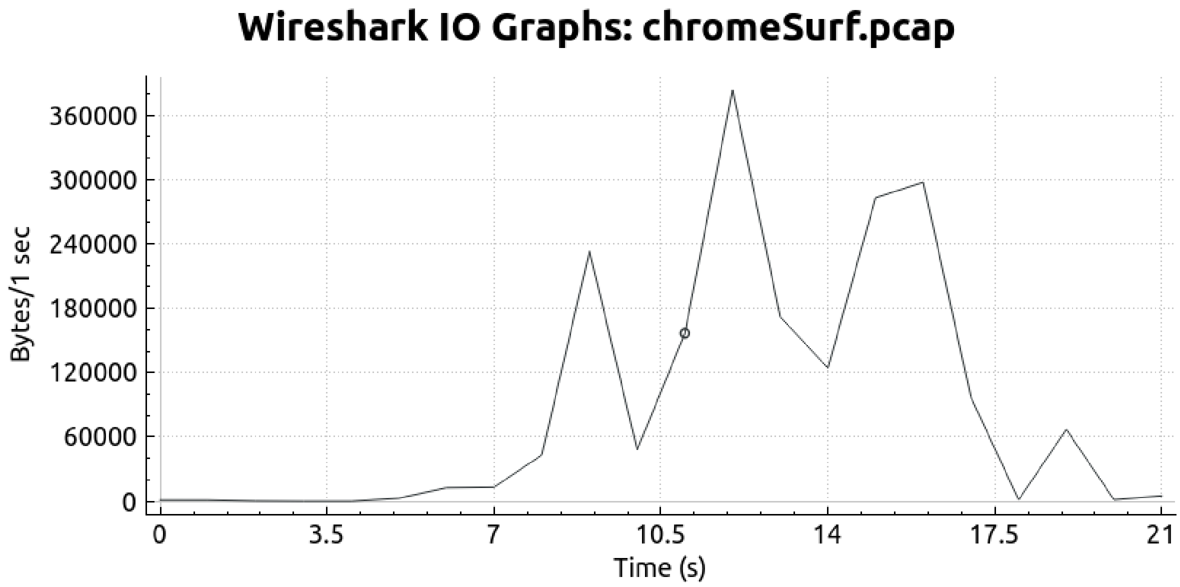
We can see here high traffic, steady volume graph. This makes total sense – the technique used by the audio and video streaming applications can't be used here – as we stream the current state at any given moment.



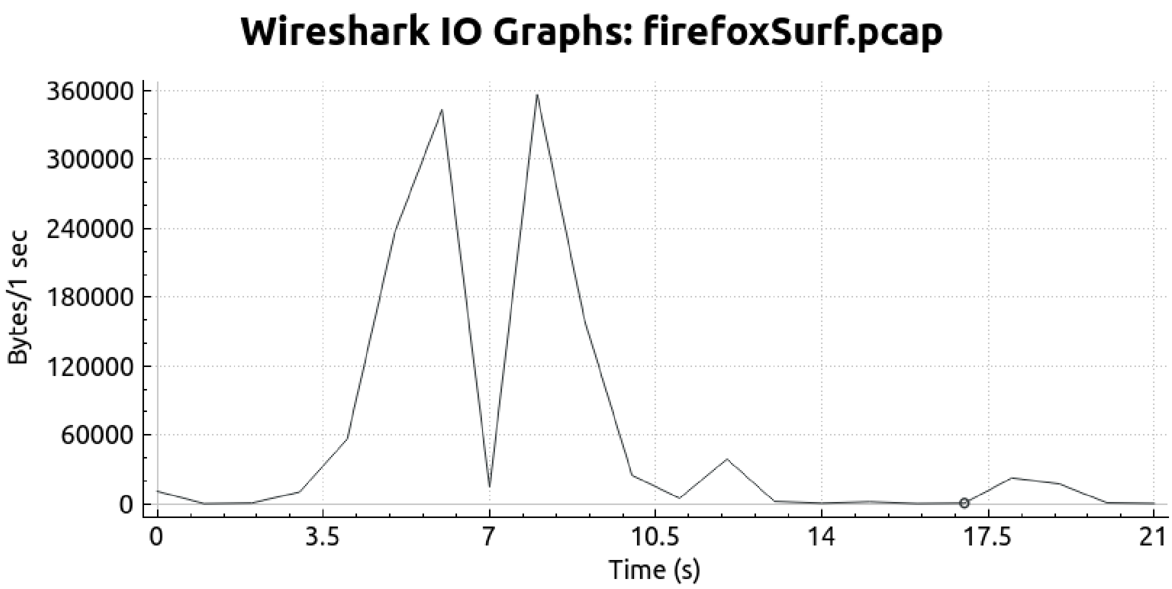
Flow Volume

These graphs represent the Bytes transferred each second. These were also produced using Wireshark plots.

Chrome

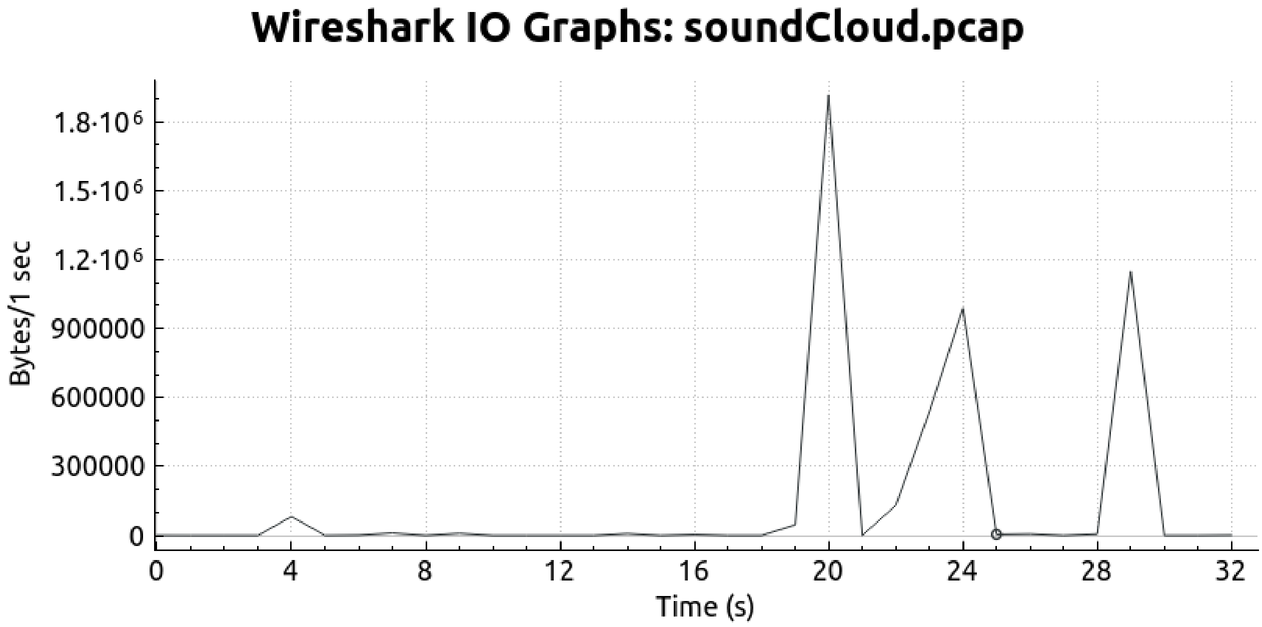


Firefox

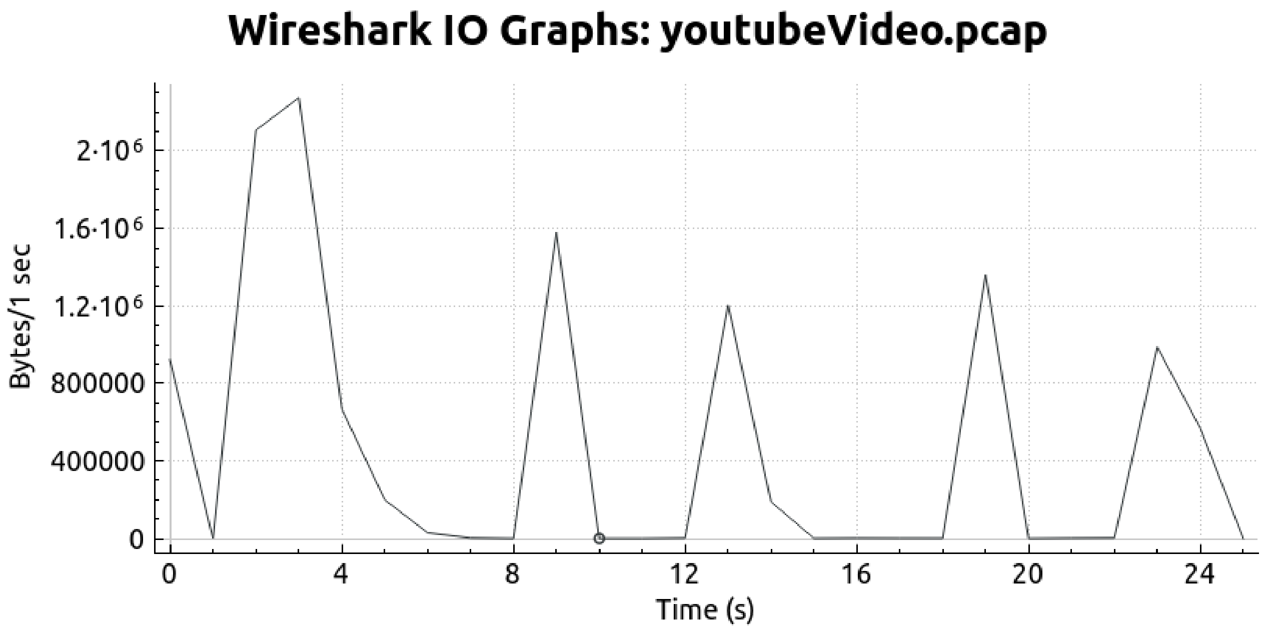


SoundCloud

We can see that the browser indeed cached the first song – there is almost no traffic in the beginning of the session.



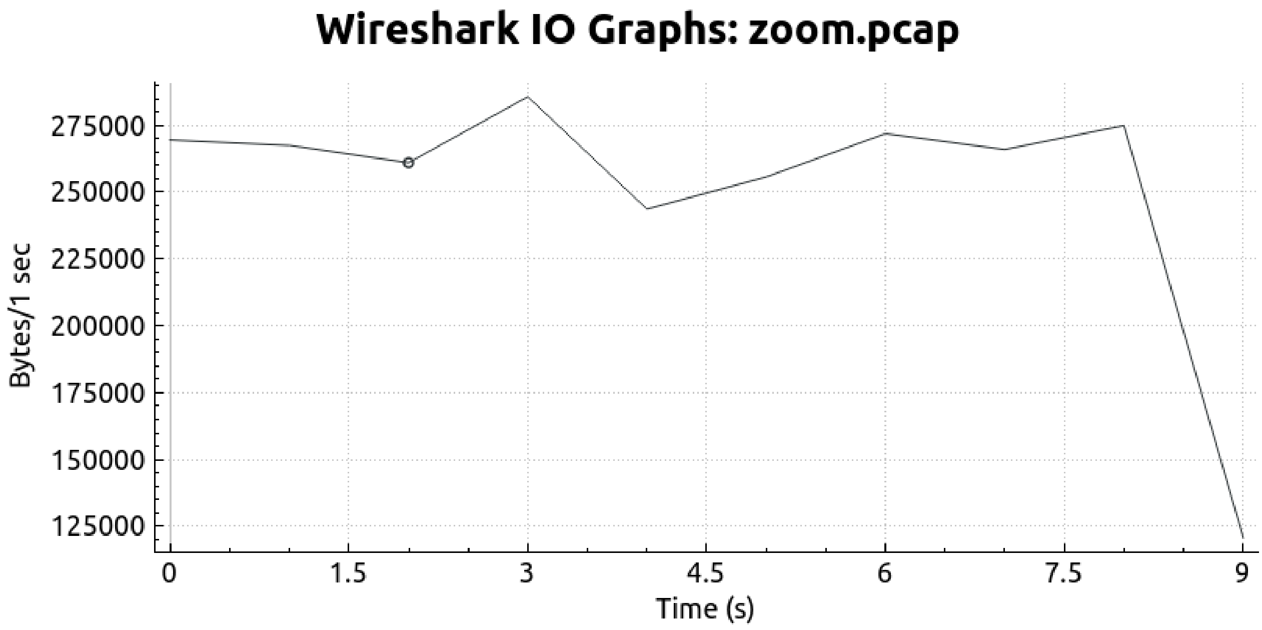
Youtube



We can see large flow volumes in the video and audio streams relative to the regular surfing.

Zoom

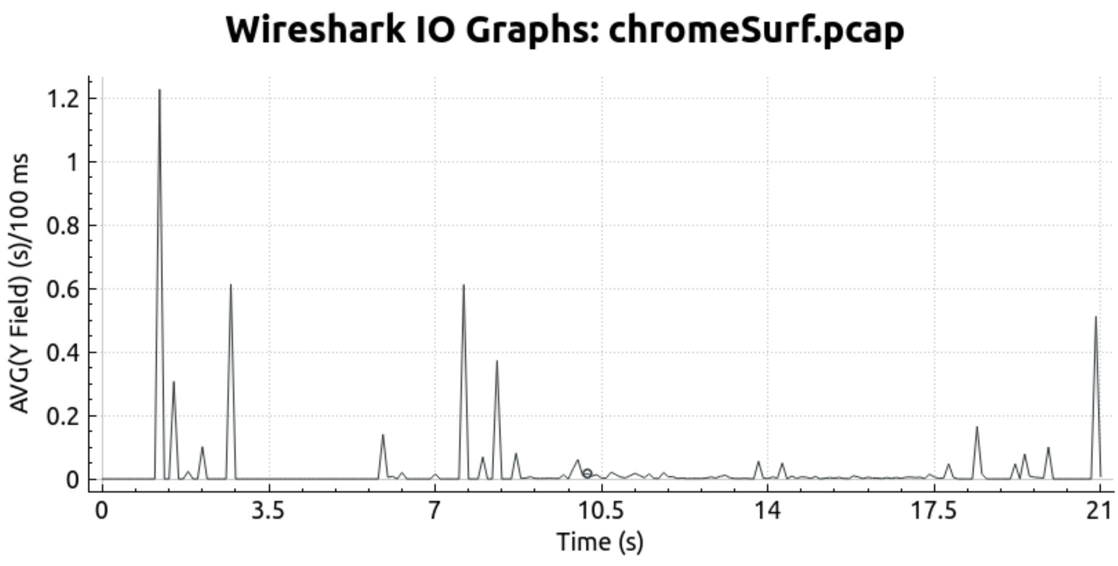
We can see here moderate to high volume but again -steady over time.



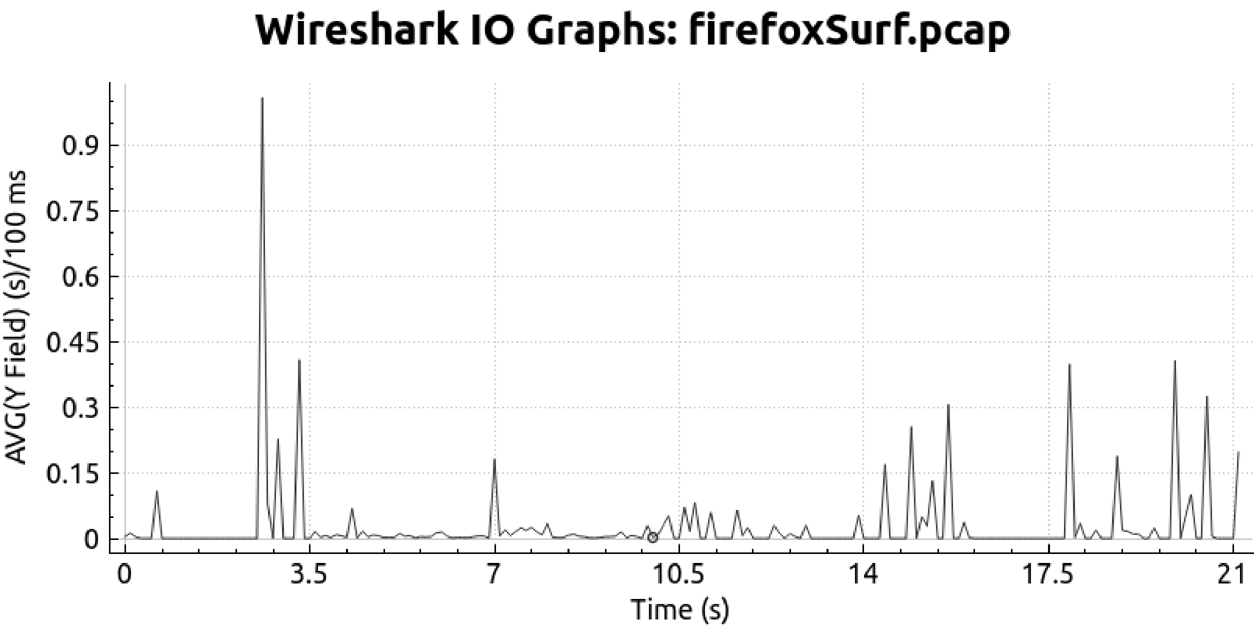
<u>Packets Inter Arrival Time</u>
To plot this in a meaningful way, I applied two types of adjustments.
1. I refined the X axis to be 0.1 seconds for finer details.
2. I used Wireshark's AVG(Y field) to show the average of the time_delta of each frame.

So, now, what we will see is the average time gap between packets in each 0.1 second window.

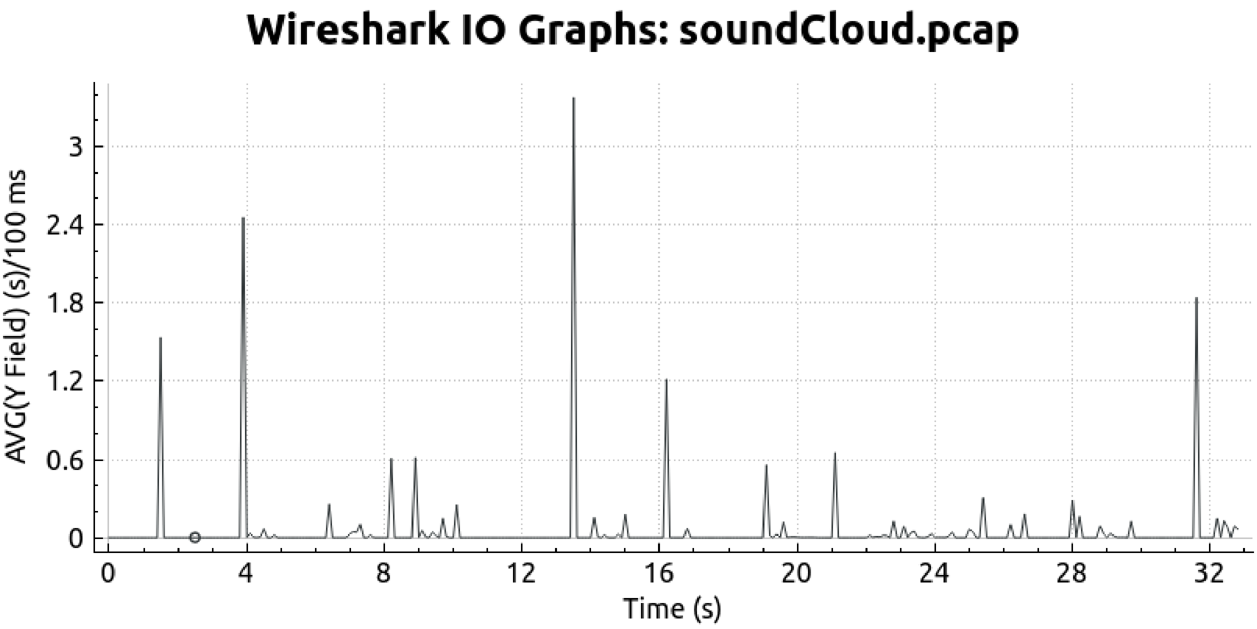
<u>Chrome</u>



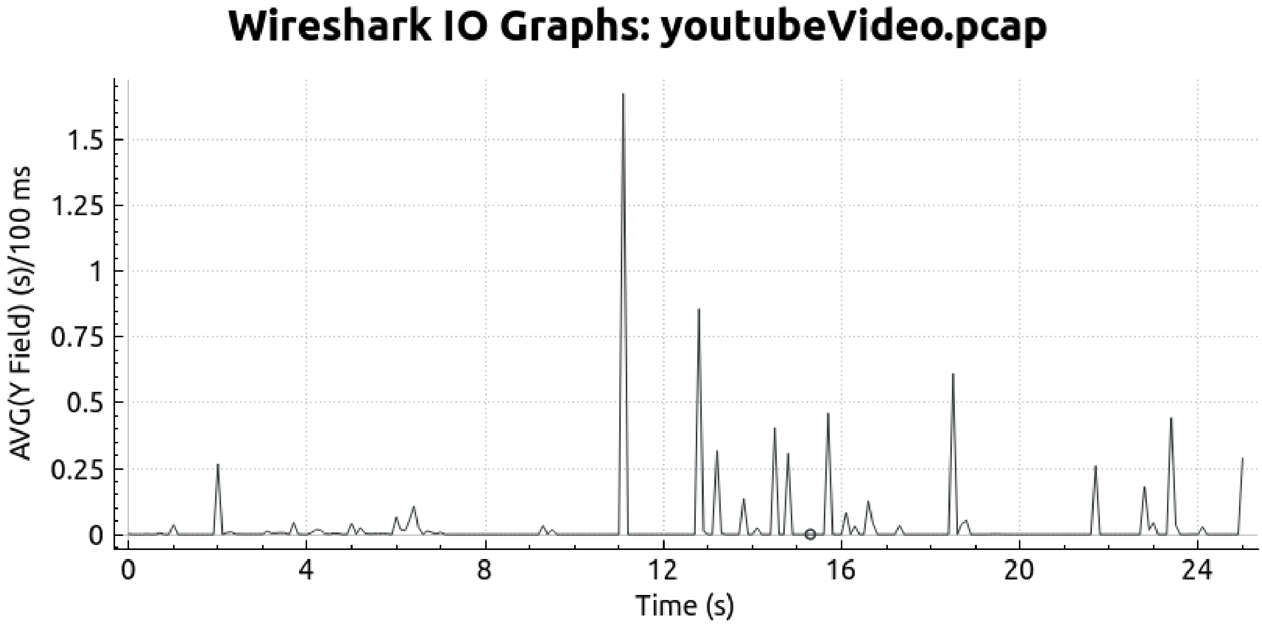
<u>Firefox</u>


Wireshark IO Graphs: firefoxSurf.pcap
AVG(Y Field) (s)/100 ms
0.9
0.75
0.6
0.45
0.3
0.15
0
0
3.5
7
10.5
14
17.5
21
Time (s)

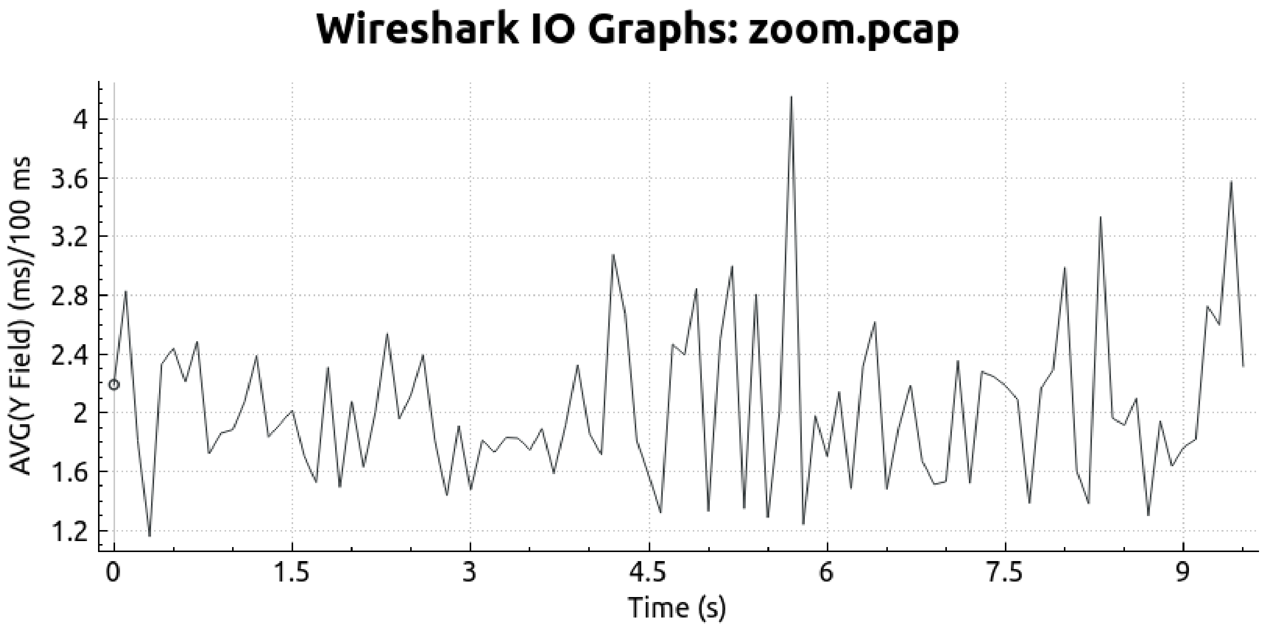
SoundCloud


Wireshark IO Graphs: soundCloud.pcap
AVG(Y Field) (s)/100 ms
3
2.4
1.8
1.2
0.6
0
0
4
8
12
16
20
24
28
32
Time (s)

Youtube



Zoom



General Explanation:

A smooth, low-value graph → Stable network with frequent packet arrivals.

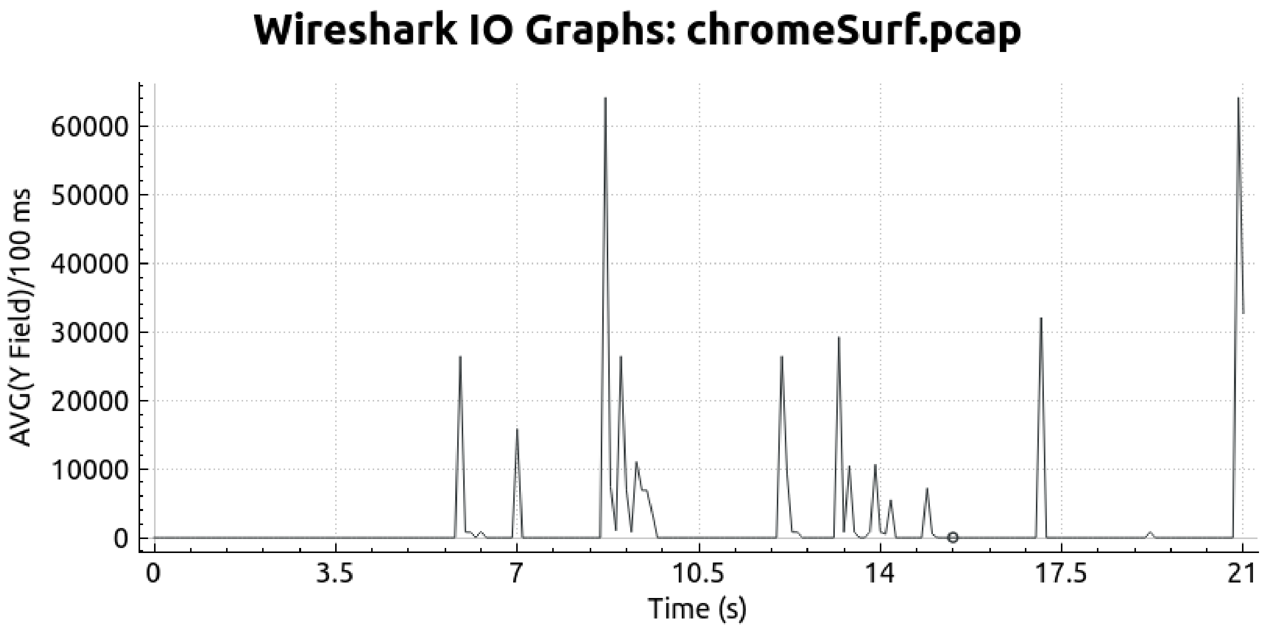
Spikes → Temporary delays, congestion, or network jitter.

Fluctuations → Bursty traffic or inconsistent packet arrival.

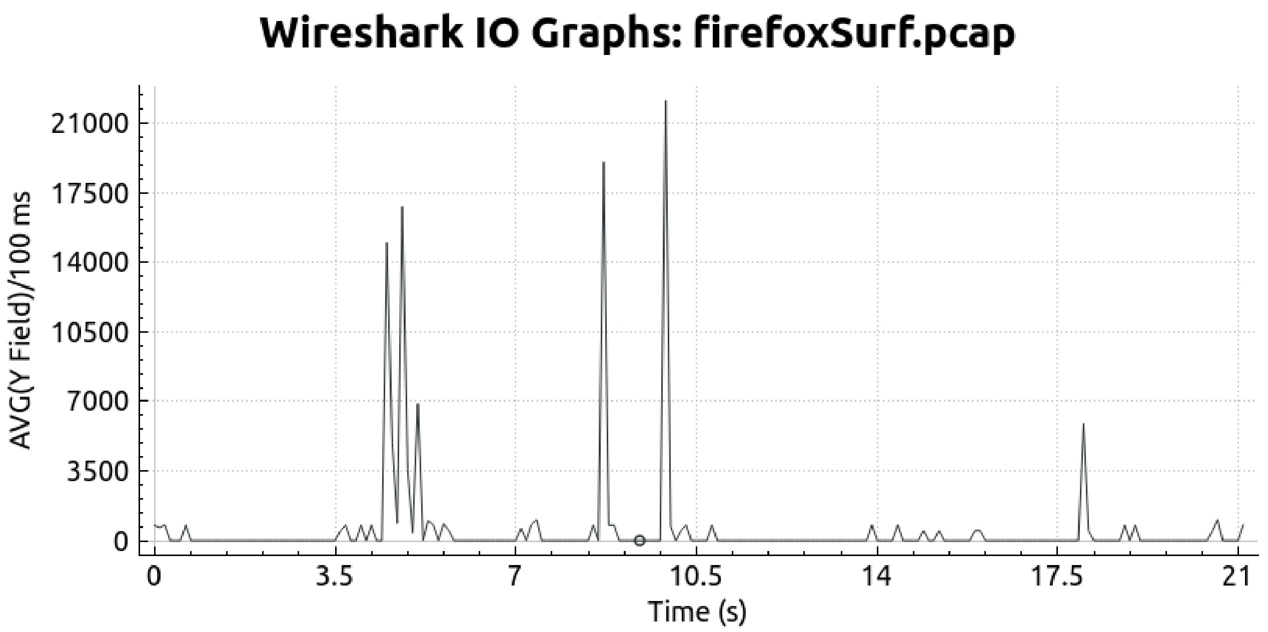
Increasing trend → Network degradation or sender slowing down.

TCP Header Field – window size
Motivation being that he TCP Window Size determines how much data can be sent before requiring an acknowledgment, it reflects flow control between sender and receiver and is useful for detecting congestion, network throttling, or performance bottlenecks.

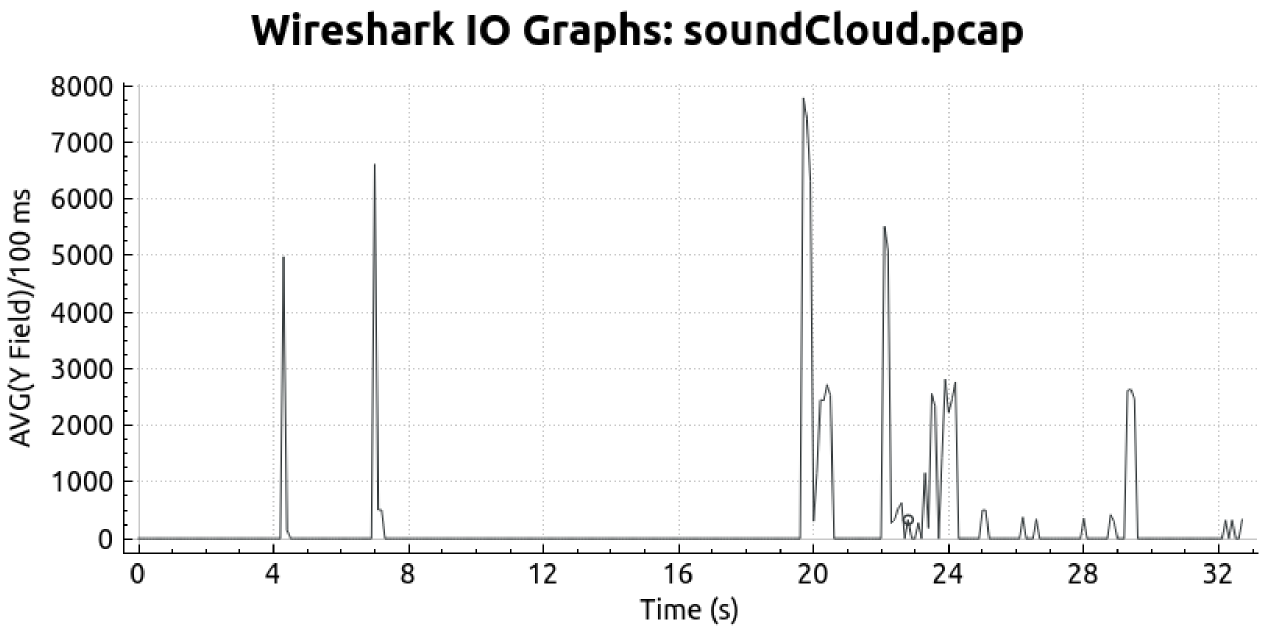
Chrome



Firefox

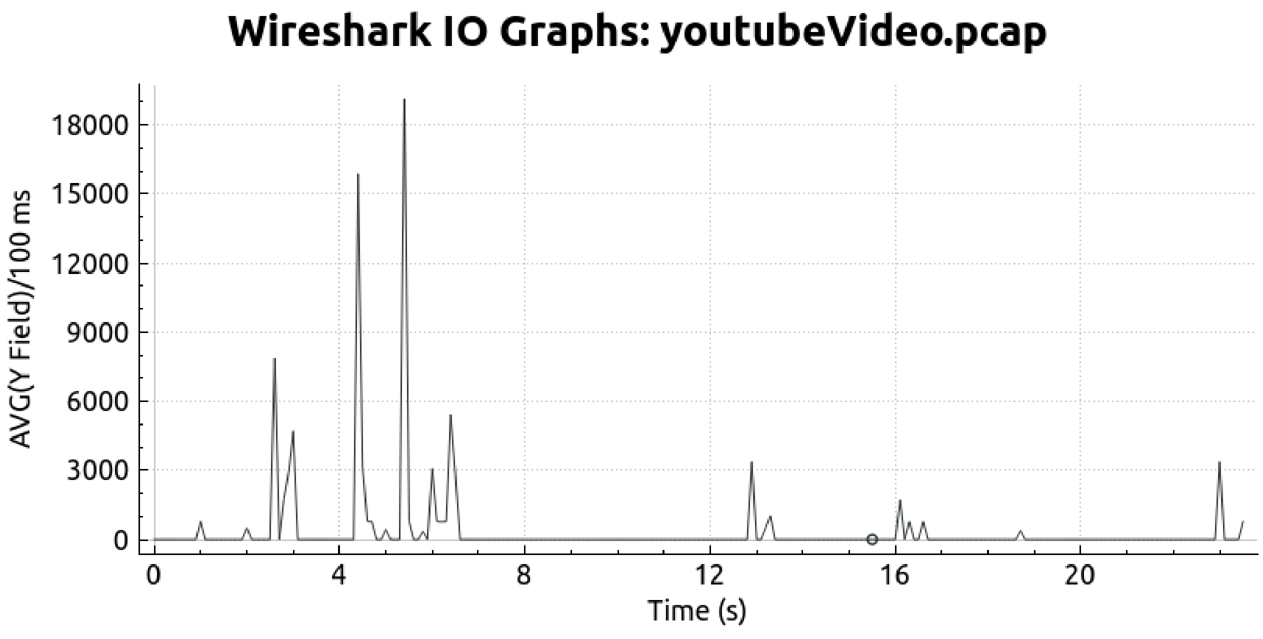


SoundCloud

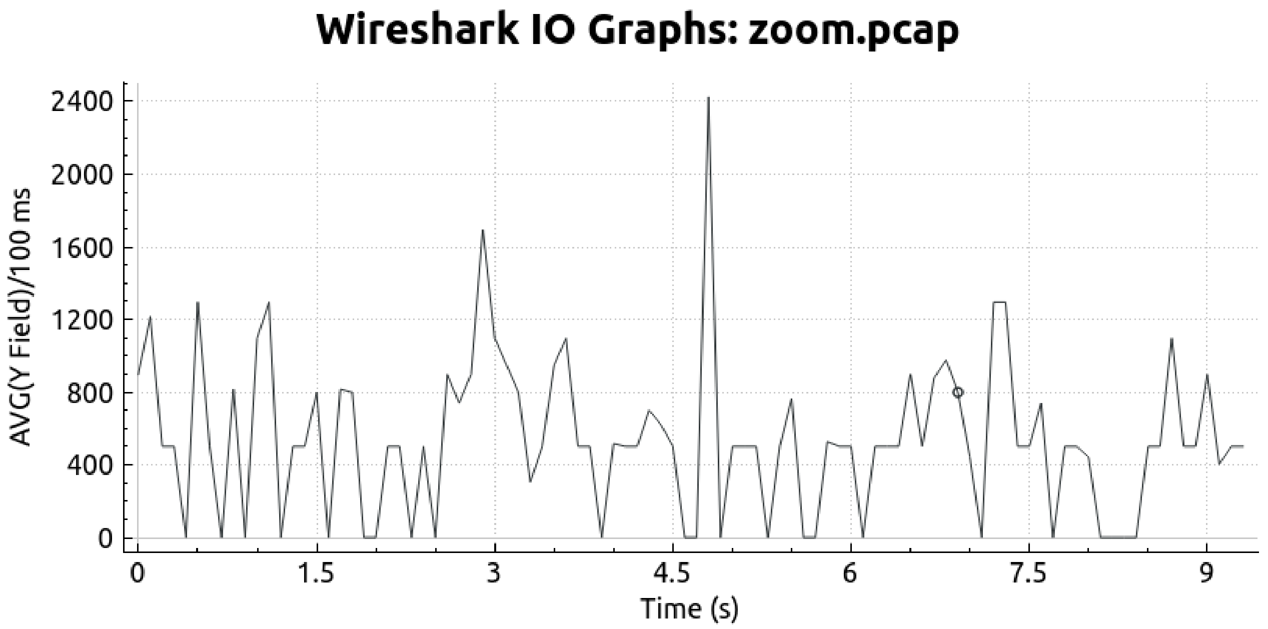


Youtube

Youtube transfers most data with UDP so most of the session the window size is low.



Zoom



General Explanation

Stable, high window size → Good network conditions, receiver can handle high data rates.

Decreasing window size → Receiver is overwhelmed or slowing down (buffer filling up).

Sudden drops to near zero → Possible network congestion, packet loss, or TCP flow control limiting transmission.

Gradual increase → TCP slow start/congestion avoidance adapting to available bandwidth.

Part 3 – Fourth Question

Scenario 1:

The hashed flow ID uniquely identifies each connection. So:

1. This allows the attacker to separate different traffic flows.

2. By analyzing packet sizes and inter-packet timing in the same manner we did in the previous part, the attacker can match patterns to known traffic fingerprints of different applications andwebsites.

Example: YouTube, as we saw, has large bursty video packets so the attacker can look for this pattern.

Scenario 2:

Without the 4-tuple hash, the attacker cannot directly separate different traffic flows.

However, they can still infer information using traffic fingerprinting techniques:

Websites and applications have unique packet size & timing patterns. For example, Chrome search sends a small query followed by a burst of responses – again – using the data we gathered in the previous section. This has lower success rate but still can be done.

Mitigation

1. Adding Noise Intentionally: if we plant fake packets in the traffic, we can confuse the attacker.

2. Multiplexing Multiple Connections: there are tools that can hide each individual connection like TLS Encrypted Client Hello.

3. Adding Random Packet Delays: again, to confuse pattern matching.

Bonus Part – Spotify And Mailing

Scenario 1:

Spotify Traffic Pattern:

1. Continuous, steady stream of medium-to-large packets which typically identify audio stream.

2. Periodic buffering spikes when Spotify preloads songs – like we saw both in audio and video stream in the previous part.

3. A single, stable flow ID exists, showing it is one persistent connection.

Email Traffic Pattern:

1. Short bursts of small packets when composing/sending emails.

2. Additional medium-sized packets if we attached files to the mail.

3. The attacker will see new short-lived flow IDs appearing when an email is sent.

Conclusion:

The attacker can clearly distinguish Spotify traffic from email traffic. The Spotify flow ID remains active with steady packet timing, while email activity appears in short lived flows.

Scenario 2:

The attacker can still match based on the characteristics above that an audio stream is active. It probably would be harder to detect mailing applications without the 4 tuple hash and seeing the connections being opened and closed each time.

Mitigations are quite similar to the previous part, we can even download the entire song we hear and than be idle for long times – hiding better the audio application.