Communication Networks - Final Project

רשימת שותפים:

- עלאא אלדין אבו חגלה, 322231580
- דוד קיטנברג, 322315300
- יואב פרץ, 211545744

Notes on the Structure of the Project (also found in `README` file)

- The project's code has several dependencies. Run the following command in order to ensure they are installed:

```
pip install scapy pyshark pandas matplotlib numpy
```
Run

- In order to generate `.pcapng` statistics plots:
  a. Place the desired PCAPNG files in the "PCAPNGs" folder in the project directory.
  b. Open the terminal in the project's directory and run the command, the output `.png` plot files will be located in the "header_analysis_results" folder in the project directory:

```
py -3 PCAPAnalyzer.py
```
Run

- In order to generate the packet timestamps, sizes and other details for the bonus section:
  a. Place the desired PCAPNG files in the "PCAPNGs/Bonus" folder in the project directory.
  b. Open the terminal in the project's directory and run the command, the output `.csv` statistics file will be located in the project directory:

```
py -3 PCAPAnalyzerBonus.py
```
Run

Background

Everyday apps, such as emailing, web surfing, and streaming, highly vary in their traffic characteristics – e.g.,
A. IP header fields.
B. TCP header fields.
C. TLS header fields.
D. packet sizes.
E. Packets inter-arrivals (the time between every two packets).
F. Flow size (namely, the overall number of packets).
G. Flow volume (namely, the overall number of bytes transmitted).

Learning traffic characteristics has several important applications. For instance, the telecom/internet provider can use it to prioritize traffic and improve user experience. An attacker can use it to learn the user's activity, thus mitigating privacy. Even if the traffic is encrypted, an attacker who can view the encrypted packets can learn from them which apps the victim used at each moment.

In this project, we'll perform an introductory study of the traffic characteristics of the packets that we use.

Part I: Answer the following Questions:

Question 1

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?

Solution

When a file transfer is slow and we suspect an issue at the transport layer (most commonly TCP), we typically look at how TCP is handling congestion control, flow control, and error recovery between the source and destination. Let's delve into the key factors that can contribute to slow file transfers, and attempt to troubleshoot each one:

Window Size and Flow Control

What to look for:

- TCP window size: If the receiver's advertised window or the sender's congestion window is too small, data will be sent in small bursts, limiting throughput.
- Window scaling: On high-bandwidth or high-latency networks, a lack of TCP window scaling can severely limit throughput.

How to troubleshoot:

- We capture packets with Wireshark (or similar software) and check the `Window Size` (in the TCP header) and `Window Scaling` options during the TCP three-way handshake.
- We ensure both endpoints (sender and receiver) support TCP window scaling if the bandwidth-delay product is large (e.g., long-distance or high-speed links).

```
⊞ Frame 639: 60 bytes on wire (480 bits), 60 bytes captured (480 bits) on interface 0
⊞ Ethernet II, Src: Raspberr_68:1e:36 (b8:27:eb:68:1e:36), Dst: AsustekC_7d:22:8c (74:d0:2b:7d:22:8c)
⊞ Internet Protocol Version 4, Src: 10.56.100.164 (10.56.100.164), Dst: 10.56.100.1 (10.56.100.1)
⊟ Transmission Control Protocol, Src Port: 22 (22), Dst Port: 56748 (56748), Seq: 2520, Ack: 51956, Len: 0
    Source Port: 22 (22)
    Destination Port: 56748 (56748)
    [Stream index: 16]
    [TCP Segment Len: 0]
    Sequence number: 2520    (relative sequence number)
    Acknowledgment number: 51956    (relative ack number)
    Header Length: 20 bytes
  ⊟ .... 0000 0001 0000 = Flags: 0x010 (ACK)
      000. .... .... = Reserved: Not set
      ...0 .... .... = Nonce: Not set
      .... 0... .... = Congestion Window Reduced (CWR): Not set
      .... .0.. .... = ECN-Echo: Not set
      .... ..0. .... = Urgent: Not set
      .... ...1 .... = Acknowledgment: Set
      .... .... 0... = Push: Not set
      .... .... .0.. = Reset: Not set
      .... .... ..0. = Syn: Not set
      .... .... ...0 = Fin: Not set
    Window size value: 2070
    [Calculated window size: 132480]
    [Window size scaling factor: 64]
⊞ Checksum: 0x102f [validation disabled]
  Urgent pointer: 0
⊞ [SEQ/ACK analysis]
```

file-20250127193830847.webp

Packet Loss and Retransmissions

What to look for:

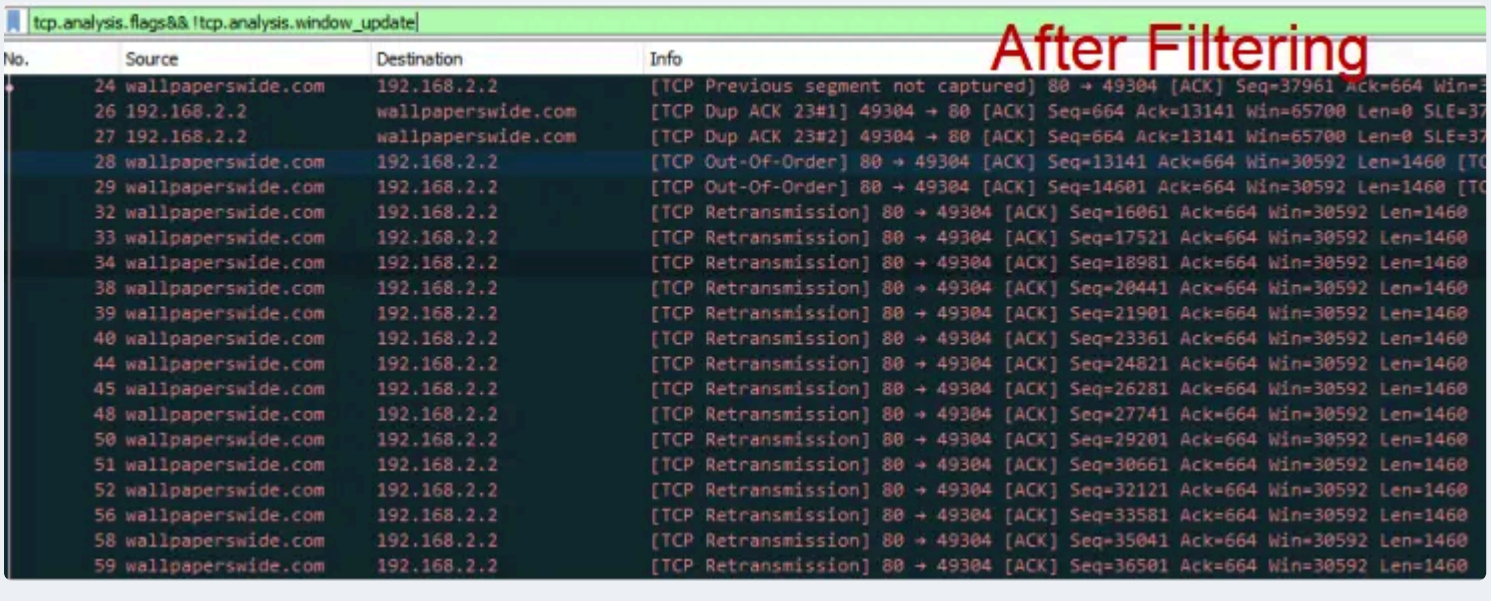
- Frequent packet losses trigger TCP retransmissions and congestion control mechanisms, slowing overall transfer.
- High Retransmission Rate in the packet capture is a red flag for problems on the network path.

How to troubleshoot:

a. In Wireshark, we look for "TCP Retransmission" or "TCP Fast Retransmission" or "Duplicate ACK" messages.

b. We identify the loss cause:

- Physical layer issues (bad cables, faulty ports, wireless interference).
- Congestion in the network path.
- Errors at intermediate network devices (routers, switches).



file-20250127194441920.webp

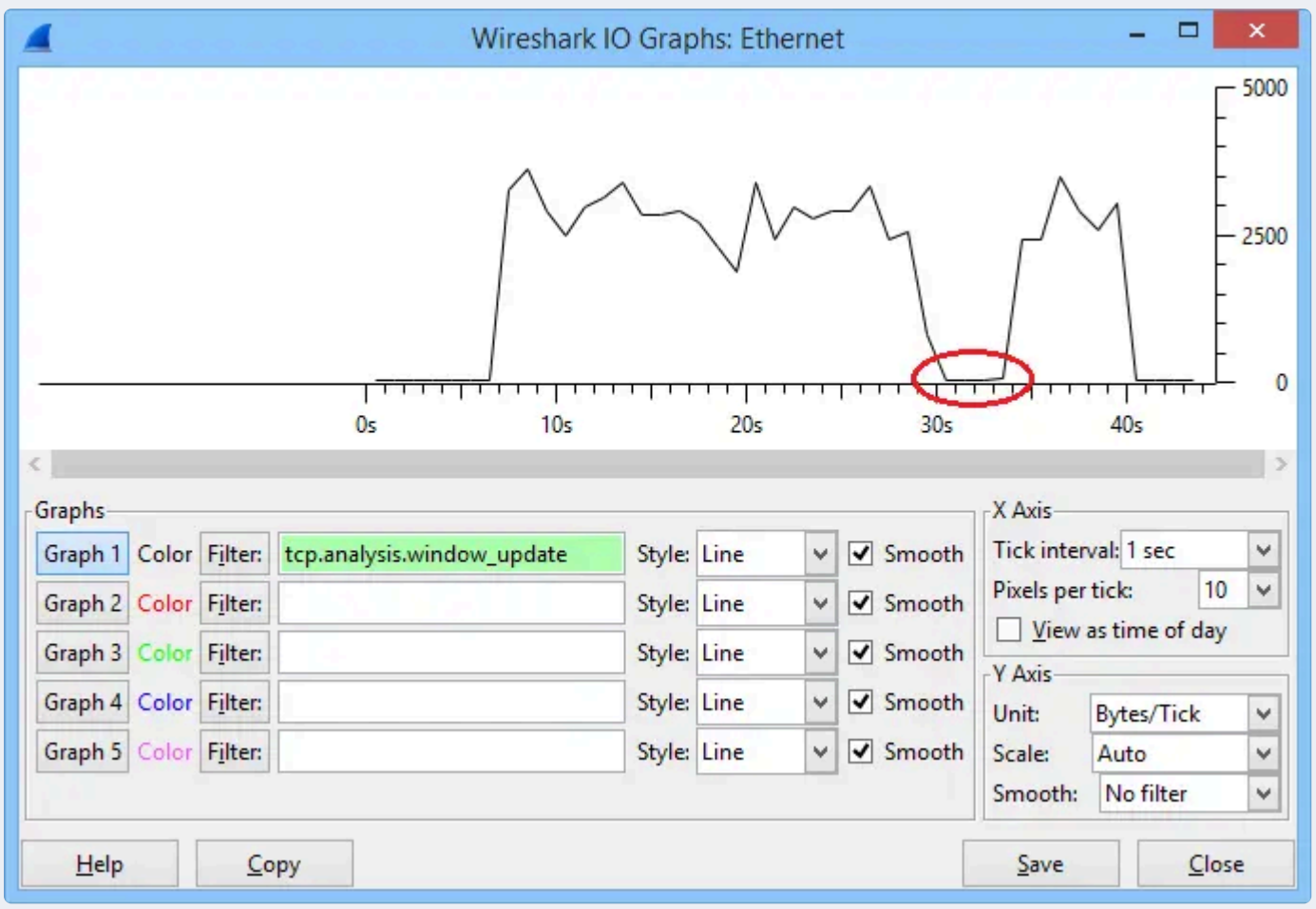
TCP Congestion Control

What to look for:

- Congestion control algorithm: Modern systems often use CUBIC, New Reno or other congestion control algorithms. Older or misconfigured systems might use outdated algorithms, resulting in suboptimal throughput.
- Slow Start and Congestion Avoidance phases: If the connection repeatedly resets to Slow Start (due to packet loss or timeouts), throughput will remain low.

How to troubleshoot:

- We examine the TCP states (Slow Start, Congestion Avoidance, etc.) by monitoring the sender's congestion window growth in a packet capture.
- We check for repetitive slow start: This happens when there is consistent packet loss or if the transfer has frequent idle periods causing TCP to reset.



file-20250127195126977.webp

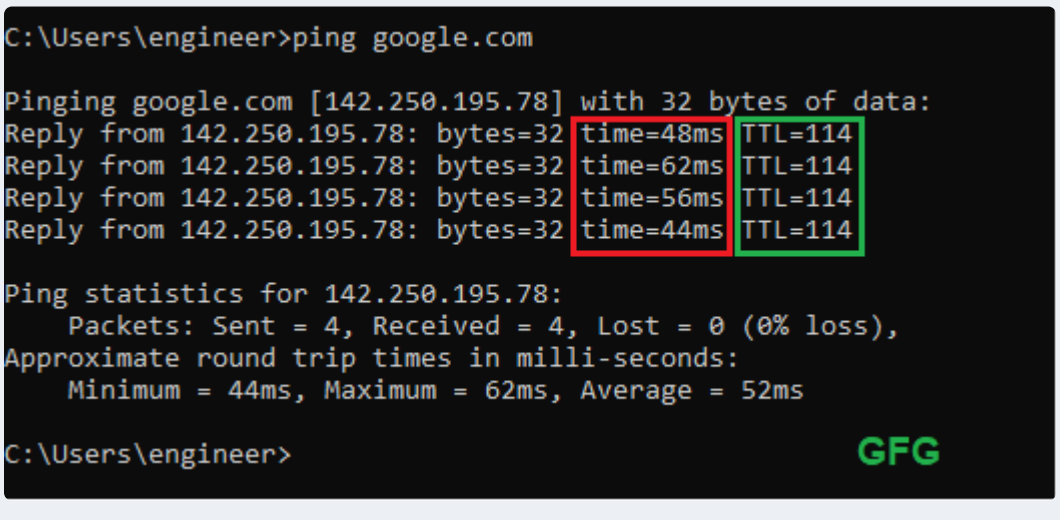
Latency and RTT (Round Trip Time)

What to look for:

- High latency can significantly reduce throughput in TCP, especially if the window size is not scaled appropriately.
- Network path might have suboptimal routing, or certain network segments might introduce large delays.

How to troubleshoot:

- We measure RTT using tools like `ping` or by looking at the TCP handshake in Wireshark (look at the time between the SYN and SYN-ACK).
- We check for route changes or issues by using `traceroute` or a similar tool to see if there are large delays at specific hops.
- We may optimize window sizes (enable window scaling) or use protocols designed for high-latency networks if needed.



file-20250127195503053.webp

```
Tracing route to www.google.com [216.58.201.164]
over a maximum of 13 hops:

  1     1 ms     1 ms     1 ms  192.168.0.1
  2     1 ms     1 ms     1 ms  192.168.5.1
  3     5 ms     5 ms     2 ms  192.168.170.1
  4     8 ms    10 ms     7 ms  172.26.199.169
  5   209 ms     8 ms    11 ms  79.red-81-46-37.customer
  6    17 ms    28 ms    18 ms  65.red-81-46-68.customer
  7     *        *        *     Request timed out.
  8     *        *        *     Request timed out.
  9    20 ms    20 ms    19 ms  176.52.253.97
 10    19 ms    20 ms    21 ms  72.14.211.154
 11     *        *        *     Request timed out.
 12    20 ms    21 ms    30 ms  74.125.253.198
 13    23 ms    20 ms    20 ms  74.125.242.179
```

file-20250127195543116.webp

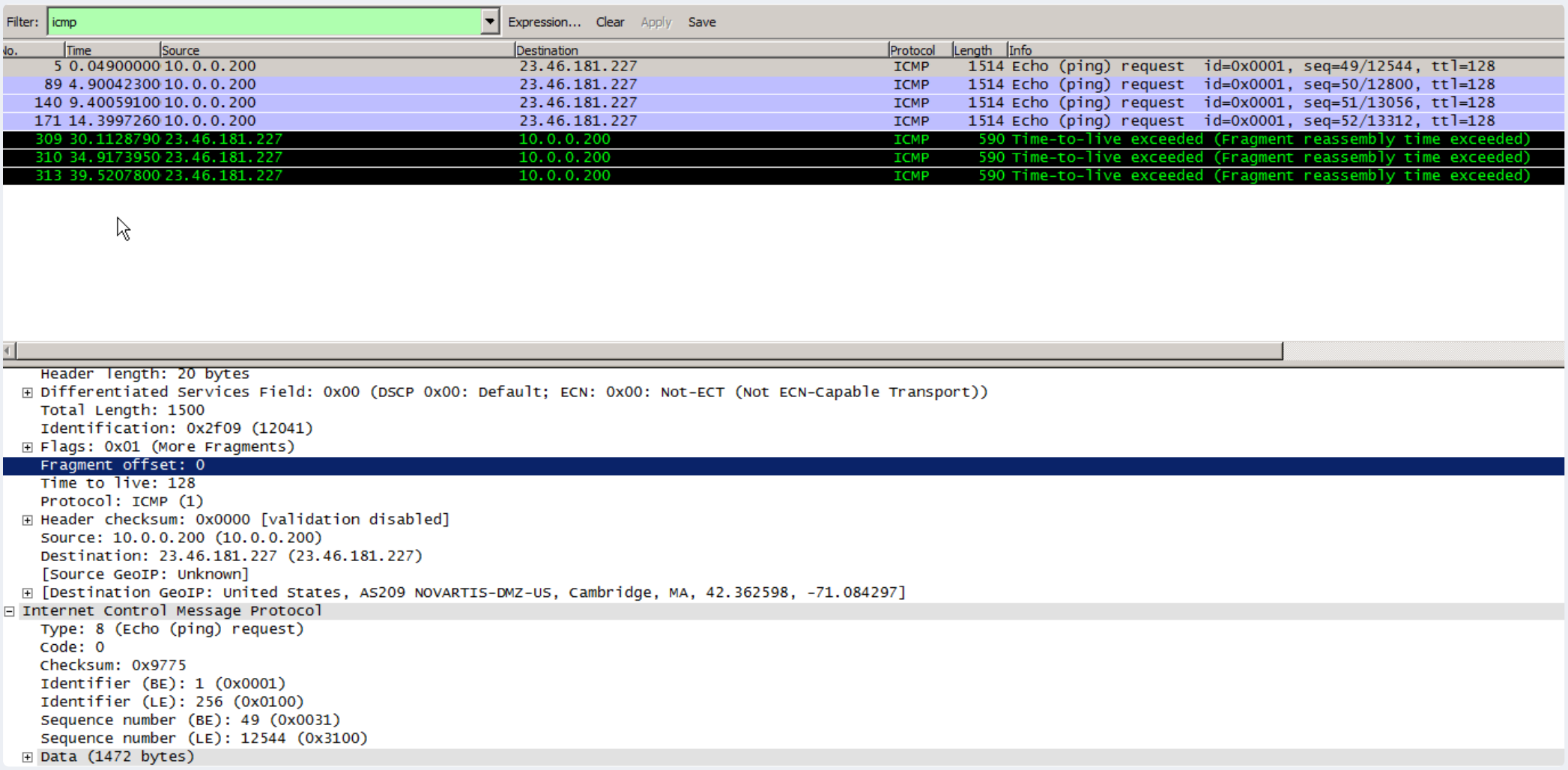
Path MTU and Fragmentation

What to look for:

- If the Path Maximum Transmission Unit (PMTU) is incorrectly set and large packets get fragmented (adds overhead) or dropped, throughput suffers.
- Misconfigured MTU on one of the network segments can cause excessive fragmentation or dropped packets.

How to troubleshoot:

- We check if PMTU Discovery is working: In Wireshark, we look for ICMP “Fragmentation Needed” messages or large packets being retransmitted.
- We may configure consistent/fixed MTU across all network interfaces on the path.



file-20250127195950122.webp

Network/Device Congestion or Rate Limiting

What to look for:

- Network devices (routers, firewalls, NAT gateways) might be overloaded or have throughput limits.
- QoS (Quality of Service) policies that deprioritize or limit the traffic we are testing.

How to troubleshoot:

- We look at link utilization on all intermediate devices (such as router interfaces). A heavily utilized link can cause congestion and packet loss.
- We may disable or check QoS settings to see if the traffic in question is deprioritized.

Question 2

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?

Solution

TCP's flow control mechanism is designed to ensure that a fast sender does not overwhelm a slower receiver. It does this by having the receiver advertise a 'window' (the receive window) that indicates how much data it can buffer at any given time. The sender uses this advertised window to determine how much data it can send before it must stop and wait for an update from the receiver.

Key Effects on Performance

- Matching Receiver Capacity
  - If the receiver has significantly lower processing power or limited buffering capabilities, it may advertise a *smaller* receive window. This essentially throttles/slows down the sender's transmission rate, preventing buffer overflows at the receiver.
- Sender Starvation
  - When the receiver's window shrinks too much (or goes to zero), the sender must pause sending new data until the receiver indicates that it can accept more. Even if the sender has a lot of data to send and powerful resources, it is forced to wait—leading to *underutilization* of the sender's capacity.
- Throughput Bottleneck
  - In cases of extreme mismatch (e.g., a powerful server sending to a small embedded device, such as a fingerprint scanner in a scenario of matching with fingerprints stored in faraway databases), the overall throughput will be limited by how quickly the receiver can process data and free up its buffer. No matter how high the sender's bandwidth or CPU capacity, the transfer speed cannot exceed the rate that the receiver can handle.
- Window Scaling
  - On modern networks, TCP Window Scaling can mitigate some performance issues by allowing larger receive windows. However, if the receiver truly lacks processing power (rather than just buffer space), it still cannot keep up with a fast sender —even with a theoretically large buffer.

Question 3

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?

Solution

Role of Routing in Multi-Path Networks

- Path Determination and Selection
  - Routing protocols (e.g., OSPF, IS-IS, BGP) are responsible for discovering and maintaining information about available network paths. They compute the best path(s) based on various metrics, ensuring that data packets follow the best (shortest, smallest delay, etc.) route from source to destination.
- Load Balancing and Redundancy
  - In environments where multiple paths exist, routing can distribute traffic across several routes to avoid congestion. This load balancing not only optimizes network resource utilization but also provides redundancy—if one path fails, alternative routes can be used to maintain connectivity.
- Quality of Service (QoS) Management
  - Advanced routing decisions can factor in QoS requirements, prioritizing latency-sensitive or high-priority traffic over the best available path, this ensures that critical applications (real-time video for example) receive the necessary performance levels.

Impact of Path Choice on Network Performance

- Latency and Delay
  - Shorter or less congested paths result in lower latency, which is critical for real-time applications such as VoIP and video conferencing.
  - A suboptimal path, even if available, might introduce extra delay, jitter, or packet reordering.
- Bandwidth and Throughput
  - Routes with higher available bandwidth can support greater data throughput.
  - If a path has limited bandwidth or is heavily loaded, it can become a bottleneck, affecting overall transfer speeds and limiting throughput potential.
- Reliability and Packet Loss
  - Some paths may be more stable or have better error rates.
  - Choosing a route with higher reliability minimizes packet loss, which is especially important for protocols like TCP that adjust their transmission rate based on perceived network conditions.
- Congestion Levels
  - A path that is frequently congested can lead to increased delays and packet drops.

    - Dynamic routing protocols can monitor congestion and reroute traffic to mitigate such issues.
- Security and Policy Constraints
    - Certain paths might be preferred or avoided based on security policies, regulatory constraints, or performance isolation needs.
    - Some networks might designate specific routes for sensitive data to ensure security and data integrity.

Factors to Consider in Routing Decisions

- Link Metrics and Costs
    - Hop Count: The number of routers between the source and destination can influence delay.
    - Latency/Delay: Measured delay is often used as a metric to choose the fastest path.
    - Bandwidth: Paths with higher bandwidth can handle larger volumes of data.
    - Packet Loss Rate: Routes with lower loss rates are preferred for reliable data delivery.
- Network Congestion and Utilization
    - Real-time monitoring of link utilization can help in choosing less congested paths.
    - Dynamic routing algorithms adjust paths based on current traffic conditions to optimize performance.
- Policy and Administrative Constraints
    - Routing decisions might be influenced by network policies, such as preferring certain service providers or avoiding specific networks.
    - Quality of Service (QoS) policies may prioritize traffic types differently based on application's requirements.
- Scalability and Stability of the Route
    - Some routes may be more scalable and stable over time ("future proof"), reducing the need for frequent changes that could disrupt traffic.
    - Routes that are frequently changing or experiencing intermittent failures are less desirable.
- Security Considerations
    - Paths that traverse networks with known security issues or inadequate protections may be avoided.
    - Security protocols might dictate routing decisions to ensure data integrity and confidentiality/privacy.

Question 4

How does MPTCP (Multi-path TCP) improve network performance?

Solution

MPTCP (Multi-path TCP) extends the traditional TCP protocol to allow a single connection to utilize multiple network paths simultaneously. This capability can lead to several improvements in network performance. We'll mention some of these improvements:

Increased Throughput

- Bandwidth Aggregation:
    - MPTCP enables the simultaneous use of multiple network interfaces (such as WiFi and cellular). By aggregating the bandwidth available on different paths, a single connection can achieve higher overall throughput than would be possible with a single path.
- Resource Utilization:
    - When one path becomes congested or has limited capacity, MPTCP can shift additional data onto less congested paths, such that all available network resources are used to the best effect.

Enhanced Resilience and Reliability

- Fault Tolerance:
    - If one of the multiple paths fails or degrades in quality, MPTCP can continue transmitting data over the remaining paths. This built-in redundancy helps maintain the connection without interruption.
- Seamless Handover:
    - In mobile environments, devices often switch between different networks (for example, moving from WiFi to cellular). MPTCP can manage these transitions seamlessly/effortlessly, such that ongoing sessions are maintained without interruption or the need for reconnection.

Improved Congestion Control

- Dynamic Traffic Balancing:

    - MPTCP continuously monitors the performance of each subflow (individual TCP connection over a specific path) and adjusts the data distribution based on current congestion levels and path quality. This dynamic balancing helps avoid congestion on any single path and improves overall performance.
- Optimized Data Transfer:
    - By intelligently shifting traffic to paths with lower congestion, MPTCP can reduce delays and packet loss, leading to more stable data transfers.

Robustness in Heterogeneous/Mixed Networks

- Handling Diverse Path Characteristics:
    - Networks often have paths with very different characteristics (e.g., high-bandwidth but high-latency vs. low-bandwidth but low-latency). MPTCP can manage these differences by using each path for the type of traffic it handles best, thereby optimizing overall connection performance. This makes MPTCP a "smart" protocol with builtin decision making.
- Adaptability to Variable Network Conditions:
    - MPTCP can react to changes in network conditions in real time, such as a sudden drop in performance on one path, by reallocating traffic to maintain the best possible connection quality.

For more details

Most of the information used in this subsection has been derived from the source: RFC 8684 - TCP Extensions for Multipath Operation with Multiple Addresses. This source contains a lot more details about MPTCP.

Question 5

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.

Solution

Potential Causes

At the Network Layer

- Congestion and Buffer Overflow
    - Cause: Routers have finite buffer sizes. When traffic exceeds the buffering capacity, packets are dropped.
    - Impact: High utilization or sudden bursts of traffic can overwhelm the router’s queues, leading to packet loss.
- Hardware or Physical Layer Issues
    - Cause: Faulty cables, damaged ports, or failing network interface cards (NICs) can corrupt or drop packets.
    - Impact: Physical defects can result in high error rates (such as CRC errors) that lead the router to discard packets.
- MTU Mismatches and Fragmentation Problems
    - Cause: If routers are configured with different Maximum Transmission Unit (MTU) sizes, packets that exceed the smaller MTU may be fragmented or dropped if fragmentation isn’t properly handled.
    - Impact: Mismatches can result in fragmentation failures or dropped packets when “Don’t Fragment” (DF) flags are set.
- Routing Misconfigurations or Suboptimal Paths
    - Cause: Incorrect routing entries or misconfigured Quality of Service (QoS) policies may force traffic over congested or suboptimal paths.
    - Impact: Traffic may traverse a route that cannot handle the volume, leading to packet loss due to congestion or policy-based drops.
- Interface Errors and Overruns
    - Cause: High error rates on interfaces—such as collisions, CRC errors, or buffer overruns—can indicate that the router’s hardware is struggling to process incoming data.
    - Impact: These errors lead to the dropping of corrupted or excess packets.

At the Transport Layer

- TCP Congestion Control Reactions
    - Cause: TCP interprets packet loss as a sign of congestion. When loss is detected, mechanisms such as slow start and congestion avoidance reduce the sender’s transmission rate.
    - Impact: Even if the underlying issue is at the network level, TCP’s congestion control can exacerbate performance issues by continually throttling throughput, resulting in further delays and retransmissions.
- Misconfigured TCP Parameters

- Cause: Inappropriate settings for TCP window sizes, retransmission timeouts, or lack of window scaling on high-bandwidth or high-latency links can lead to wasteful data transfer.
- Impact: These misconfigurations can contribute to suboptimal performance, making the system more sensitive to packet loss.

Recommended Resolution Steps

For Network Layer Issues

- We may examine Router Interface Statistics
  - Action: We can use monitoring tools (e.g., SNMP, router CLI commands) to check for interface errors such as CRC errors, collisions, or buffer overruns.
  - Goal: To identify if physical layer problems or hardware issues are contributing to the packet loss.
- We check for Congestion
  - Action: We can monitor traffic loads and buffer utilization on the affected interfaces. Look for periods of high utilization that correlate with packet loss.
  - Goal: To determine if the router is experiencing congestion and whether traffic bursts are overwhelming its buffering capacity.
- We may validate MTU Settings
  - Action: We can confirm that the MTU settings are consistent across the routers and that Path MTU Discovery (PMTUD) is functioning correctly.
  - Goal: To prevent packet drops due to fragmentation issues or MTU mismatches.
- We may review Routing Configurations and QoS Policies
  - Action: We can ensure that routing tables are accurate and that QoS settings are properly prioritizing traffic. We also check for routing loops or misrouted traffic.
  - Goal: To ensure that traffic is taking optimal paths and that policies aren't inadvertently causing drops.
- We may inspect Physical Connections
  - Action: We can physically inspect cables and connectors between routers, and potentially replace any that show signs of wear or damage.
  - Goal: To eliminate physical faults as a source of the problem.

For Transport Layer Considerations

- We may analyze TCP Statistics
  - Action: We can use packet capture tools (like Wireshark) to inspect TCP flows. Look at retransmission rates, congestion window adjustments, and duplicate ACKs.
  - Goal: To confirm that TCP is reducing its rate in response to packet loss and that it isn't compounding the issue.
- We may optimize TCP Parameters
  - Action: If necessary, we can adjust TCP configurations—such as window scaling, retransmission timeout settings, or selecting a more aggressive congestion control algorithm—to better suit the network conditions.
  - Goal: To improve TCP performance under conditions of packet loss, in order to allow the transport layer to recover better.

Part II: Read the following papers (included in the `.zip` file of this project)

- FlowPic_Encrypted_Internet_Traffic_Classification_is_as_Easy_as_Image_Recognition.pdf
- Early_Traffic_Classification_With_Encrypted_ClientHello_A_Multi-Country_Study.pdf
- Analyzing_HTTPS_encrypted_traffic_to_ide.pdf

For each paper, write:

- What is the main contribution of the paper?
- What traffic features does the paper use, and which are novel?
- What are the main results (you may copy the figures from the paper), and what are the insights from their results?

You may use AI tools, but you must check reliability of the answer. In particular, write the prompts that you used.

First Research Article - FlowPic

Main Contribution of the Paper

The paper presents FlowPic, a novel network classification approach that converts flow-level network data—specifically, packet sizes and arrival times—into image representations. These images are then analyzed by a Convolutional Neural Network (CNN) for traffic classification. Unlike methods that rely on packet payloads, FlowPic preserves user privacy by focusing solely on metadata

(such as packet size and timing). Even though FlowPic uses a balanced dataset and a single CNN architecture across multiple classification tasks, it achieves high accuracy in identifying different types of network traffic.

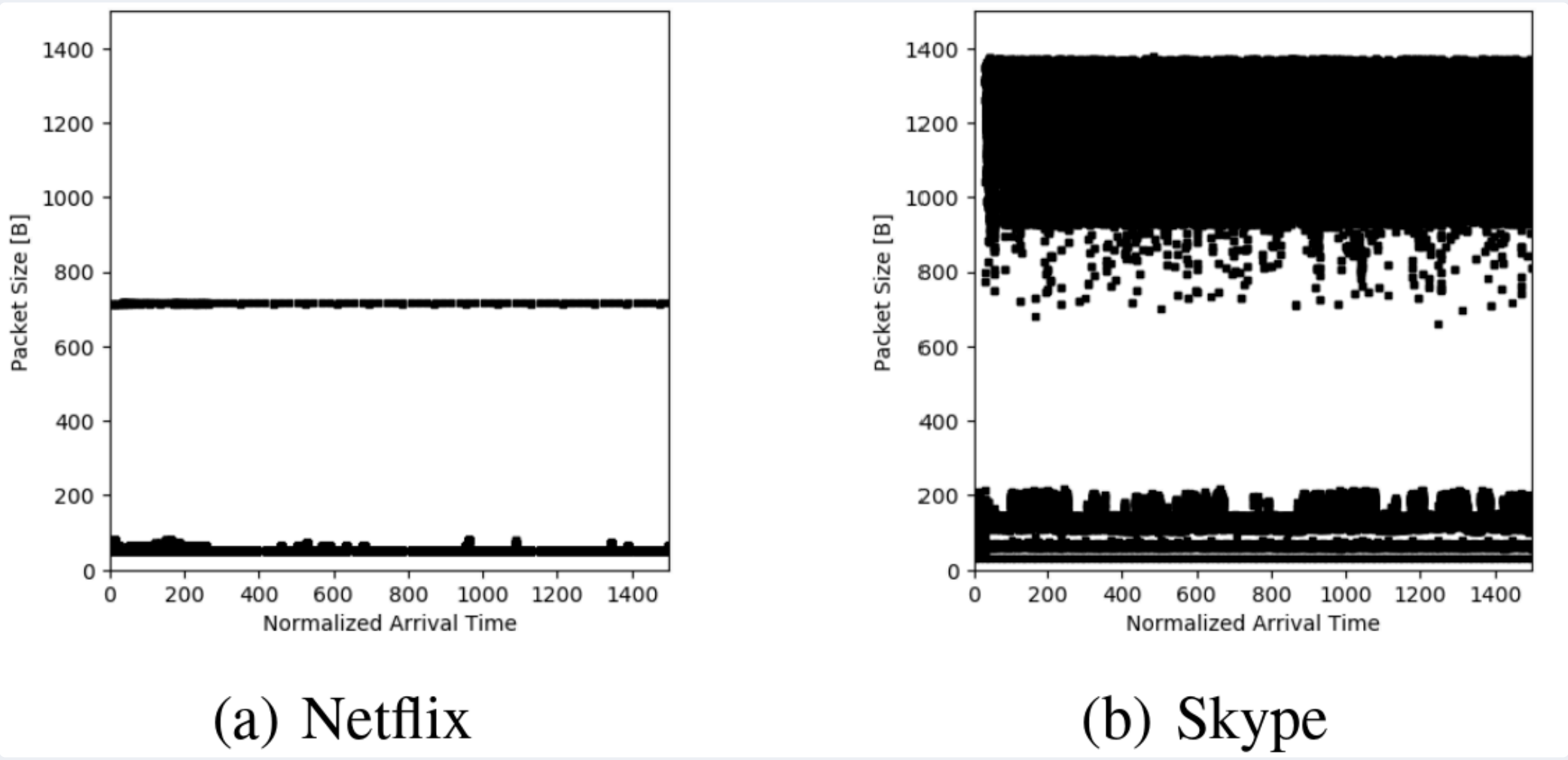
Traffic & Novel Features

Traffic Features Used by the Method

a. Packet Size Distribution
   FlowPic creates a 2D histogram (image) based on packet sizes (up to 1500 bytes), capturing how frequently packets of a certain size occur. By focusing on packet sizes under this limit, the method eliminates oversized packets that could skew the classification process.
b. Arrival Time Intervals
   Packet arrival times are grouped into specific time intervals. This temporal information enriches the representation, enabling the CNN to detect timing-based patterns in network flows.
c. Unidirectional Flows
   The analysis considers each flow independently in a unidirectional manner rather than using bidirectional session-based tracking. This design choice allows for early-stage classification—decisions can be made as soon as enough data is gathered from a single flow, without needing to wait for the complete bidirectional session (client $\Longleftrightarrow$ server).

Novel Features Used by the Method

a. 2D Histogram Representation
   FlowPic's distinctive feature is how it converts network flow data into a 2D image. Each pixel in this image corresponds to a combination of packet size and arrival time interval, with the pixel intensity reflecting how often a packet of a certain size arrives within a particular time window.



Computer Science - Ariel University/Year II/Semester I/Communication Networks/Final Project/Gallery & Excalidraw/image.webp

b. CNN-Based Direct Classification
   By leveraging a CNN, FlowPic bypasses the need for manual or statistical feature engineering. The network learns relevant features directly from the histogram images, simplifying the workflow and potentially unveiling patterns that might go unnoticed with manual approaches.
c. Encryption Resistance
   Since FlowPic relies on metadata (packet size and timing) rather than payload content, it remains effective even when traffic is encrypted (e.g., VPN or Tor). The anonymization and encryption techniques applied to the payload do not disrupt FlowPic's fundamental methodology, and respect the user's privacy and data confidentiality.
d. Generalization Ability
   Because FlowPic's CNN is trained on the overall behavior of categories (rather than specific applications), it can generalize more effectively and classify new applications that were absent during training.

Main Results and Insights

In most classification scenarios, FlowPic either outperforms or matches other modern methods. Notably, many existing techniques are customized for specific use cases or rely on unbalanced datasets, whereas FlowPic is built for a broad range of tasks using a

balanced dataset.

Problem	FlowPic Accuracy (%)	Best Previous Result	Remark
Non-VPN Traffic Categorization	85.0%	84.0% Pr., Gil et al.	Different categories. Used unbalanced dataset.
VPN Traffic Categorization	98.4%	98.6% Acc., Wang et al.	Classify raw packets data. Not including browsing category.
Tor Traffic Categorization	67.8%	84.3% Pr., Gil et al.	Different categories. Used unbalanced dataset.
Non-VPN Class vs. All	97.0% (average)	No previous results.	///
VPN Class vs. All	99.7% (average)	No previous results.	///
Tor Class vs. All	85.7% (average)	No previous results.	///
Encryption Techniques	88.4%	99.0% Acc., Wang et al.	Classify raw packets data, not including Tor category.
Applications Identification	99.7%	93.9% Acc., Ya-mansavascilar et al	Different classes.

- Tor Traffic Categorization
  FlowPic's accuracy drops to around 67.8% for Tor traffic, trailing behind another method that achieves 84.3%. This shortfall is attributed to Tor's heavy anonymization, which alters packet sizes and timings to such an extent that FlowPic struggles to extract meaningful patterns.
- Encryption Techniques Identification
  FlowPic achieves 88.4% accuracy, whereas the leading approach reaches 99.0%. However, the higher-scoring method leverages packet payload manipulation, which is at odds with FlowPic's goal of maintaining privacy. Despite the lower accuracy, FlowPic's results are still noteworthy given its privacy-friendly design and reliance on non-payload features.

Overall, FlowPic offers a privacy-focused, generalizable, and highly effective means of classifying network traffic. While certain specialized tasks, such as deeply anonymized Tor traffic classification, remain challenging, FlowPic demonstrates impressive performance across diverse categories without compromising user privacy.

Second Research Article - Encrypted ClientHello

Main Contribution of the Paper

The main contributions of the paper are:

- The development of hybrid Random Forest Traffic Classifier (hRFTC), a novel encrypted traffic classification algorithm that integrates unencrypted TLS handshake payload, flow-based time series, and packet size statistics as classification features.
- Demonstrating that hRFTC significantly outperforms state-of-the-art classifiers, including packet-based, flow-based, and hybrid algorithms.
- Evaluating the robustness of existing early Traffic Classification (eTC) algorithms in handling heterogeneous/mixed data in Encrypted ClientHello (ECH) scenarios and their ability to generalize from small datasets.
- Showing that even the best hybrid Traffic Classification (TC) algorithms trained in one geographic region/country need retraining in another due to differences in traffic patterns.

Traffic & Novel Features

Traffic Features Used by the Method

The paper uses both packet-based and flow-based features for classification:

- Packet-based features: Derived from the unencrypted parts of the TLS handshake, including ClientHello (CH) and ServerHello (SH) messages.
- Flow-based features: Extracted from sequences of Packet Sizes (PSs) and Inter-Packet Times (IPTs).
- TLS Version: The TLS version is an unencrypted part of a TLS metadata, and with it we can better understand the type of traffic being passed over the network.
- Cipher Suite List: The ordered list of cipher suites offered by the client.
- Presence of Extensions: The variants of extensions present in the ClientHello.
- Extension Count: The number of extensions.
- Basic Packet Lengths / Sizes: The size of the ClientHello message or the first few packets.

Novel Features Used by the Method

- Packet Selection Criterion: Unlike prior algorithms that analyze a fixed number of packets, hRFTC processes packets only until the first downlink packet with application data, reducing classification delay.

- Integration of QUIC-Specific Features: Since HTTP/3 heavily relies on QUIC, the algorithm is extended to consider QUIC Connection IDs and encrypted QUIC handshake messages.
- Hybrid Feature Set: The combination of flow-based statistics and packet-based TLS metadata significantly improves classification accuracy while maintaining low delay.

The novel features in this paper do not rely solely on concrete or "tangible" classification samples such as the presence or absence of certain TLS parameters. The main novel features used in the paper rely on the shape of the traffic flow, the structure, ordering and peculiarities of how TLS parameters are advertised or arranged.

These features are not concrete and require deeper inspection of the raw TLS handshake bytes, extension sequences, or unique handshake patterns that standard classifiers do not commonly leverage.

---

3. What are the main results (including figures) and insights?

The main results of the study focus on the effectiveness of hRFTC compared to existing algorithms:

a. Classification Performance:
- hRFTC outperforms leading packet-based TC algorithms:
  - 56.3% improvement over RB-RF
  - 63.3% improvement over MATEC
  - 59.6% improvement over BGRUA.
- hRFTC reduces inaccuracy in hybrid classifiers by approximately 50%.

b. Feature Importance:
- Packet size features contribute the most to classification accuracy, making up over 50% of feature importance.
- Despite ECH encryption, payload features still contribute about 30% to classification.

c. Generalization Ability:
- The hRFTC classifier retains high performance even when trained on only 10% of the dataset, reducing the macro F-score by only 7%.
- However, classification performance drops significantly when applied to unseen geographic locations, highlighting the need for location-specific retraining.

d. Impact of QUIC PADDING:
- QUIC PADDING complicates classification because all QUIC handshake packets have the same lengths, reducing the classifier's ability to differentiate flows.

Figures

Figure 1: TLS 1.3 Handshake with Encrypted ClientHello

This figure illustrates the structure of a TLS 1.3 handshake, focusing on the transition from plaintext ClientHello (CH) to Encrypted ClientHello (ECH).

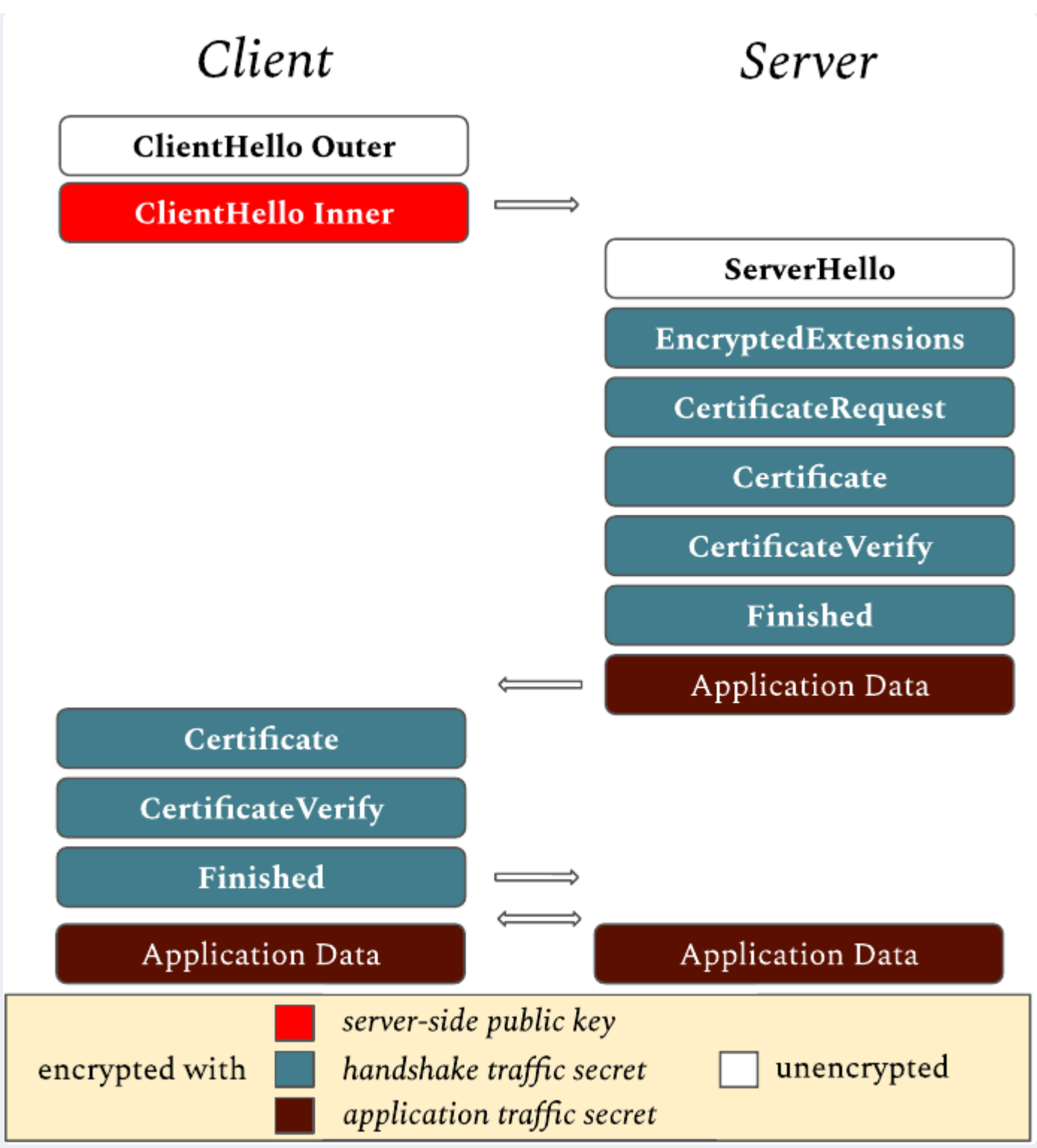


image-5.webp

Key Takeaways:

- The ECH mechanism enhances security by encrypting the CH, making it harder for traffic classifiers to use SNI-based classification.
- Some TLS parameters (e.g., supported versions and cipher suites) remain unencrypted for backward compatibility, which traffic classifiers can still leverage.

Figure 2: Hybrid Random Forest Traffic Classifier (hRFTC)

This figure depicts the architecture of hRFTC, the hybrid classifier that integrates packet-based and flow-based features.

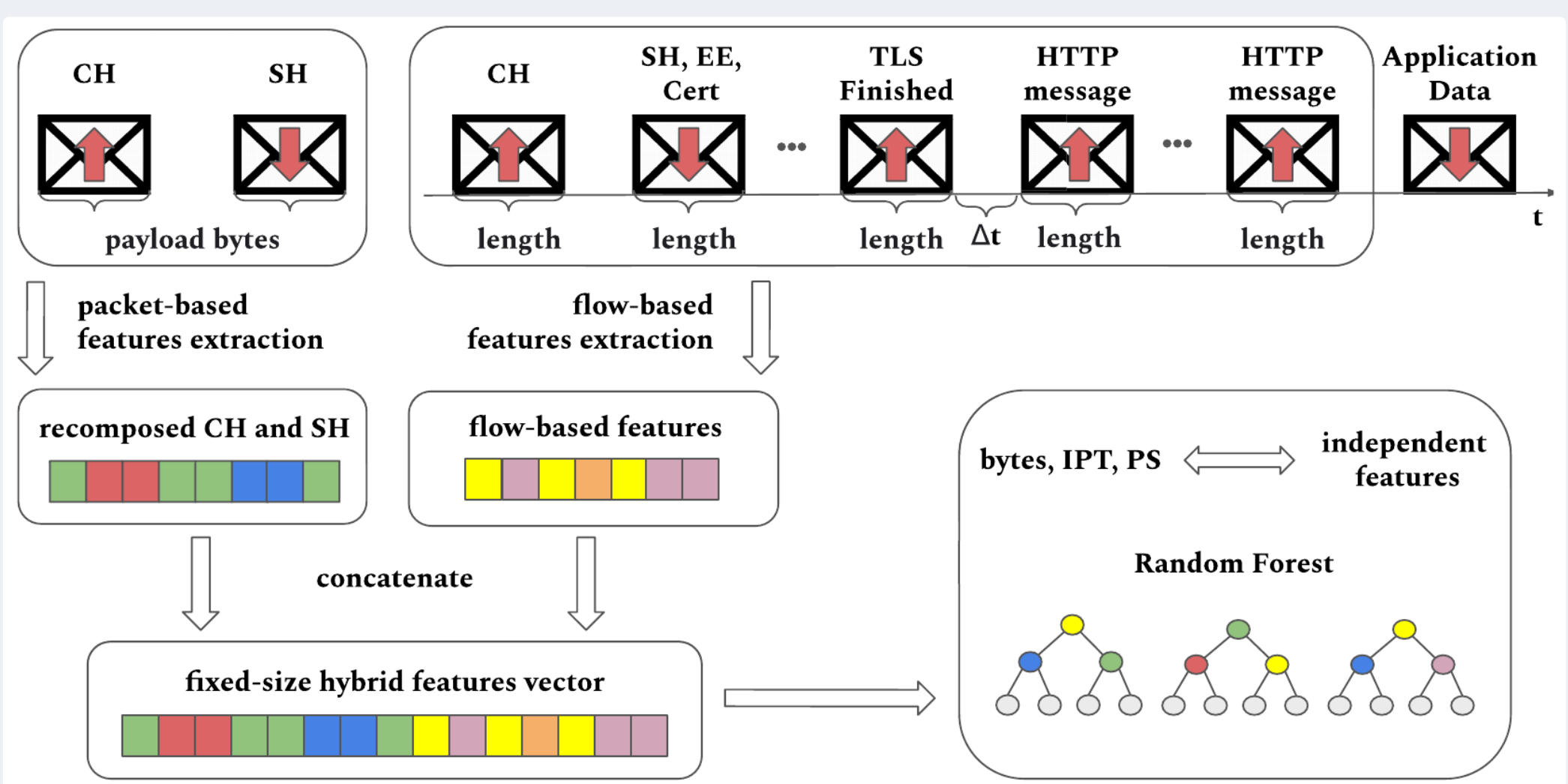


image.png

Key Takeaways:

- Packet-based features are extracted from unencrypted TLS handshake fields (e.g., CH parameters).
- Flow-based features leverage Packet Sizes (PSs) and Inter-Packet Times (IPTs) to improve classification accuracy.
- The Random Forest model processes both feature types for improved classification, leading to higher robustness than previous classifiers.

Figure 3: Recomposed Payload Modification for QUIC Flows

This figure shows modifications in QUIC traffic classification to handle encrypted TLS over QUIC.

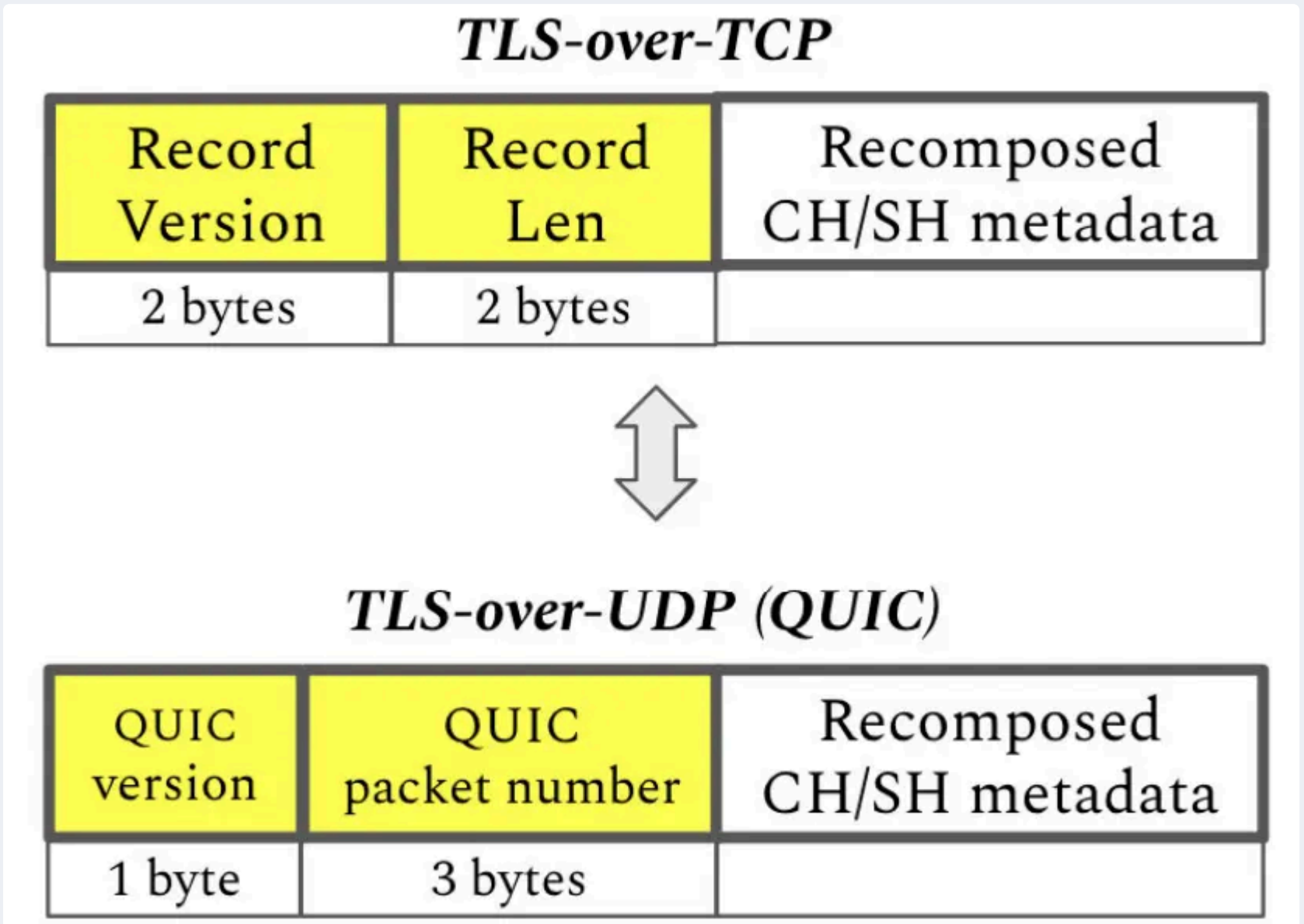


image-6.webp

Key Takeaways:

- QUIC introduces Connection IDs instead of TCP's standard 5-tuple flow tracking.
- QUIC PADDING frames make it difficult to distinguish flows based on packet sizes alone.
- The classifier adapts by using QUIC-specific fields (e.g., QUIC Version, Packet Number) as alternative identifiers.

4. Figure 4: F-score Depending on the Training Subset Size

This figure shows the generalization ability of hRFTC across different dataset sizes.

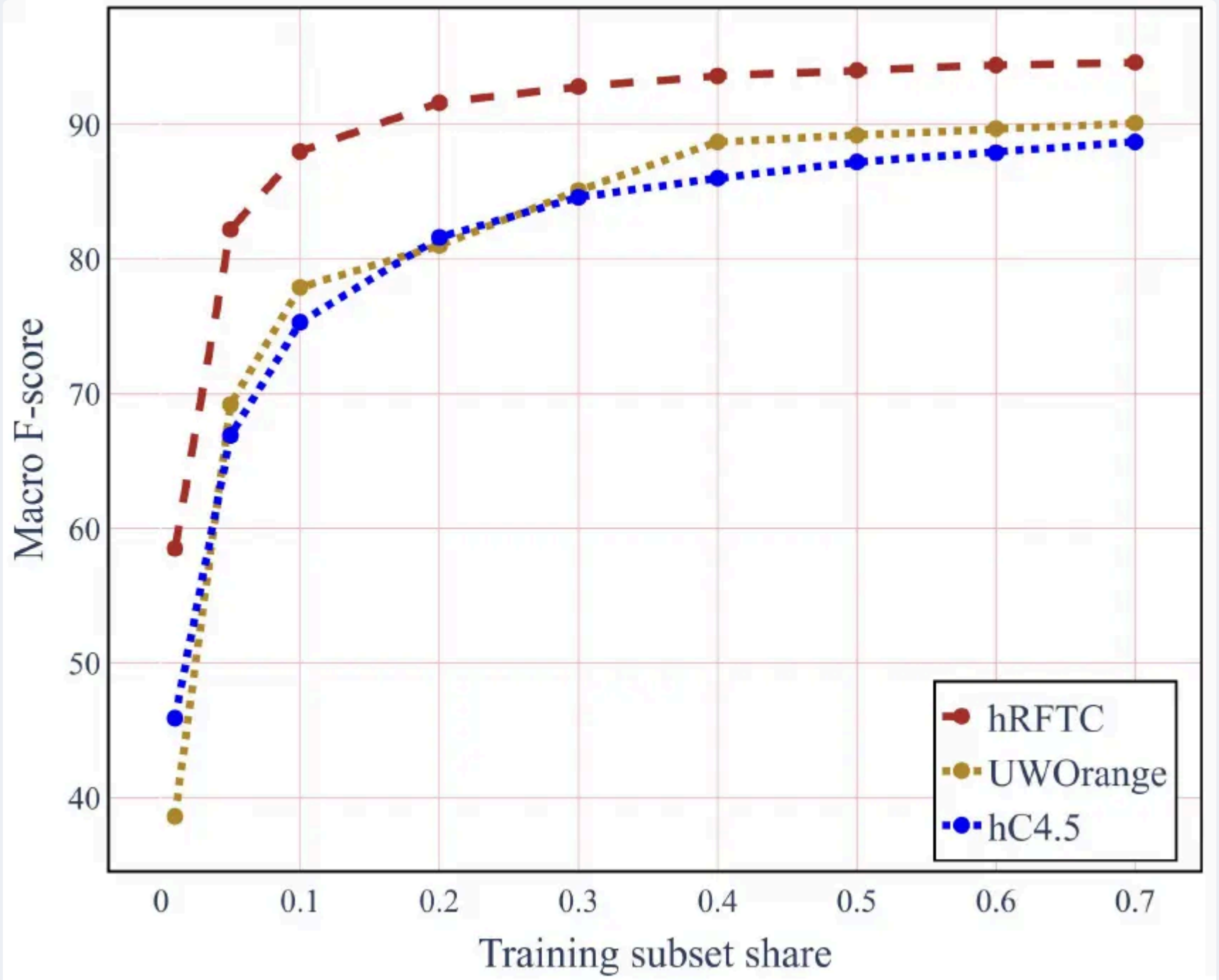


image-7.webp

Key Takeaways:

- hRFTC maintains high accuracy even with just 10% of training data, dropping only 7% in F-score compared to full training.
- This highlights the model's strong learning efficiency, outperforming other hybrid classifiers (e.g., hC4.5 and UW).
- However, geographic variations in traffic still require retraining, as classifiers trained in one location perform poorly in another.

5. Feature Importance Analysis (Table 12 & Table 13)

These tables quantify the impact of different traffic features in classification using Impurity-based feature importance.

Rank	Feature	Impurity-based Feature Importance
1	CH Cipher Suites length	1.00
2	DL PSs Cumulative Sum	0.62
3	DL PSs Sorted Unique #1	0.50
4	UL PSs Std	0.46
5	UL PSs Cumulative Sum	0.45
6	UL PSs #2	0.44
7	DL IPTs Sum	0.43
8	DL PSs 25th percentile	0.43
9	DL PSs average	0.42
10	UL PSs 75th percentile	0.41
11	SH Cipher Suite	0.40
12	DL PSs Std	0.40
13	UL PSs Sorted Unique #2	0.40
14	UL PSs max	0.38
15	DL PSs 50th percentile	0.37
16	UL PSs average	0.36
17	DL PSs 75th percentile	0.33
18	UL IPTs Sum	0.28
19	UL PS #1 (CH length)	0.27
20	DL PS #2	0.26
21	UL IPTs min	0.26
22	UL PS 750-1000B freq	0.25
23	UL PSs Sorted Unique #3	0.25
24	CH Extensions Length	0.24
25	SH Extension Type #2	0.23
26	UL IPTs 25th percentile	0.23
27	UL IPTs 50th percentile	0.22
28	SH Extension Type #1	0.22
29	DL PS #3	0.21
30	UL IPTs max	0.2

image-8.webp

Payload		IPT	PS			
CH	SH	IPT Stats	PS Hist	PSs Unique	PS Stats	PS Pattern
0.18	0.09	0.16	0.05	0.11	0.16	0.26
0.27		0.16	0.57			

image-9.webp

Key Takeaways:

- Packet size statistics contribute to over 50% of classification accuracy, reinforcing the importance of flow-based features.
- Despite encryption (ECH), payload features still retain around 30% importance, meaning some TLS metadata remains useful for classification.

6. Table 11: F-score Performance Comparison of Classifiers

This table compares hRFTC with leading packet-based, flow-based, and hybrid classifiers.

Class	F-score [%]						
	Hybrid Classifiers			Flow-based Classifier	Packet-based Classifiers		
	hRFTC [proposed]	**UW** [35]	**hC4.5** [34]	**CESNET** [63]	**RB-RF** [24]	**MATEC** [33]	**BGRUA** [32]
BA-AppleMusic	**92.1**	89.5	80.2	89.2	25.5	13.1	14.5
BA-SoundCloud	**99.6**	98.9	97.8	98.7	84.4	81.8	82.0
BA-Spotify	**93.6**	90.8	89.0	88.5	16.3	0.0	3.6
BA-VkMusic	**95.7**	89.7	88.5	91.8	2.6	2.1	3.2
BA-YandexMusic	**98.5**	93.2	93.7	92.5	1.8	0.2	0.1
LV-Facebook	**100.0**	99.7	99.8	99.8	**100.0**	**100.0**	**100.0**
LV-YouTube	**100.0**	**100.0**	99.9	**100.0**	**100.0**	99.0	98.4
SBV-Instagram	**89.7**	74.7	76.5	78.8	10.0	6.3	6.4
SBV-TikTok	**93.3**	81.8	81.8	76.3	38.3	34.3	34.5
SBV-VkClips	**95.7**	94.0	91.3	92.4	53.2	37.7	46.0
SBV-YouTube	**98.2**	96.6	94.7	96.4	1.1	0.2	0.2
BV-Facebook	**87.7**	78.2	79.7	77.6	5.6	3.2	3.8
BV-Kinopoisk	**94.1**	84.1	85.8	89.8	5.4	4.0	4.1
BV-Netflix	**98.5**	97.2	95.2	93.7	50.7	52.3	56.1
BV-PrimeVideo	**91.3**	86.7	84.1	84.7	32.5	24.7	26.8
BV-Vimeo	**94.8**	90.5	90.2	81.4	72.0	19.5	68.6
BV-VkVideo	**88.6**	80.5	80.4	79.7	10.5	0.0	0.1
BV-YouTube	**85.9**	84.3	77.0	78.5	22.3	19.6	20.2
Web (known)	**99.7**	99.5	99.4	99.4	98.0	98.0	98.0
Macro-F-score (average)	**94.6**	89.9	88.7	88.9	38.4	31.4	35.1

LV is Live Video, (S)BV is (Short) Buffered Video, and BA is Buffered Audio.

image-10.webp

Key Takeaways:

- hRFTC outperforms all other classifiers, achieving 56.3%–63.3% higher accuracy than prior packet-based algorithms.
- It also reduces classification errors by 50% compared to previous hybrid models.
- However, QUIC's PADDING frames lower classification accuracy for QUIC-based flows, as all QUIC handshake packets have the same lengths.

Key Insights

a. hRFTC's Hybrid Approach Significantly Outperforms State-of-the-Art Models
   - Integrating TLS metadata with flow-based statistics boosts classification accuracy.
   - Unlike purely packet-based classifiers, hRFTC remains effective despite ECH encryption.

b. Flow-Based Features Are More Reliable Than Packet-Based Features
   - Packet size statistics (PSs) and inter-packet times (IPTs) provide more than 50% of classification accuracy.
   - Even though TLS metadata is increasingly encrypted, flow-based features still enable accurate classification.

c. Geographic Differences Require Retraining
   - Despite its robustness, hRFTC needs location-specific training due to variations in network infrastructure and CDN distribution.

d. QUIC PADDING Frames Reduce Classification Accuracy
   - Since QUIC's handshake packets have identical sizes, traditional size-based methods fail.
   - The modified recomposed payload method in hRFTC partially mitigates this issue.

Third Research Article - HTTPS Encrypted Traffic Analysis

Main Contribution of the Paper

The primary contribution of this paper is to demonstrate that a passive attacker can accurately identify a target's operating system, browser, and application only from encrypted HTTPS traffic. This is the first attempt to achieve this level of classification with high accuracy. The authors achieve this by:

- Introducing novel features based on browser bursty behavior and SSL characteristics.
- Constructing a large dataset with more than 20,000 sessions, which includes different operating systems (Windows, Linux-Ubuntu, OSX), browsers (Chrome, Internet Explorer, Firefox, Safari), and applications (YouTube, Facebook, Twitter).
- Using a Support Vector Machine (SVM) with Radial Basis Function (RBF) kernel.

This model helped achieving a classification accuracy of 96.06%. These findings indicate that encryption alone does not fully protect against traffic analysis attacks, and attackers are able to infer user information with high confidence.

Traffic Features Used and Novel Features:

Traffic Features Used by the Method

- Number of forward and backward packets:
  - Counts the packets sent and received, which can indicate traffic volume and communication patterns.
- Total bytes transmitted in each direction:
  - Measures the overall data transferred, which helps in distinguishing between different applications and usage behaviors.
- Minimum, maximum, mean, and standard deviation (STD) of inter-arrival times:
  - Helps characterize the timing patterns of traffic flows, differentiating between real-time applications and bulk data transfers.
- Packet size statistics (min, max, mean, variance):
  - Different applications and protocols have different ways to distribute packet sizes, which helps in classification.
- Total packets in a session:
  - The number of packets exchanged during a connection is useful for identifying short vs. long-lived sessions.
- Mean Time-To-Live (TTL) value:
  - Can indicate the operating system, as different systems use distinct TTL values.

Novel Traffic Features Used by the Method

- TCP Initial Window Size and Scaling Factor: Identifies OS differences based on TCP congestion control settings.
- SSL/TLS-Specific Features:
  - Number of SSL compression methods:
    - Determines the number of compression methods supported by the client, which varies across browsers.
  - SSL extension count:
    - Helps differentiate browsers based on their use of specific SSL extensions.
  - SSL cipher method count:
    - Identifies different encryption algorithms supported by different clients.
  - SSL session ID length:
    - The session ID length varies based on browser and server configurations.
  - Forward SSL version:
    - Indicates the SSL/TLS version being used, which is often unique to certain browser-OS combinations.
- Browser Bursty Behavior Features:
  - Number of bursts in forward and backward traffic:
    - Browsers often exhibit bursts of traffic due to loading webpage elements in parallel.
  - Peak throughput measurements (min, max, mean, STD) in each direction:
    - Helps characterize browser behavior when retrieving web pages.
  - Inter-arrival time statistics for bursts:
    - Measures timing between data bursts to detect browser and application-specific patterns.
  - Number of keep-alive packets:
    - Some applications and browsers send keep-alive packets at different rates.
  - Maximum segment size (MSS) in TCP:
    - MSS values can differ between operating systems and network stacks, aiding in classification.

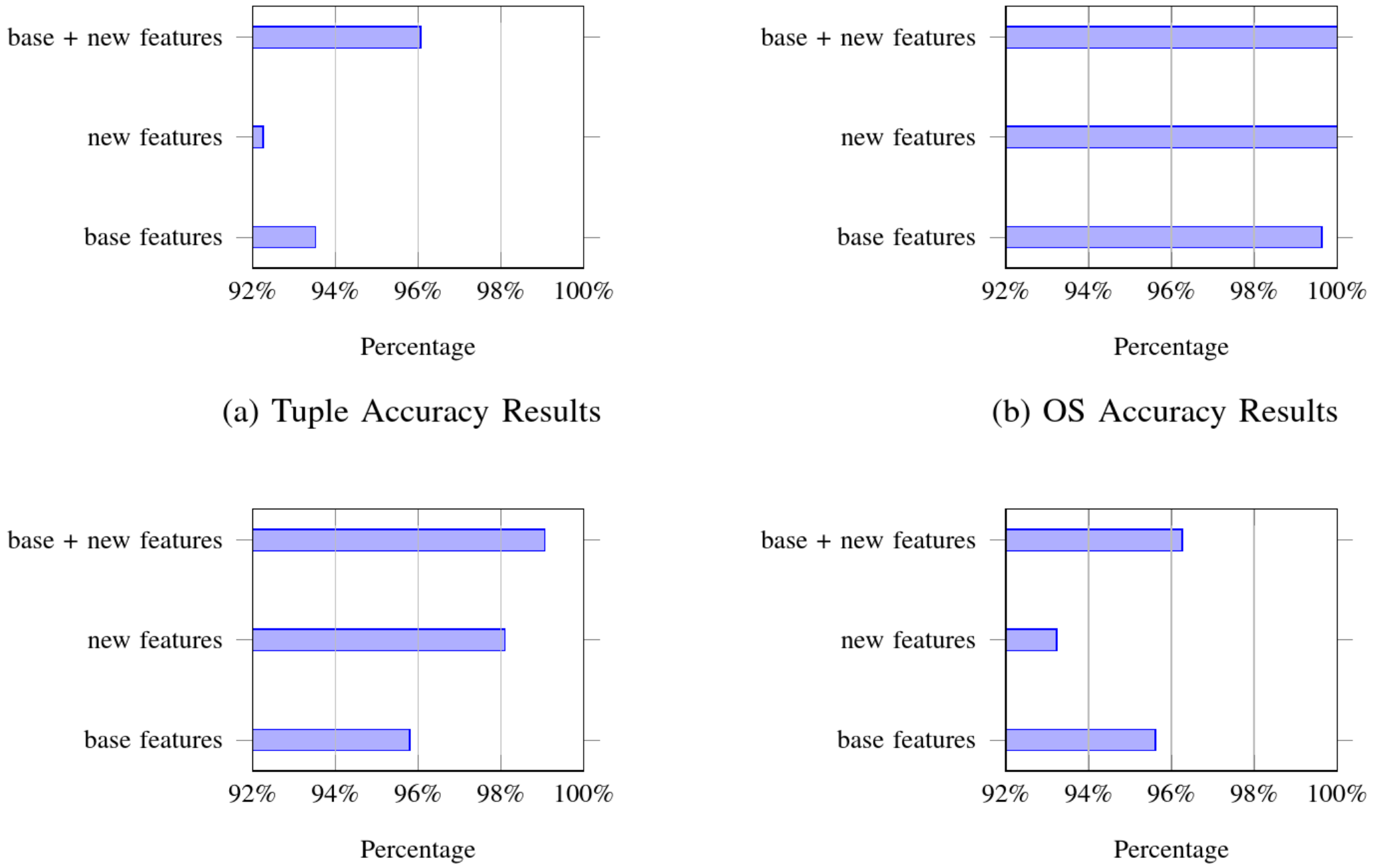
Main Results and Insights

The authors perform a series of experiments using an SVM classifier with RBF kernel and report the following key findings:

Classification Accuracy Results

- Tuple Classification (OS, Browser, Application): 96.06%
- OS Classification: 100%
- Browser Classification: ~99%
- Application Classification: ~98%

Figure 2: Accuracy results for different feature sets (Base vs. New vs. Combined)



(a) Tuple Accuracy Results

(b) OS Accuracy Results

image-1.webp

- Using only base features: 93.52% accuracy.
- Using only new features: 95.34% accuracy.
- Using both sets of features: 96.06% accuracy.

Figure 3: Browser Bursty Behavior Example

- The authors show that browsers exhibit a characteristic burst pattern, which helps classification.

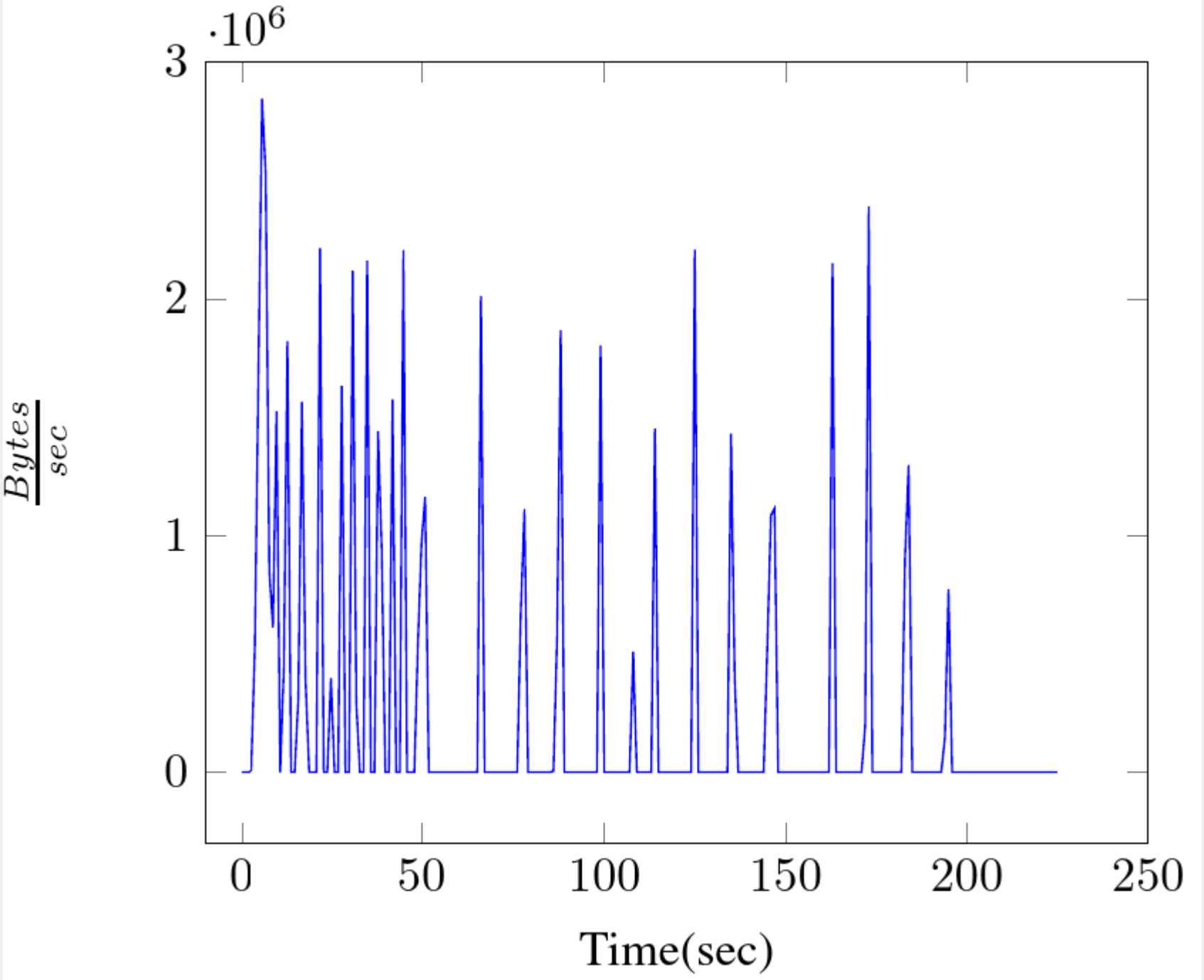


image-2.webp

Figure 4: Confusion Matrices

- The confusion matrices demonstrate that misclassification occurs primarily in cases labeled as "Unknown," suggesting that the classifier is highly confident in its other classifications.

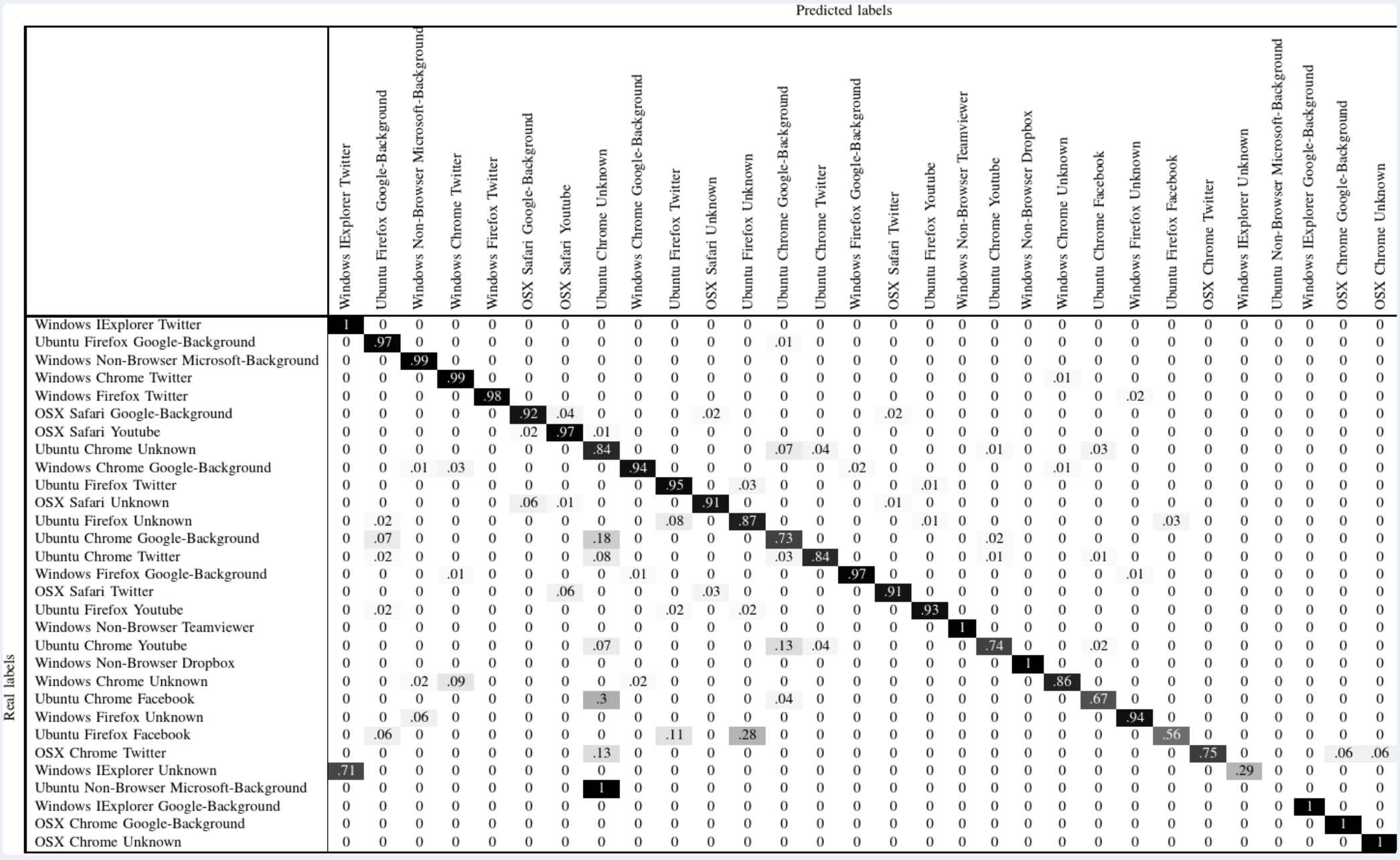


image-3.webp

Predicted labels

Real labels	Windows	Ubuntu	OSX
Windows	1	0	0
Ubuntu	0	1	0
OSX	0	0	1

(b) OS Confusion Matrix

Predicted labels

Real labels	Chrome	Firefox	IExolorer	Safari	Non-Browser
Chrome	.97	.02	0	0	0
Firefox	.01	.98	0	0	0
IExolorer	0	0	1	0	0
Safari	.01	0	0	.99	0
Non-Browser	.03	0	0	0	.96

(c) Browser Confusion Matrix

Predicted labels

Real labels	Dropbox	Facebook	Google-background	Microsoft-Background	Teamviewer	Twitter	Youtube	Unknown
Dropbox	.98	0	.02	0	0	0	0	0
Facebook	0	.62	.04	0	0	.04	0	.29
Google-background	0	0	.95	0	0	.01	.01	.03
Microsoft-Background	0	0	0	.96	0	0	0	.04
Teamviewer	0	0	0	0	1	0	0	0
Twitter	0	0	0	0	0	.98	0	.01
Youtube	0	0	.03	0	0	.02	.93	.01
Unknown	0	.02	.04	.01	0	.05	.01	.86

(d) Application Confusion Matrix

image-4.webp

Key Insights

a. High Accuracy Despite Encryption:
- The study shows that despite SSL/TLS encryption, HTTPS traffic retains distinguishable patterns that allow OS, browser, and application classification.

b. Novel Features Improve Classification:
- Adding SSL- and burst-related features significantly improves performance compared to traditional traffic analysis methods.

c. Privacy and Security Implications:
- Since an attacker can infer detailed user information passively, this work highlights the need for enhanced privacy mechanisms such as traffic obfuscation techniques.

Part III

Traffic Characteristics of Interest:
A. IP header fields.
B. TCP header fields.
C. TLS header fields.
D. packet sizes.

a. Use Wireshark (or another software) to record your PC's traffic. Please open only one app at a time and minimize "noise" traffic (e.g., SW updates, notifications, etc.).

Consider the following apps: (Due to TLS, you may need to save the keys to decrypt the traffic)

- Web-surfing 1: browse web pages using a browser (e.g., Google Chrome).
- Web-surfing 2: browse web pages using another browser (e.g., MS Edge).
- Audio streaming: e.g., Spotify, where you can choose voice-only (e.g., podcast, or song without video clip).
- Video streaming: e.g., YouTube.
- Video conferencing: e.g., Zoom.

a. Present plots comparing those apps' traffic characteristics (e.g., the characteristics A, B, C and D mentioned above). You may use Wireshark plots. Your code must correctly run on Linux, and will be checked on Linux prompt, using Python's clean installation. If you use Python packages. you should state that in the `.pdf` file.
b. Explain the plots literally. E.g., say: "app X typically uses much larger packets than app Y".
c. Record your PC's traffic again. Suppose you're an attacker who tries to learn which apps the user accessed. Consider 2 options:
   - The attacker knows each packet's size, time stamp, and hash of the 4-tuple (source IP, dest IP, source port, dest port) flow ID.
   - The attacker knows only the size and time stamp for each packet. Check to what extent the attacker can identify which site/apps the user visited, even if the traffic is fully encrypted (or at least anonymized). Explain why the attacker can or cannot identify the site and how we can mitigate this attack.

Bonuses (10 pt):
Repeat the test when one main app is open, but another is used occasionally. E.g., the user listens to music using Spotify and occasionally sends emails. If you have an idea for a slightly different aspect of this problem, you are welcome to email one of the lecturers and get permission to implement it.

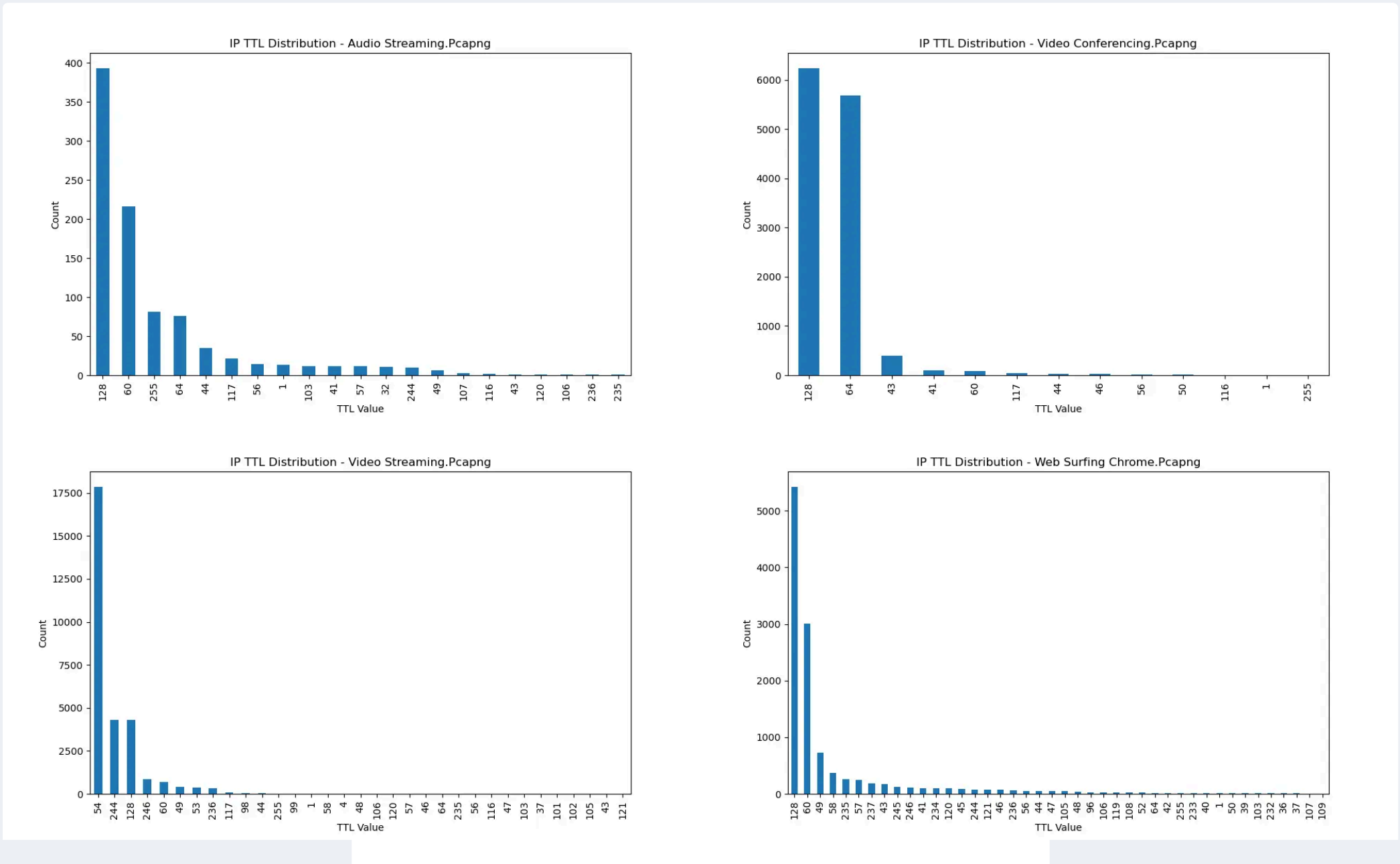
Solution

Six `.pcapng` recordings have been captured. Here's a description for each:

a. `audio_streaming.pcapng`: This recording was captured by turning on Spotify and playing a podcast in the background for approximately 35 seconds.
b. `video_conferencing.pcapng:` This recording was captured by joining a Zoom video-audio conference with two participants, the conference lasted for around 2 minutes.
c. `video_streaming.pcapng`: This recording was captured by joining a Twitch live stream and having it play in the background for around 40 seconds.
d. `web_surfing_edge.pcapng`: This recording was captured by opening the Edge web browser, navigating to Google, navigating to the website WikiHow, accessing a few `.gif` images, navigating back to the Google main search page and then navigating to Wikipedia, and then accessing a `.png` file.
e. `web_surfing_chrome.pcapng`: This recording was captured identically as the Edge browser case, except the recording was performed on the Google Chrome browser.
f. `mixed_traffic.pcapng` (Bonus 10pts): Mixed traffic capture. Played Spotify music in the background, sent an e-mail to recipient and navigated using Edge browser to a Wikipedia page.

Plot Analysis

IP Headers



ip_ttl.webp

a. Audio Streaming (Spotify Podcast)

- The most frequent TTL values are around 128 and 64.
- TTL values 128 and 64 are typically used by Windows and Linux-based servers, which are common for streaming services like Spotify, YouTube Music, or Apple Music.
- Indicates that most packets originate from a relatively close source.

b. Video Conferencing (Zoom)

- The dominant TTL values are 128 and 64.
- There is a significant drop-off in packet counts as TTL values increase.
- There are significant bursts in IP header counts in the TTL values of 128 and 64, and this is common in video conferencing scenarios, because we want to pass the highest amount of data in as little time as possible.
- This suggests that Zoom traffic is primarily coming from a small number of server sources.

c. Video Streaming (Twitch)

- A very high number of packets with TTL values near 128.
- Other TTL values are present but significantly less frequent.
- Suggests that Twitch servers are located relatively close to the source.

d. Web Surfing (Google Chrome)

- The TTL values are more varied but still concentrated around 128 and 64.
- Some packets have significantly lower TTL values, indicating varied sources.
- Likely reflects requests to multiple content delivery networks (CDNs) and different websites.

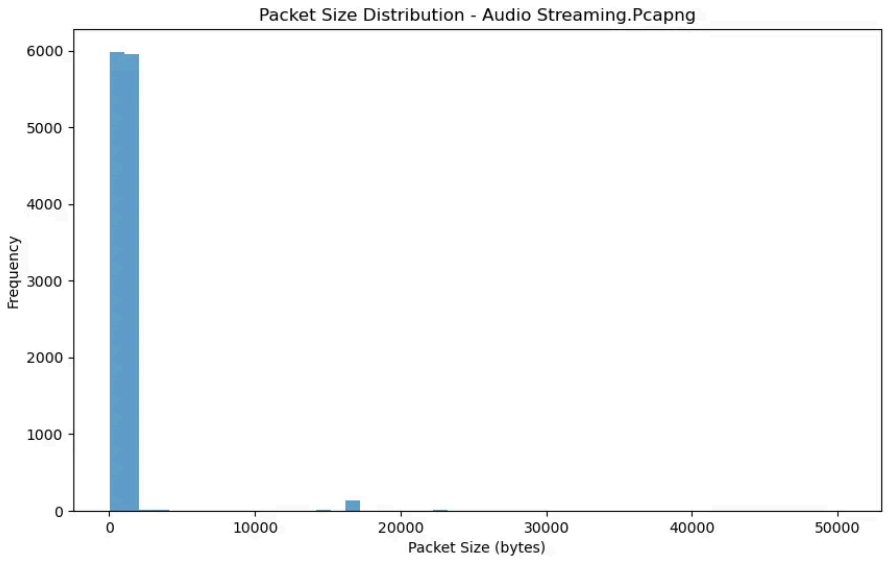
e. Web Surfing (Microsoft Edge)

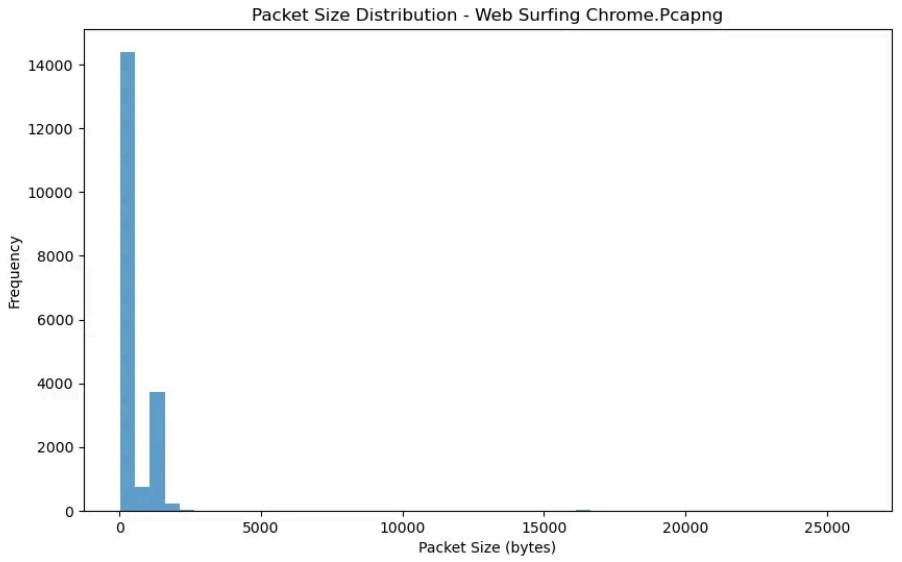
- Similar pattern to Chrome, with dominant TTL values around 128 and 64.
- Some lower TTL values are present, indicating diverse server origins.
- Edge and Chrome access a similar variety of sources.

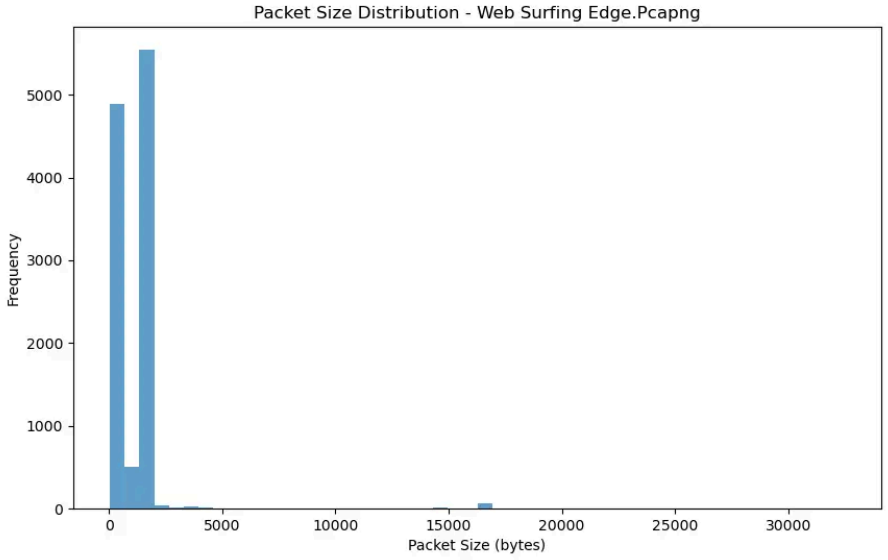
Key Observations:

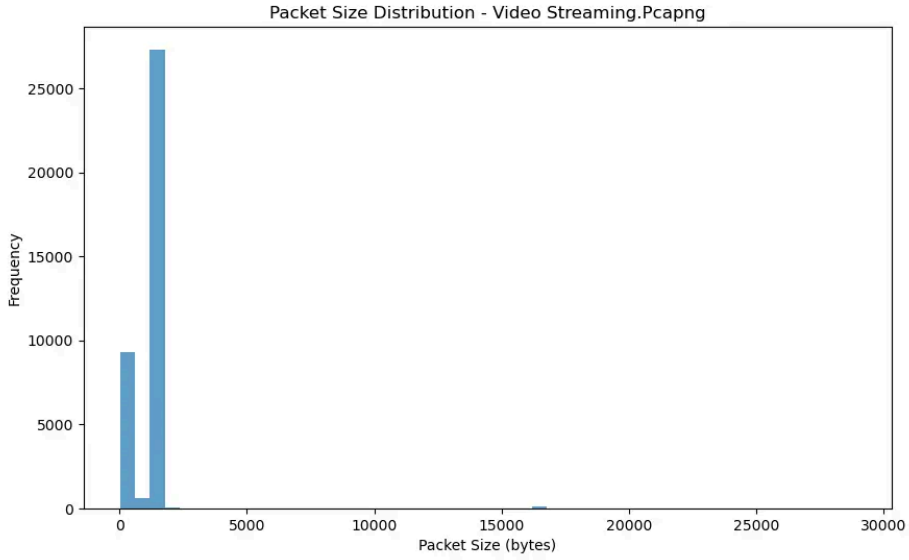
- 128 and 64 TTL values dominate across all activities, suggesting that most traffic originates from a small set of well-defined servers.
- Video streaming generates the most packets, as seen in the Twitch dataset.
- Web browsing shows a broader TTL distribution, likely due to requests to different domains and CDNs.
- Audio and video conferencing involve a smaller but consistent set of TTL values, which suggests communication with dedicated servers.

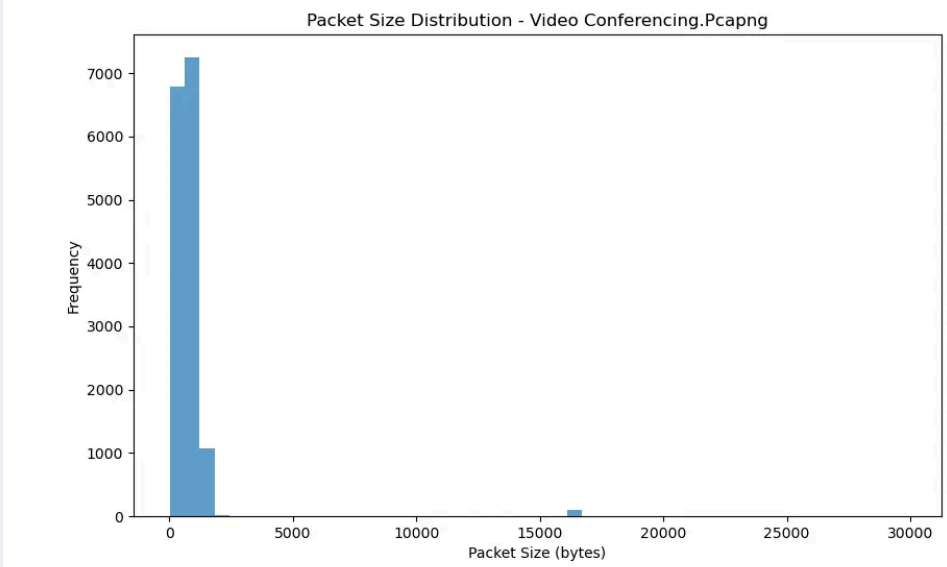
Packet Sizes











packet_sizes.webp

a. Audio Streaming (Spotify Podcast)

- The majority of packets are small-sized, concentrated at the lower end of the spectrum.
- A few larger packets appear sporadically, but they are rare.
- Indicates that audio streaming mainly consists of small, frequent data packets.

b. Web Surfing (Google Chrome)

- Most packets are small to medium-sized, with a dominant peak at low values.
- Some packets are moderately large, likely from downloading webpage assets like images or scripts.
- Reflects a mix of small control packets and occasional larger content downloads.

c. Web Surfing (Microsoft Edge)

- The pattern is very similar to Chrome.
- Small packets dominate, with a few large ones scattered.
- Suggests that both browsers follow a similar network behavior for web surfing.

d. Video Streaming (Twitch)

- The highest number of packets observed among all activities.
- Most packets are small but occur at an extremely high frequency.
- Some larger packets are present, possibly related to buffering or metadata.
- Suggests that Twitch streaming relies on a continuous flow of small data chunks.

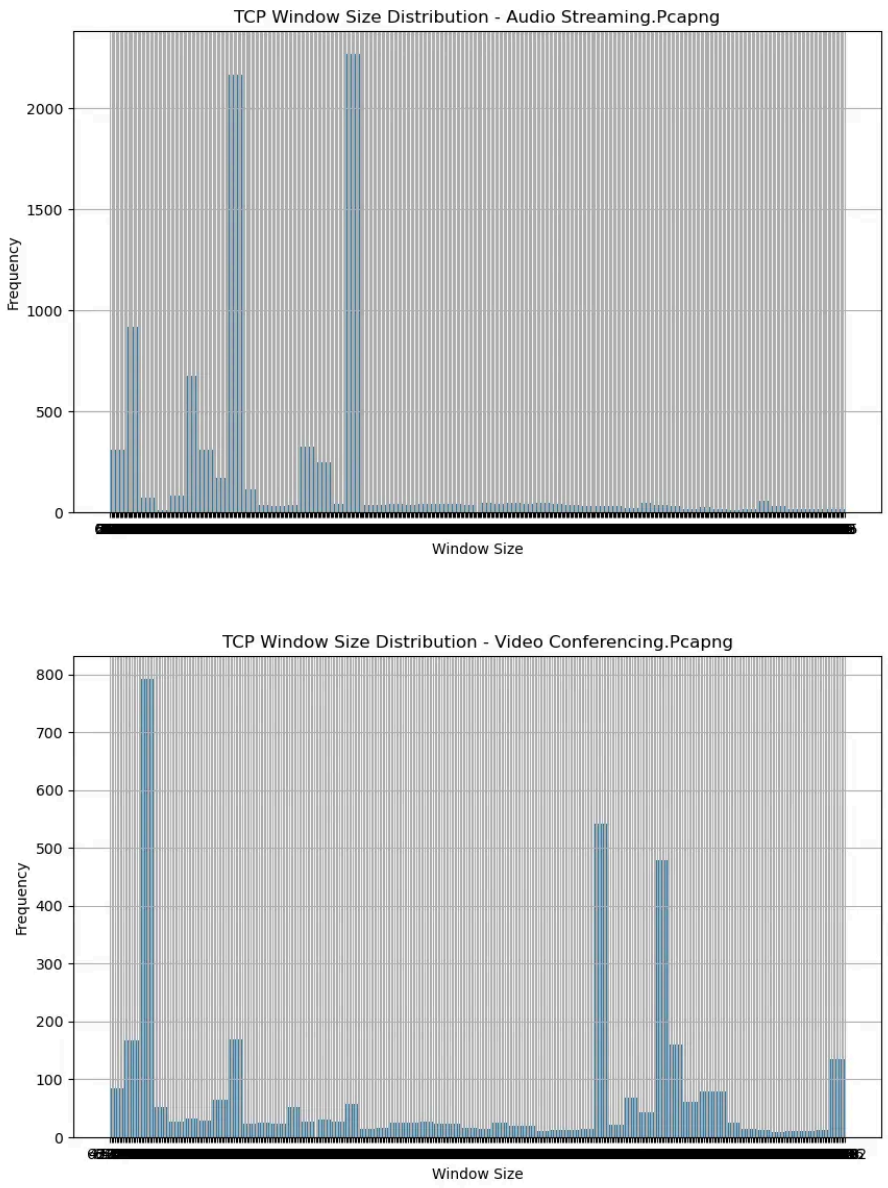
e. Video Conferencing (Zoom)

- Small packet sizes dominate, similar to audio streaming.
- A few larger packets appear, possibly for video keyframes or control messages.
- This suggests that Zoom uses many small packets for real-time communication rather than large data bursts.

Key Observations:

- Audio and Video Conferencing: Mainly small packets, as expected for real-time communication.
- Video Streaming (Twitch): The highest packet count, dominated by small data chunks.
- Web Browsing (Chrome & Edge): Mixed sizes, with some larger packets likely due to image and script downloads.
- Larger Packets Appear Occasionally: In all cases, but they are rare, meaning most communication happens with small packets.

TCP Headers



tcp_window_sizes.webp

a. Audio Streaming (Spotify Podcast)

- The window size distribution is highly varied, with multiple peaks at different sizes.
- Some dominant values appear frequently, possibly due to steady streaming behavior.
- This suggests that TCP flow control is adapting dynamically but with certain preferred window sizes.

b. Video Streaming (Twitch)

- A very large number of packets are present across a wide range of TCP window sizes.
- The distribution is spread out, indicating adaptive window scaling.
- The high frequency of large window sizes indicates good bandwidth utilization.
- This aligns with the need for continuous, high-speed data flow for video streaming.

c. Video Conferencing (Zoom)

- Peaks at specific window sizes, with several clusters.
- A mix of small and moderate window sizes suggests dynamic congestion control.
- This makes sense for video conferencing, where real-time data must be delivered with minimal latency.

d. Web Surfing (Microsoft Edge)

- A widely spread distribution with some dominant peaks.
- Large window sizes appear frequently, likely used when downloading larger assets like images and scripts.
- Shows that TCP windowing adjusts dynamically based on content type.

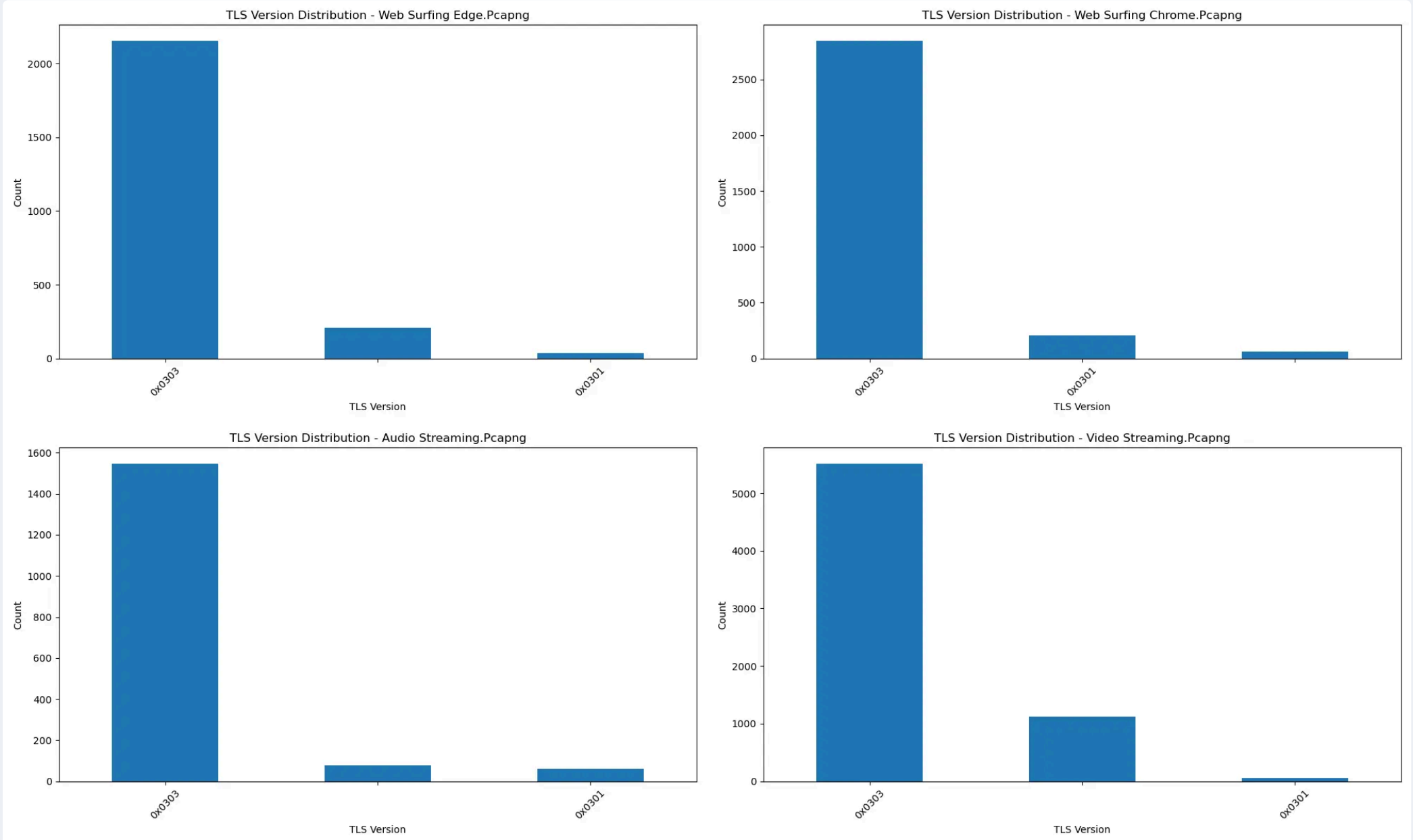
e. Web Surfing (Google Chrome)

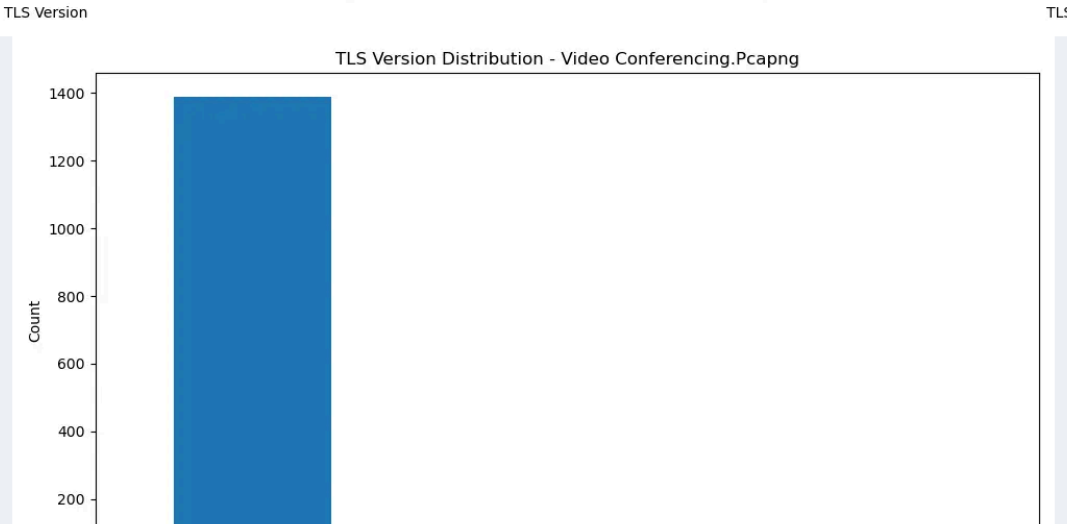
- Similar to the Edge case, with a diverse range of window sizes.
- Some window sizes appear significantly more frequently, likely due to standard TCP configurations.
- This suggests Chrome and Edge use comparable network optimization strategies.

Key Observations:

- Video Streaming (Twitch) has the highest number of packets, and the window size distribution suggests aggressive bandwidth utilization.
- Web surfing (Chrome & Edge) shows a dynamic distribution, reflecting the need to load varied content types.
- Audio and Video Conferencing have a more controlled distribution, likely due to real-time constraints where latency is more critical than throughput.
- Larger window sizes indicate resourceful usage, common in streaming and web browsing.
- Smaller window sizes are more common in conferencing, likely to avoid buffer buildup and latency issues.

TLS Headers





tls_versions.webp

a. Audio Streaming (Spotify Podcast)

- TLS 1.3 is dominant, with most packets using it.

- A small number of connections use TLS 1.2.
- This indicates that Spotify prefers the latest TLS version for encryption but still allows legacy connections.

b. Video Streaming (Twitch)

- TLS 1.3 is the primary protocol, used in the vast majority of packets.
- Some traffic uses TLS 1.2, but it is a minority.
- Since Twitch uses HTTPS-based delivery, secure connections are essential, and TLS 1.3 provides faster handshakes and better security.

c. Video Conferencing (Zoom)

- TLS 1.3 is dominant, with nearly all connections using it.
- A few packets rely on TLS 1.2, but it is rare.
- Secure encryption is crucial for real-time communications, and TLS 1.3 enhances performance by reducing latency.

d. Web Surfing (Google Chrome)

- TLS 1.3 is overwhelmingly used, indicating that most modern websites and services prefer it.
- A small number of TLS 1.2 connections exist, likely due to accessing older websites or certain CDNs.
- A very minimal presence of TLS 1.1, suggesting near obsolescence.

e. Web Surfing (Microsoft Edge)

- Very similar to Chrome.
- TLS 1.3 dominates, with a small number of TLS 1.2 connections.
- A few TLS 1.1 packets are present, likely due to compatibility with older sites.

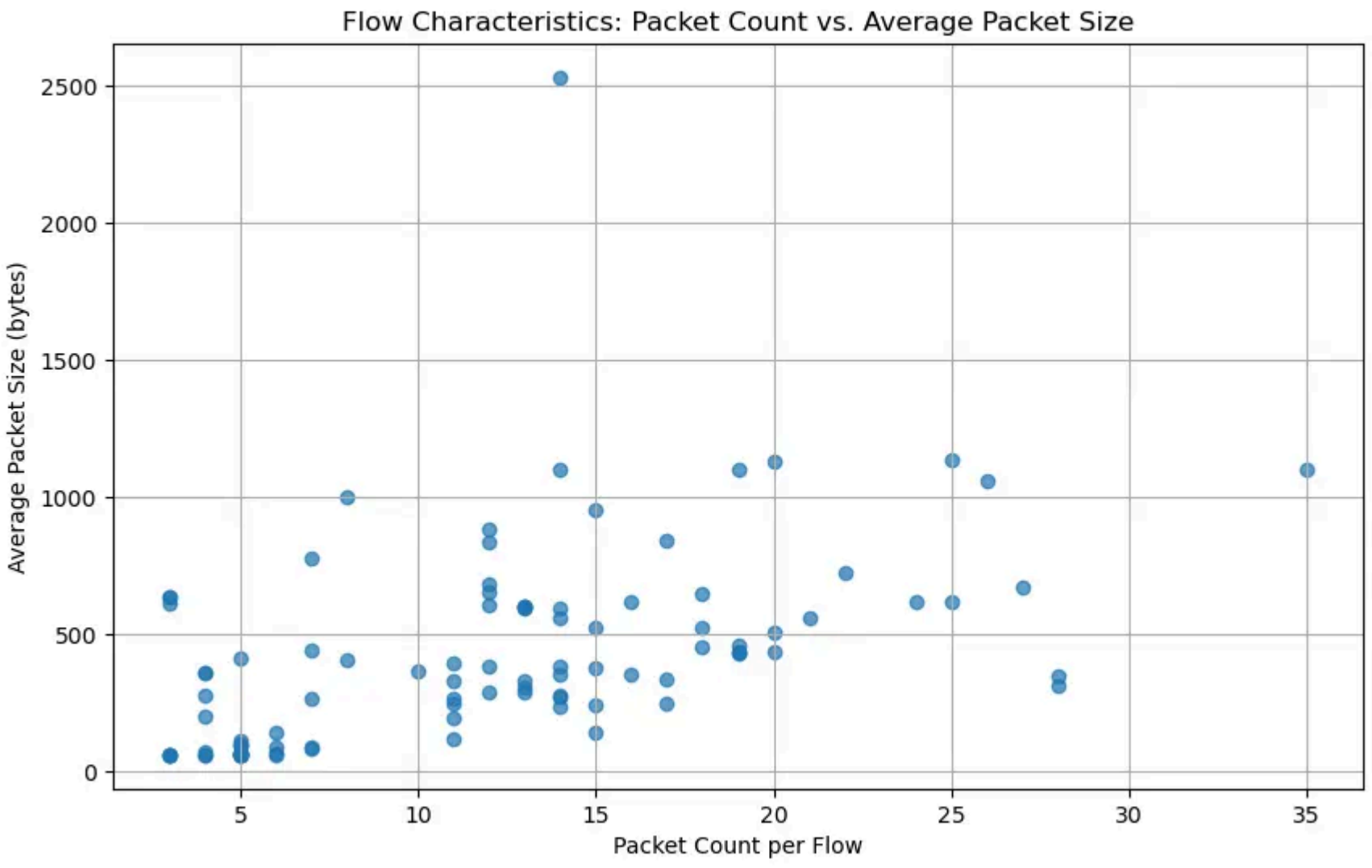
Key Observations:

- TLS 1.3 is the dominant encryption protocol across all activities.
- TLS 1.2 is still in use but rare, suggesting backward compatibility for older services.
- TLS 1.1 is nearly extinct, only appearing in small traces.
- Web surfing (Chrome & Edge) shows similar TLS usage patterns, likely because both browsers follow modern security standards.
- Real-time services like video conferencing and streaming strongly favor TLS 1.3, likely to optimize speed and security.

Attacker's View: Two Scenarios (Mixed Capture)

1. Attacker Knows Packet Size, Timestamp, and 4-Tuple Hash (Flow ID)

- Observation:



center

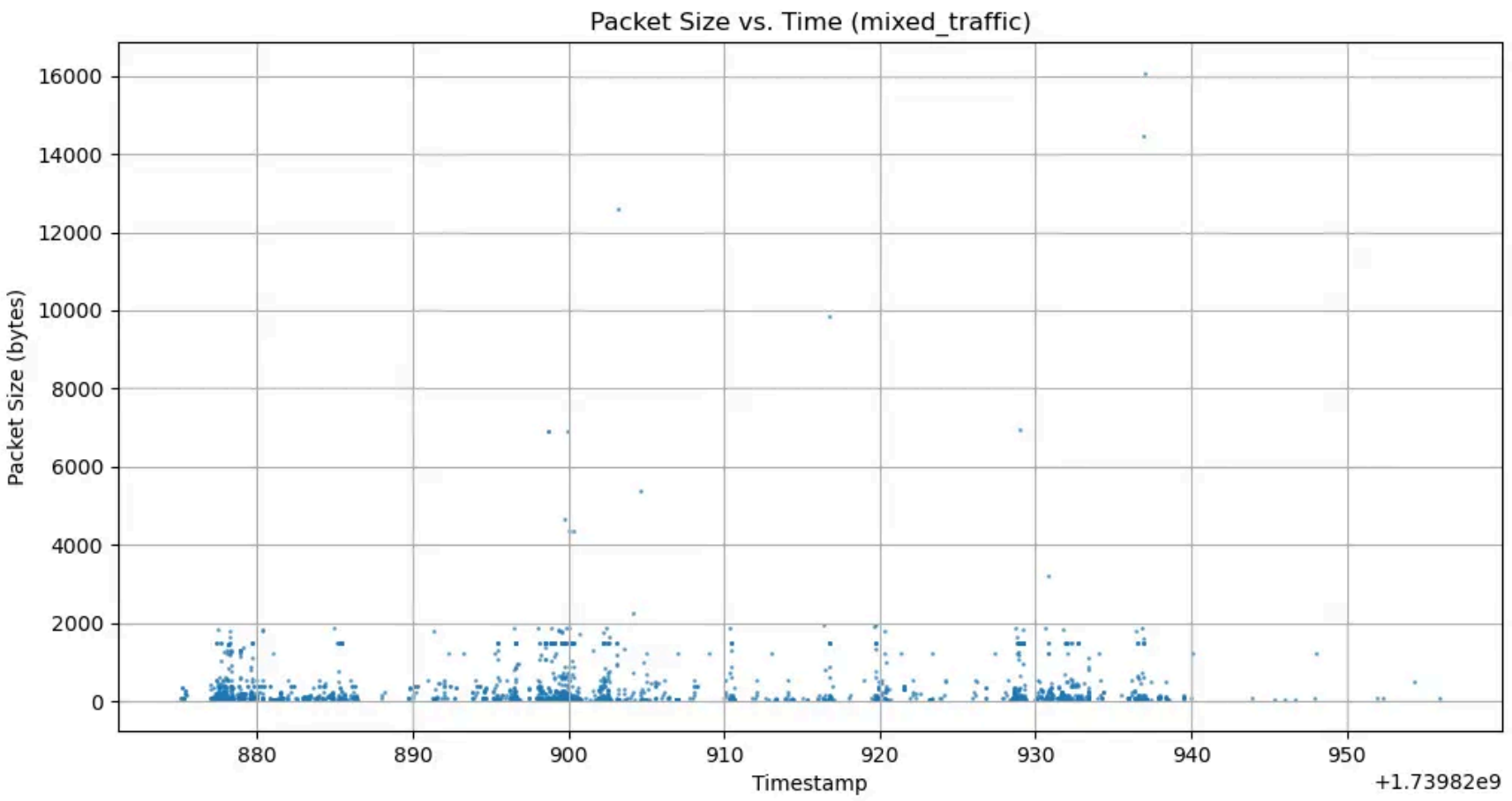
- Flow Identification: With the 4-tuple hash, the attacker can group packets belonging to the same connection. This means that instead of seeing isolated packets, the attacker sees complete flows (e.g., all packets belonging to an email session or

a video call).

- Pattern Recognition: By examining the flow statistics (e.g., duration, average packet size, packet count), the attacker can build “signatures” for different apps. For example, a video streaming session might show a continuous stream of large packets, while an email session might show bursty activity with smaller packets.
- Temporal Correlation: The exact timestamps help the attacker to notice when flows start and stop, possibly correlating those with known user behavior (e.g., the time a user started a video call).
- With the full metadata (size, timestamp, flow ID), the attacker has a richer dataset, making it relatively easier to fingerprint the type of application or even a specific site. This is because many applications exhibit distinctive traffic patterns (e.g., continuous streaming vs. sporadic bursts). This has been confirmed in the previous section, where we have observed the different and unique traffic behaviour for different applications.

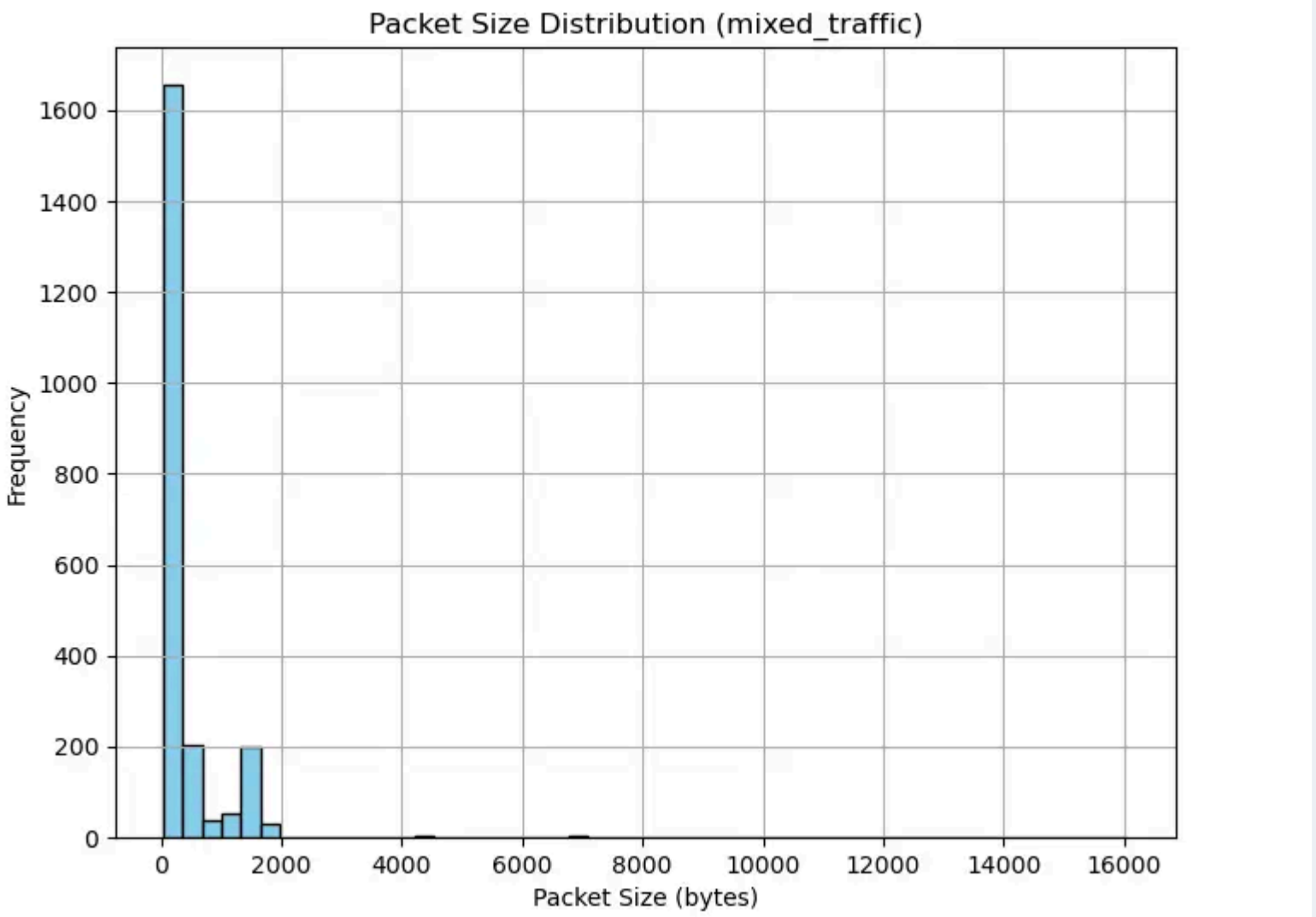
2. Attacker Knows Only Packet Size and Timestamp

- Observation:



center

- Limited Grouping Ability: Without the 4-tuple hash, the attacker cannot directly group packets into flows. However, they can still try to infer groupings based on the timing and size of packets. For instance, clusters of packets arriving in a short period may suggest a flow.
  - Traffic Fingerprinting: Even with just sizes and timestamps, different apps produce characteristic patterns. For example:
    - Web Browsing: Might display a mix of small control packets and larger data packets with irregular timing.
    - Audio Streaming: Often shows regular packet intervals with relatively consistent sizes.
    - Video Streaming or Conferencing: Typically produces high-volume, continuous streams with less variation in packet sizes.
  - Statistical Fingerprinting: An attacker can use statistical techniques (like histograms of packet sizes and inter-arrival time analysis) to guess the underlying application.



center

- Even without flow identifiers, the patterns in packet sizes and timestamps can serve as a fingerprint. While the attacker might face more noise and ambiguity (since they cannot perfectly correlate packets to flows), sufficient historical data, or thoroughly and highly tested and tried traffic models with high success rates, such as the ones discussed in the previous research papers could still allow reasonably accurate inferences about which site or app is in use.

Why Encryption/Anonymization May Not Fully Mitigate the Attack

- Metadata Exposure:
  Encryption typically protects the payload but not the metadata (i.e., packet sizes, timestamps, and even sometimes header information). As a result, even fully encrypted traffic still leaks information that can be used for fingerprinting, as not all metadata fields are encrypted.
- Unique Traffic Patterns:
  Many applications have unique and consistent traffic patterns (e.g., video streaming vs. web browsing). An attacker can build a library of these patterns and match observed traffic to known signatures. The accuracy of the hacker's data library increases the more the attacker invests in observing the victim's network behaviour and learning their usage patterns.

Mitigation Strategies

a. Packet Padding:
   - What: We may add extra data to packets so that all packets (or packets of a particular type) have a uniform, fixed size.
   - Benefit: This makes size-based fingerprinting much harder.

b. Traffic Shaping and Obfuscation:
   - What: We may introduce random delays or send dummy packets to mask true timing patterns.
   - Benefit: This disrupts the clear temporal patterns that attackers rely on.

c. Tunneling through VPNs or Anonymization Networks:
   - What: A user may wrap traffic inside another encrypted tunnel.
   - Benefit: This not only encrypts content but can also standardize packet sizes and timing to some extent, making it harder to correlate with specific apps.

d. Batching Data Transfers:
   - What: We may combine small transmissions into larger, less frequent ones.
   - Benefit: This reduces the granularity of observable traffic patterns.

Disadvantages for this general approach of obfuscation

...

To summarize

Based on the generated `.csv` data observed from the `mixed_traffic.pcapng` capture, and the plots generated by our script, we might observe clear differences in traffic patterns between different applications. The attacker's ability to infer the app in use increases when they have both the flow identifiers and full timing/size data. Even if only size and timestamp are available, distinctive patterns remain that can be exploited for fingerprinting.

By implementing mitigation strategies like packet padding and traffic shaping, it is possible to obscure these patterns, thereby reducing the risk of an attacker correctly identifying which site or app is being accessed, even when the payload is encrypted.

Submission instructions

The submission is in groups of up to 4 students.

a. GitHub: Open a GitHub repository. The repo should include:
   a. A `README` file that briefly explains the project and provides compilation/running instructions.
   b. Your final report, as a `.pdf` file.
   c. A directory named `/src/`, containing all your code.
   d. A directory named `/res/` containing all your result files that are not too large (e.g., figures).
   e. Do NOT upload large files (e.g., `.pcap` files), as GitHub's free account does not support this.
b. LinkedIn: Each student in the group should open a LinkedIn profile (if he doesn't have one yet), and link the GitHub's repo page to his profile as a post and/or under "projects".
c. Moodle: Submit a `.zip` file. The filename should include the IDs of all the students in the group, e.g. `012345678_9876543_77711111.zip`. The `.zip` file should include:
   a. A link to your GitHub repo. Including `README` that explain what you did and the summary of the papers.
   b. The `.pcap` files you recorded. If the `.pcap` files are large, upload them to the cloud and insert a link to them in the `.pdf` file. Please don't submit the `.pcap` as it is; consider filtering it.
   c. The code for parsing the `.pcap` files should run on every computer. Hence, every decision should be general and not specific to your computer. E.g., use a relative path and not the absolute path of your machine. Your code must correctly run on Linux, and will be checked on Linux.
   d. Write clean and clear code, including identifiers, function names, and parameters (instead of arbitrary numbers within the code).
   e. Handle edge cases and bugs, such as trying to open a file that does not exist and division by zero.
   f. You may use code you find on the Internet and even AI tools, but please explicitly write the sources and promotes you used.
   g. A private teacher or guide may help you, but they should not write your code (or parts from it) themselves.

Good luck!