# PART 1:

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?

To tackle the user's reported issue of slow file transfers a one should examine the transport layer. This layer, primarily managed by protocols like TCP and UDP, ensures that data is transmitted accurately and efficiently across networks. When performance falters, several factors could be at play.

Network congestion often emerges as a primary reason for sluggish file transfers. This occurs when multiple devices or applications vie for limited bandwidth, leading to delays as data packets queue up for transmission. For instance, in a busy office setting, simultaneous activities like video streaming and large file downloads can overwhelm the network, reducing the speed of individual transfers. High latency presents another challenge, particularly when the sender and receiver are separated by vast distances or when data takes inefficient routes through the network. This delay in packet travel time slows down the acknowledgment process in TCP.

Packet loss adds further complexity to the issue. When packets fail to reach their destination due to network errors, interference, or overloaded buffers, the transport protocol must retransmit the lost data, introducing additional delays. Even minimal packet loss can significantly affect performance, especially over long distances where retransmissions compound the problem.

Protocol inefficiencies also play a role, particularly with TCP. If settings such as window sizes are too small or congestion control algorithms are overly cautious, the amount of data sent at once is restricted, lowering overall throughput. In cases where UDP is used, the lack of built-in error correction can lead to incomplete transfers if not properly managed, necessitating manual retransmission efforts.

Beyond these core issues, application layer inefficiencies can indirectly influence transport layer performance. Software that is not optimized may fail to leverage available bandwidth fully or could introduce unnecessary overhead, further hampering transfer speeds.

The troubleshooting process begins with the use of network diagnostic tools like Wireshark to capture and analyze packet data. By inspecting the flow of packets, it becomes possible to detect signs of trouble, such as frequent retransmissions, out-of-order packets, or prolonged delays between acknowledgments. These clues can indicate whether congestion, latency, or packet loss is the primary culprit. Following this, bandwidth and latency tests offer a clearer picture of network performance. Tools like iperf measure the maximum achievable throughput, while ping or traceroute assess round-trip times and highlight potential bottlenecks along the path. If bandwidth is limited or latency is excessive, adjustments to the network infrastructure may be warranted.
Reviewing network configurations is another critical step. Quality of Service (QoS) settings on routers and switches can prioritize certain types of traffic, sometimes at the expense of file transfers. Ensuring that file transfer data receives appropriate priority can alleviate congestion-related delays. Additionally, optimizing the application responsible for the transfer is vital. Adjusting buffer sizes or enabling features like TCP window scaling can enhance efficiency.

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?

TCP's flow control mechanism plays a critical role in ensuring efficient and reliable data transmission, especially in scenarios where there is a disparity in processing power between the sender and receiver. Flow control in TCP is primarily managed through the **receiver's advertised window (rwnd)**, which dictates how much data the sender can transmit before requiring an acknowledgment from the receiver. This mechanism prevents buffer overflow at the receiver and ensures that data is processed at a sustainable rate.

When the **sender has significantly higher processing power than the receiver**, several performance implications arise:

**Receiver Buffer Saturation**: The receiver, having limited processing capabilities, may struggle to handle incoming data at the rate the sender can provide. This results in the receiver advertising a smaller **rwnd**, causing the sender to slow down transmission. If the receiver's buffer fills up too quickly, it may frequently advertise a zero window, leading to TCP's "Zero Window" state, where the sender halts transmission until the receiver can process more data. This introduces additional latency and reduces throughput.

**Increased Latency Due to Window Updates**: Since the receiver processes data slowly, its buffer will take longer to clear, leading to delays in window size updates. The sender, despite having high processing power, must wait for these updates before continuing data transmission. This creates **stop-and-wait behavior**, degrading overall throughput efficiency.

**TCP Window Probing (Zero Window Probes)**: If the receiver consistently advertises a zero window due to slow processing, the sender periodically sends "Zero Window Probes" (ZWP) to check if the receiver is ready to accept more data. Frequent probing increases overhead on the network and delays effective data transmission.

**TCP Congestion Control vs. Flow Control**: While TCP flow control is based on the receiver's buffer capacity, **TCP congestion control** (e.g., slow start, congestion avoidance) responds to network conditions. If the sender's high processing power enables rapid packet generation, but the receiver's limited processing power slows down acknowledgments, the sender might interpret the delay as network congestion. This can mistakenly trigger congestion control mechanisms, further reducing throughput unnecessarily.

**Impact on High-Throughput Applications**: Applications requiring high-speed data transfer, such as video streaming, large file transfers, or database replication, may experience **suboptimal performance** if the receiver's processing power is a bottleneck. The sender will not be able to fully utilize available bandwidth, resulting in **underutilization of network resources**.

**Possible Workarounds**: In some cases, techniques like **Selective Acknowledgment (SACK)** or **window scaling** can help mitigate performance degradation. SACK allows the receiver to acknowledge non-contiguous data segments, reducing retransmission overhead, while window scaling enables larger window sizes, reducing the frequency of acknowledgments. However, these solutions are limited if the receiver's processing power is inherently too low.

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?

**Role of Routing in a Network with Multiple Paths:**

Routing in a network plays a crucial role in determining the path that data packets take from the source to the destination. When multiple paths exist, the routing algorithm must decide which path is optimal based on various factors, impacting overall network performance. The primary goal of routing is to ensure efficient, reliable, and fast data transmission while minimizing congestion, latency, and packet loss.

**Impact of Path Choice on Network Performance:**

The choice of a specific path significantly affects the performance of the network. If a router selects a **high-latency path**, packets may take longer to reach their destination, increasing delay-sensitive application response times. In contrast, if a **high-bandwidth path** is chosen, the network can handle large amounts of data efficiently, improving throughput for applications such as video streaming and file transfers. However, selecting a path that is already experiencing congestion can lead to **packet drops, retransmissions, and jitter**, degrading the quality of service (QoS). An optimal routing decision helps balance traffic load across multiple paths, preventing bottlenecks and ensuring smooth data flow.

**Factors Considered in Routing Decisions:**

Several factors influence routing decisions when multiple paths exist. **Latency** is a critical factor, especially for real-time applications such as VoIP and online gaming, where excessive delay leads to performance degradation. **Bandwidth availability** determines how much data can be transmitted efficiently, making it essential for applications requiring high-speed data transfers. **Congestion levels** on different paths are also taken into account; routing algorithms often avoid heavily congested routes to maintain efficiency.

Another key consideration is **network reliability**. Some paths may be more stable than others, depending on the frequency of link failures, router downtimes, or fluctuating network conditions. Redundant paths provide **fault tolerance**, ensuring that if one route fails, traffic can be rerouted dynamically to another available path without service disruption.

In modern networks, **Quality of Service (QoS) policies** also affect routing decisions. Certain applications require prioritized traffic handling, so routers may implement policies to favor voice or video traffic over standard web browsing or file downloads. Additionally, in **Software-Defined Networking (SDN)**, centralized controllers analyze real-time network conditions and make intelligent routing decisions to optimize traffic flow dynamically.

Security concerns also influence path selection. **Some routes may be less secure**, exposing traffic to potential interception or attacks. Routing algorithms might prefer trusted paths with **encrypted tunnels (e.g., VPNs)** or avoid routes through potentially compromised nodes.

## 4. How does MPTCP improve network performance?

Multipath TCP (MPTCP) is an enhancement to the traditional TCP protocol that allows a single connection to utilize multiple network paths simultaneously. This capability significantly improves network performance by increasing throughput, enhancing reliability, and optimizing resource utilization.

**Throughput Enhancement and Load Balancing:**

MPTCP increases network throughput by distributing data across multiple paths, enabling better utilization of available bandwidth. Traditional TCP is limited to a single path, meaning that if one link has lower bandwidth or congestion, performance is restricted. With MPTCP, data is transmitted over multiple interfaces (e.g., Wi-Fi and cellular), aggregating their bandwidth and **increasing the overall transmission speed**. This feature is particularly beneficial for applications requiring high data rates, such as cloud services, real-time video streaming, and large file transfers.

**Improved Reliability and Fault Tolerance:**

By maintaining multiple subflows across different network paths, MPTCP provides resilience against **path failures or degradations**. If one path experiences congestion, high latency, or complete failure, MPTCP dynamically reroutes traffic through alternative paths without disrupting the session. This redundancy minimizes packet loss and ensures **continuous and stable connectivity**.

**Optimized Network Resource Utilization:**

MPTCP dynamically adapts to network conditions by **shifting traffic between paths based on real-time performance metrics**. When a network path becomes congested or degrades in quality, MPTCP automatically prioritizes faster, more efficient routes. This adaptability ensures that network resources are used optimally, reducing bottlenecks and preventing unnecessary load on a single path. This optimization leads to more **efficient bandwidth management, reduced congestion, and better overall network performance**. and also **reduces latency** and makes faster data delivery.

**Security and Redundancy Benefits:**

Using multiple paths for a single connection enhances security by **spreading data across different network interfaces**, making it more difficult for attackers to intercept complete communication streams. Additionally, if one path is compromised, MPTCP can continue transmitting data over secure alternatives, ensuring **higher availability and robustness against cyber threats**.

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.

Packet loss between two routers is a critical issue that affects network performance, causing delays, retransmissions, and degraded quality of service (QoS). To analyze the root causes, it is essential to examine both the **Network Layer** and the **Transport Layer** since each contributes to packet loss in different ways.

### Network Layer Causes of Packet Loss:

One of the primary reasons for packet loss at the network layer is **congestion**. When routers are overwhelmed with incoming traffic that exceeds their processing capacity or available buffer space, packets are dropped due to **queue overflow**. This typically occurs in high-traffic environments or when **bandwidth is insufficient** to handle the data load. Another major cause is **routing issues**, where suboptimal path selection, frequent route changes, or misconfigured routing protocols result in packets being sent on unstable or overloaded routes, leading to packet drops. **Network hardware failures** such as faulty cables, damaged interfaces, or malfunctioning routers also contribute to packet loss. If a router's interface is physically degraded or if a fiber-optic link has high attenuation, packets may be lost before reaching the next hop. **Packet fragmentation** is another key factor, as large packets that exceed the Maximum Transmission Unit (MTU) must be split into smaller fragments. If a fragment is lost, the entire packet must be retransmitted, increasing packet loss rates.

### Transport Layer Causes of Packet Loss:

At the transport layer, **buffer overflows at the receiver's end** can lead to dropped packets, particularly when the sender transmits data faster than the receiver can process. This is often exacerbated by **poor TCP flow control mechanisms**, where the advertised window (rwnd) fails to adapt quickly to changing network conditions, resulting in unacknowledged packets being discarded. **High retransmission rates** due to excessive packet loss can also cause performance degradation, as TCP will continuously resend packets, leading to unnecessary network congestion. In UDP-based communications, **packet loss is more severe** because UDP lacks retransmission mechanisms, meaning that lost packets are never recovered unless handled at the application layer. Another transport-layer issue is **aggressive congestion control algorithms**, such as TCP Reno or TCP Cubic, which react to perceived congestion by reducing the congestion window (cwnd). If the network is falsely detected as congested due to random packet drops, performance will be unnecessarily throttled.

### Solutions to Mitigate Packet Loss:

Resolving packet loss requires targeted solutions at both layers. At the network layer, **reducing congestion through Quality of Service (QoS) mechanisms** can help prioritize critical traffic while preventing unnecessary drops. Implementing **load balancing** across multiple paths can distribute traffic more efficiently, avoiding bottlenecks. Upgrading network hardware may will solve some of the issues.

At the transport layer, optimizing **TCP window scaling** allows for better handling of varying network conditions, ensuring that flow control adapts dynamically to receiver capacity. Using **Selective Acknowledgment (SACK)** reduces unnecessary retransmissions by only resending lost packets instead of entire segments. (In networks that rely on UDP, Forward Error Correction (FEC)

mechanisms can be applied at the application layer to reconstruct lost packets.) If packet loss is incorrectly triggering congestion control mechanisms, **using a more adaptive congestion control algorithm like BBR (Bottleneck Bandwidth and Round-trip propagation time)** can improve performance in high-latency networks.

# PART 2:

FlowPic Encrypted Internent Traffic classification is as easy as image recognition

- What is the main contribution of the paper?

The main purpose of the paper is to introduce and evaluate FlowPic, a novel approach for classifying encrypted internet traffic. The authors argue that traditional methods, which rely on handcrafted feature extraction and shallow machine learning, struggle with the increasing prevalence of encryption (VPNs, Tor). Their core idea is to represent network flows as images, termed "FlowPics," and then leverage the power of Convolutional Neural Networks (CNNs), a deep learning technique highly successful in image recognition, to classify the traffic.

- What traffic features does the paper use, and which are novel?

The paper uses flow-based features, specifically packet sizes and packet arrival times, as the basis for creating FlowPics. These are *not* novel features in themselves; many previous works have used these. However, the *way* these features are used is novel.

FlowPic as a 2D Histogram: The key innovation is the creation of FlowPics. Instead of using packet sizes and arrival times as direct inputs to a classifier, they construct a two-dimensional histogram. The x-axis represents normalized packet arrival time, and the y-axis represents packet size and The pixel intensity represents the number of packets with a specific size arriving at a specific time. This 2D histogram, visualized as an image, becomes the input to the CNN. This representation captures the *distribution* of packet sizes over time, which is richer information than simply looking at individual packet sizes or arrival times. This specific 2D histogram representation is novel.

Deep Learning for Traffic Classification:
By leveraging Convolutional Neural Networks (CNNs) on these FlowPic images, the method circumvents the need for complex feature engineering. The CNN automatically learns relevant spatial

patterns within the FlowPic, which is a significant departure from traditional hand-crafted feature approaches.
Traditional traffic classification models typically rely on recognizing specific applications (e.g., if the data pattern resembles YouTube, it is classified as video).
However, their method learns general patterns of traffic categories, such as how video traffic behaves in general, rather than just identifying specific services like YouTube or Netflix.

Image-Based Classification: The use of FlowPics allows the authors to leverage the power of CNNs, which are designed for image recognition. This is a novel application of CNNs in the context of network traffic classification. While some prior work used neural networks for traffic classification, they typically used them with hand-crafted statistical features or directly on packet payloads. The idea of converting flow data into images and then using CNNs for classification is the core novelty of this paper.
Encryption-Agnostic Representation:
The approach works equally well for unencrypted traffic as well as traffic encrypted via VPN or Tor. Since it uses only packet metadata (size and timing), it avoids privacy concerns associated with deep packet inspection.

Also, unlike previous approaches, which relied on time-based features and split the classification task into two separate steps (VPN vs. Non-VPN and Tor vs. Non-Tor), their method addresses both problems simultaneously.
While their approach achieved 89.0% accuracy for VPN classification and 94.8% for Tor, the new model operated under more challenging conditions (by handling all encryption types in a single task) and still outperformed their results in both cases.

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

High Accuracy in Most Scenarios:
The approach achieves very high accuracy in classifying non-VPN and VPN traffic (85.0% and 98.4% respectively for traffic categorization, and up to 99.7% in class-vs-all and application identification tasks). This demonstrates that the FlowPic representation captures the essential characteristics of these flows effectively.

Challenges with Tor Traffic:
Tor traffic categorization lags behind (67.8% for multi-class categorization and 85.7% for class-vs-all). The lower performance is attributed to the way Tor aggregates and encrypts traffic, which smooths out some of the distinguishing features that the CNN relies on.

Generalization to Unseen Applications:
One striking result is the method's ability to correctly classify applications that were not part of the training set. For example, even when the model is trained without data from a specific application (such as Facebook video), it still achieves nearly 99.9% accuracy when classifying that application. This indicates that the CNN learns intrinsic traffic behavior rather than merely memorizing application-specific details.

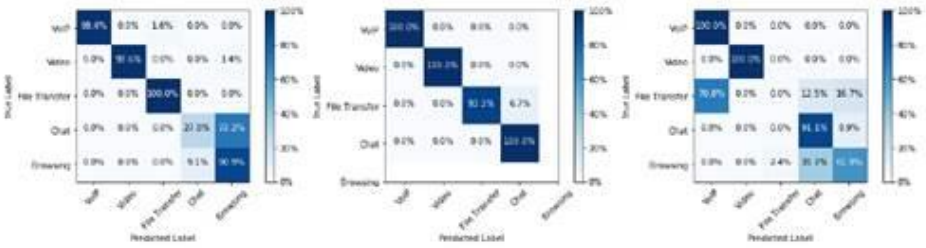
Robustness to Encryption:
The FlowPic method works effectively across different encryption types (non-VPN, VPN, and even Tor for certain tasks), making it a versatile tool for real-world network environments where encryption is common. their network achieves an accuracy of 88.4% classifying encryption Techniques.

Reduced Need for Feature Engineering:
By transforming raw flow data into an image, the method removes the need for the complex, manual feature extraction process. This not only simplifies the pipeline but also leverages the well-developed field of image recognition to address traffic classification.

Efficient Classification Using Short Time Window:
Our model successfully classifies internet traffic using only a short time window of a unidirectional flow, without requiring full bidirectional session analysis (usually we need to analyze packets from client to server and server to client). This enables faster processing, reduced computational overhead, and real-time deployment, while still achieving high accuracy, even for encrypted traffic (VPN, Tor).

Practical Considerations:
The method requires only two pieces of information per packet (size and time), resulting in minimal storage and computational overhead. This efficiency makes it feasible for real-time deployment in network monitoring and security systems.

the first image explains how well FlowPic classifies different traffic categories in detail, while the second image summarizes the model's overall performance across multiple classification tasks.



(a) Non-VPN (b) VPN (c) Tor

Figure 4: Confusion matrices of the 3 traffic categorization problems; over non-VPN, over VPN, and over Tor.

| Problem | FlowPic Acc. (%) | Best Previous Result | Remark |
|---|---|---|---|
| *Non-VPN Traffic Categorization* | 85.0 | 84.0 % Pr., Gil *et al.* [15] | Different categories. [15] used unbalanced dataset |
| *VPN Traffic Categorization* | 98.4 | 98.6 % Acc., Wang *et al.* [7] | [7] Classify raw packets data. Not including browsing category |
| *Tor Traffic Categorization* | 67.8 | 84.3 % Pr., Gil *et al.* [15] | Different categories. [15] 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. % Acc., Wang *et al.* [7] | [7] Classify raw packets data, not including Tor category |
| *Applications Identification* | 99.7 | 93.9 % Acc., Yamansavascilar *et al.* [10] | Different classes |

early traffic classification with encrypted clientHello

- What is the main contribution of the paper?

This research tackles the early traffic classification (eTC) problem, which aims to determine the type of transmitted data (such as video streaming, web browsing, or VoIP calls) to ensure optimal Quality of Service (QoS) in modern networks. Today, over 97% of global internet traffic is encrypted using Transport Layer Security (TLS), making traditional classification methods ineffective. Previously, traffic classification relied on metadata such as the Server Name Indication (SNI), which reveals the domain name a client connects to. However, with the introduction of Encrypted ClientHello (ECH) in TLS 1.3, such identifying information is now hidden, significantly complicating real-time traffic classification. Despite this challenge, not all TLS parameters are fully encrypted, leaving certain elements that still provide hints about the type of traffic being transmitted. The paper introduces a novel traffic classification algorithm, the hybrid Random Forest Traffic Classifier (hRFTC), which leverages these remaining unencrypted TLS parameters alongside flow-based statistical features such as packet size distributions and inter-arrival times. By combining these features, hRFTC provides a robust classification approach that works effectively even under ECH encryption.
additionally, it highlights critical gaps in current eTC research and sets the foundation for future studies on QoS impact, QUIC migration handling, P2P classification, and multi-flow correlation analysis. By addressing the fundamental problem of encrypted traffic classification in an increasingly privacy-focused internet, the research presents a scalable, practical solution that is applicable to real-world network management and security scenarios.

- What traffic features does the paper use, and which are novel?

The paper uses a combination of packet-based and flow-based traffic features to classify encrypted traffic effectively, particularly in environments where Encrypted ClientHello (ECH) is enabled in TLS 1.3. The key innovation lies in its new hybrid approach, which integrates traditional packet-level features with statistical flow-level features, allowing it to remain effective even when conventional TLS-based metadata is hidden.

The packet-based features include TLS handshake metadata, where unencrypted elements from the ClientHello (CH) and ServerHello (SH) messages are analyzed. But since ECH encrypts the SNI (Server Name Indication), these remaining unencrypted fields provide the classifier with limited distinguishing data so that using this features alone are insufficient when ECH is active.
One of the key innovations in hRFTC is its enhanced recomposed payload representation, which extends beyond RB-RF by incorporating additional TLS metadata that remains visible despite encryption, such as the lengths of Encrypted Extensions and Certificate Chains. Instead of relying purely on raw handshake payload bytes, hRFTC identifies structural variations in these extensions and maps them into a standardized feature vector. This method ensures that even when traditional classifiers fail due to SNI encryption, hRFTC can still extract meaningful differences between services.

Flow-based features play a crucial role in classification, particularly packet size (PS) statistics, which include mean, median, variance, and distribution-based analysis in both uplink (UL) and downlink (DL)

traffic. So even when ECH is active, traffic types still exhibit distinct packet size patterns, making classification possible. Another significant feature is inter-packet time (IPT) analysis, which tracks timing patterns between packets to distinguish different traffic behaviors. While IPT can be affected by network conditions, it remains useful for differentiating real-time communication from buffered media. A key innovation in hRFTC is its early flow statistics approach, which analyzes only the packets leading up to the first application data packet. Unlike traditional flow-based models that require thousands of packets, this method enables classification before actual payload transmission, making it highly efficient for real-time QoS management. This marks a major improvement over previous classifiers, which required extended observation periods before achieving accurate results.

Another significant contribution is the adaptation of hRFTC to QUIC-specific traffic, a necessity given that HTTP/3 relies on QUIC rather than TCP. Since traditional TCP-based features are no longer applicable in QUIC, the study extends the RB-RF classifier to handle QUIC flows by incorporating QUIC Connection IDs, Initial Packet sizes, and other QUIC handshake properties. The study also discovers that QUIC PADDING frames, which normalize handshake packet sizes, complicate classification but do not entirely prevent it.

To sum up, RFTC introduces a low-latency feature extraction method, using only the handshake and the first few service packets, which makes it significantly more efficient than prior approaches. Additionally, unlike existing classifiers, the study identifies QUIC handshake behavior and TLS extension lengths as key features that remain unchanged by encryption and can still be leveraged for classification.
Overall, the paper innovates by integrating both packet and flow features, extending traditional methods to work with TLS 1.3 and QUIC, and discovering previously unknown classification signals in encrypted traffic. These advancements allow hRFTC to outperform all state-of-the-art classifiers, particularly in environments where ECH is used.

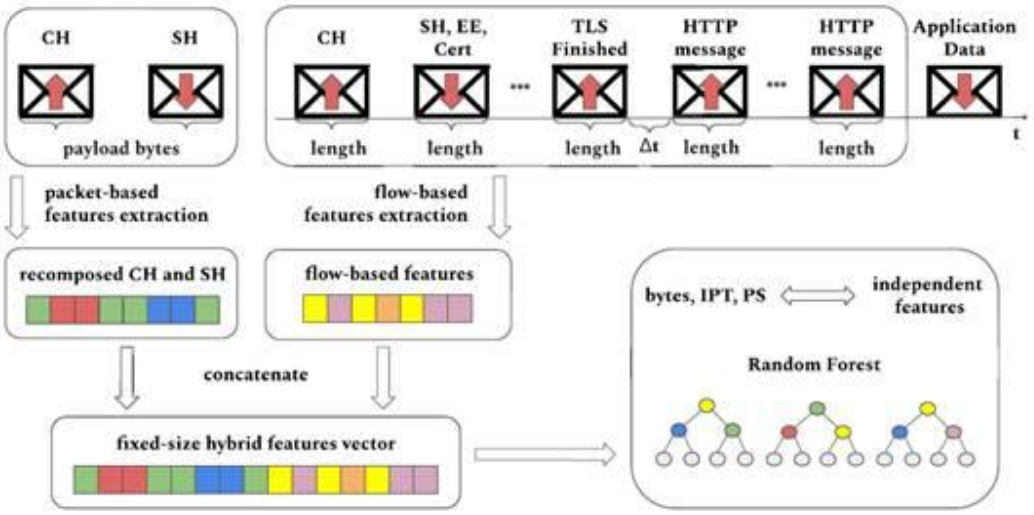



FIGURE 2. Hybrid random forest traffic classifier.

This diagram represents how hRFTC extracts packet-based and flow-based features from encrypted traffic, processes them using a machine learning classifier (Random Forest), and achieves highly accurate traffic classification.

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

Adding to the previous answers, we now will concentrate on the main results from the study. To validate the performance of hRFTC, the authors collected a large-scale encrypted traffic dataset from six countries across North America, Europe, and Asia. And got the following results:

hRFTC Achieves High Accuracy: The results demonstrated that using only the available unencrypted TLS parameters results in a classification F-score of just 38.4%, highlighting the limitations of relying solely on TLS handshake metadata. However, by incorporating flow-based statistical analysis, hRFTC achieves an unprecedented F-score of 94.6%, significantly outperforming all existing state-of-the-art classifiers.
The paper rigorously compares hRFTC's performance against several state-of-the-art traffic classification algorithms, encompassing packet-based, flow-based, and hybrid methods. hRFTC consistently outperforms these existing classifiers in terms of F-score, thereby proving its superiority in handling ECH traffic. This highlights the effectiveness of their specific combination of features and their novel packet selection strategy.

**TABLE 11.** Full dataset per class F-score for different 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.

Robustness with Limited Training Data: An important finding is that hRFTC maintains strong performance even when trained with a significantly reduced dataset. The authors note it performs admirably with just 10% of the data, exceeding the performance of other methods trained with 40%. This has significant practical implications, as it suggests that hRFTC can be deployed and updated more efficiently with less labeled data. Also, reducing training data from 70% to 10% led to only a 7% drop in macro F-score.

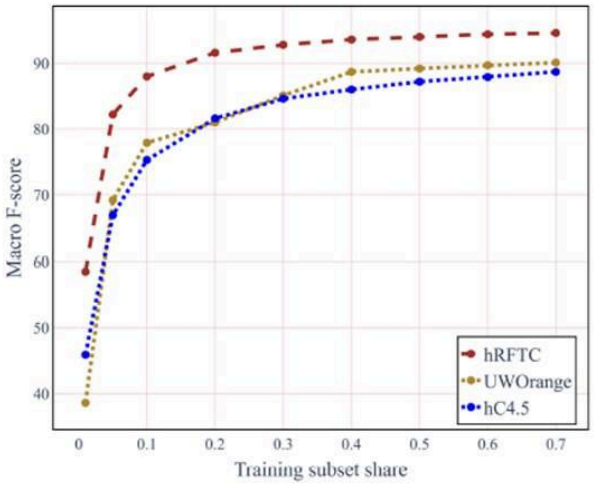



**FIGURE 4.** F-score depending on the training subset share.

performance in new geographic locations:
The study also tested how well classifiers handle traffic from previously unseen regions. When trained on one set of countries and tested on another, all classifiers (including hRFTC) experienced significant performance drops. Geographical differences in traffic patterns, Content Delivery Network (CDN) routing, and network infrastructure impact classification accuracy.
While hRFTC generalizes well within known locations, it struggles with unseen geographic regions, requiring retraining on region-specific datasets for optimal performance.
For example, the "Share in Dataset" column shows that some countries, like Russia and Germany, had a large presence in the dataset.
Despite this, when tested separately, these countries still exhibit lower F-scores, proving that the issue is not just about dataset size but also regional traffic differences.

**TABLE 14.** TC quality depending on training locations.

| Test Country | Share in Dataset | Training Country | Classifier Macro F-score [%] | | |
|---|---|---|---|---|---|
| | | | hRFTC | hC4.5 | UW |
| Germany | 18.8% | Others | 38.4 | 26.9 | 19.5 |
| Kazakhstan | 3.0% | Others | 57.3 | 32.3 | 27.5 |
| Russia | 29.2% | Others | 49.8 | 35.6 | 20.9 |
| Spain | 16.3% | Others | 38.5 | 34.4 | 12.6 |
| Turkey | 25.2% | Others | 35.1 | 26.0 | 16.4 |
| USA | 7.5% | Others | 49.2 | 41.4 | 21.3 |

Role of Packet Size Features and QUIC Padding: The research highlights the importance of packet size features in achieving accurate classification. However, they also observe that QUIC PADDING, which makes all QUIC handshake packets the same length, complicates traffic classification. While QUIC PADDING does impact accuracy, the authors conclude that its influence is limited. This suggests that while QUIC PADDING presents a challenge, it doesn't entirely negate the usefulness of packet size information or prevent effective classification.

The study raises several important research challenges that remain unsolved:

QUIC allows connections to switch networks without renegotiating TLS, which removes handshake visibility.
How can classifiers adapt to mid-flight connection changes?
Current classifiers focus on centralized services, but P2P traffic remains largely unclassified under ECH encryption.
Can correlated network flows improve classification? Many applications generate multiple concurrent flows (e.g., a video stream + ad requests).
How can classifiers leverage correlations between multiple flows to improve accuracy?

Analyzing HTTPS Encrypted traffic to identify User's Operating system, Browser and Application

- What is the main contribution of the paper?

The main contribution of the paper is that they are showing that they are able to classify encrypted network traffic and to infer which operation system, browser and application the user is using on his desktop or laptop computer, solely from HTTPS-encrypted traffic. Using supervised machine learning, particularly SVM with RBF, the authors achieve a high classification accuracy by leveraging both traditional traffic features and novel characteristics such as browser bursty behavior and SSL handshake patterns. They provide a dataset of over 20,000 labeled sessions, covering various OS, browsers, and applications.
The findings reveal serious privacy risks. This opens the door to passive surveillance, targeted attacks, and commercial tracking without decryption. The results highlight the need for stronger defenses against network-level privacy breaches and potential attackers.

- What traffic features does the paper use, and which are novel?

Demonstration of HTTPS Traffic-Based Identification:
This paper is the first to demonstrate that an attacker can accurately identify a user's operating system, browser, and application purely from HTTPS-encrypted traffic. The authors achieve this using supervised machine learning, specifically Support Vector Machines (SVM) with a Radial Basis Function (RBF) kernel, reaching an impressive 96.06% accuracy. This success is due to a combination of traditional traffic features and newly introduced features that exploit subtle behaviors in encrypted network flows.

Traffic Features Used in the Study:
The traffic analysis is based on two sets of features: baseline features, which are commonly used in previous research, and new features that the authors introduce to improve classification accuracy. The baseline features include packet-based statistics such as the number of packets sent and received, total bytes transferred, and packet sizes. Timing features, including inter-arrival time differences between packets, as well as session-based attributes like total session packets and throughput rates, are also considered. These conventional metrics alone allow for classification with 93.52% accuracy, but they are insufficient to capture the full depth of the network fingerprint.

Novel Features Introduced for Enhanced Accuracy:
To improve classification, the authors introduce new traffic features based on SSL/TLS behaviors, TCP/IP characteristics, and browser burst patterns. The number of SSL compression methods, the list of supported cipher suites, and the length of SSL session IDs are found to be unique among different browsers and applications, allowing for more precise differentiation. TCP attributes such as the initial window size, scaling factor, and maximum segment size also contribute to OS classification, as different operating systems configure these parameters differently. Additionally, the authors identify a bursty behavior pattern in browser traffic, where data transmission occurs in peaks followed by silent periods. By analyzing peak throughput, inter-arrival time between bursts, and the frequency of bursts in both directions, the model is able to further refine its classifications.

By integrating these new features with traditional ones, classification accuracy improves significantly, reaching 96.06%.

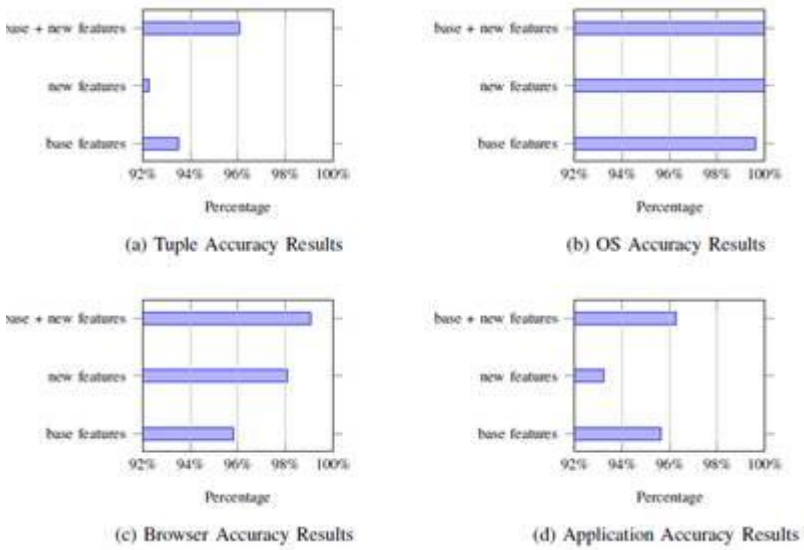
Security and Privacy Implications:
This finding challenges the assumption that HTTPS encryption is sufficient to protect user privacy. Even without decrypting traffic, attackers can infer a user's operating system, browser, and application based on network flow patterns. This opens the door to passive surveillance, targeted cyberattacks, and commercial tracking, as companies and adversaries alike could exploit this method to identify and profile users without their knowledge.

Study Strengths and Future Considerations:
the findings highlight a serious privacy concern, demonstrating that encryption alone does not guarantee anonymity. Future research will need to focus on countermeasures that can mitigate these traffic fingerprinting techniques and improve online privacy protections.

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

The paper demonstrates that an attacker can classify a user's operating system (OS), browser, and application from HTTPS-encrypted traffic with high accuracy. The authors train a Support Vector Machine (SVM) with an RBF kernel, testing three different feature sets: baseline features, new features, and a combination of both.

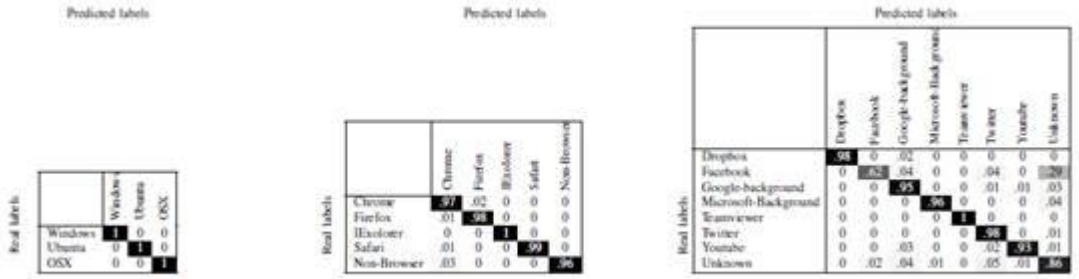
Using baseline features alone, classification accuracy reaches 93.52%.
Using new features alone, the accuracy is similar but slightly lower than when using both feature sets.
Combining baseline and new features increases the classification accuracy to 96.06%, confirming that the newly introduced features provide valuable additional information.
The results show that OS classification is nearly perfect, browser classification is highly reliable, and application classification remains accurate, though some misclassifications occur between background traffic and unknown labels.



(a) Tuple Accuracy Results
(b) OS Accuracy Results
(c) Browser Accuracy Results
(d) Application Accuracy Results

The OS confusion matrix indicates near-perfect classification, with almost zero misclassifications. Browser classification is highly accurate, with occasional mix-ups between Chrome and Firefox. Application classification has minor confusion between background traffic and unknown labels, but it

remains mostly reliable. With the use the Support Vector Machine (SVM) and with Radial Basis Function (RBF) as the kernel function they had excellent performance.
(Interestingly, Facebook traffic is misclassified as "Unknown" in 29% of cases due to its background activity, that is because Facebook lacks a distinct network signature, as it combines text, images, videos, etc.)



(b) OS Confusion Matrix (c) Browser Confusion Matrix (d) Application Confusion Matrix

Insights:
OS Classification is Nearly Perfect:
The results show that the model can identify a user's operating system with almost 100% accuracy. This suggests that OS-level network configurations, TCP/IP stack parameters, and SSL/TLS implementations introduce distinct traffic patterns that are difficult to mask.

Browser Classification is Highly Accurate:
Browser classification reaches accuracy levels exceeding 97%, confirming that different browsers generate unique traffic patterns due to variations in how they handle SSL sessions, initiate connections, and manage data bursts. These findings suggest that browser developers might need to introduce randomization or obfuscation to prevent passive fingerprinting.

Application Classification is Strong but Not Perfect:
Application classification remains highly effective, with some errors occurring primarily between background traffic and unknown labels. This is expected since some applications generate similar traffic patterns when operating in the background. Additionally, some traffic labeled as "unknown" could still belong to an application category, making it difficult to verify correctness.

New Features Improve Accuracy Significantly:
The addition of new features—especially SSL/TLS behaviors, TCP attributes, and browser burst analysis—boosts accuracy from 93.52% to 96.06%. This confirms that these features capture subtle but useful distinctions between operating systems, browsers, and applications. The browser's bursty behavior, in particular, proves to be a highly reliable fingerprinting metric.

Encryption Alone is Not Sufficient for Privacy:
Despite HTTPS encryption, traffic metadata and flow patterns still reveal a user's OS, browser, and application. This highlights a major privacy risk, as adversaries can passively monitor encrypted traffic and infer detailed user information without needing to decrypt the payload. The study challenges the assumption that HTTPS fully protects user anonymity.

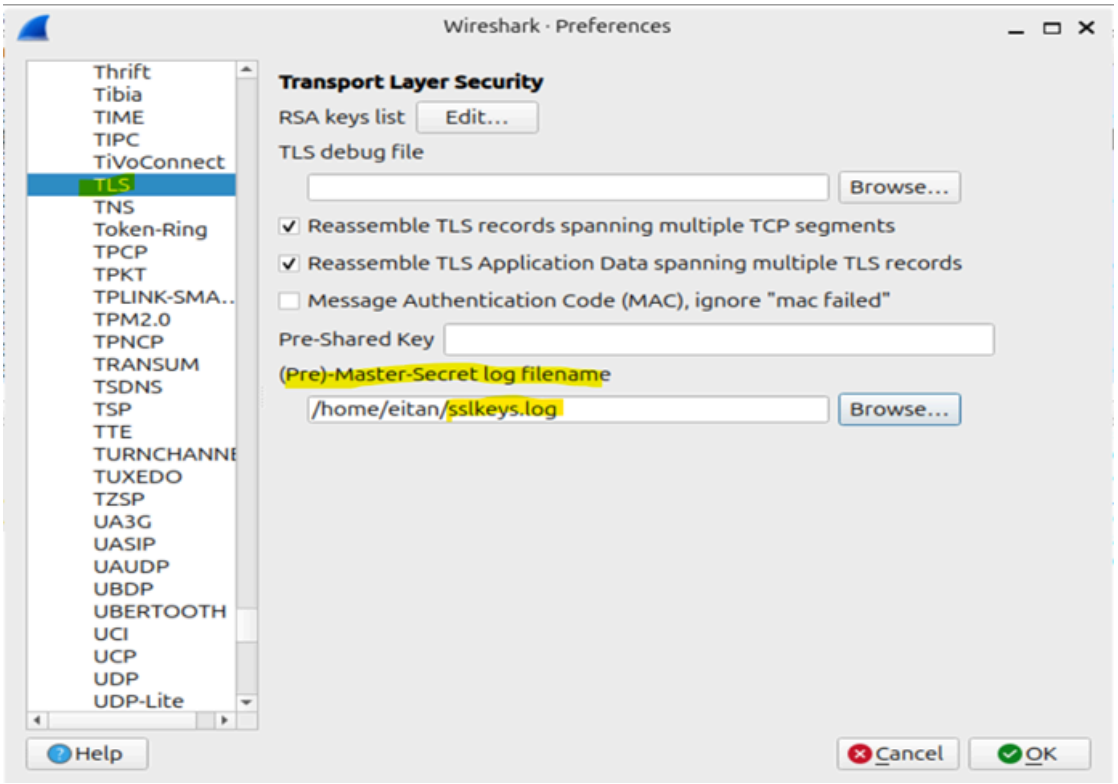
# PART 3:

## 1. הקלטות ב-Wireshark

### שמירת מפתחות ה-TLS

נשמור את מפתחות ה-TLS בקובץ


נוסיף את המפתחות ל-Wireshark כדי לפענח את התעבורה:

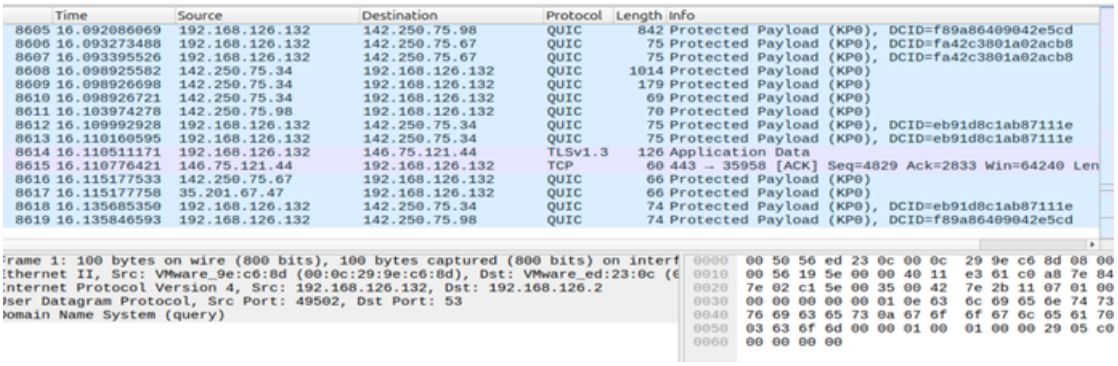


## הקלטה 1- Chrome Browsing

נפתח הקלטה חדשה ב-Wireshark

נגלוש בדפדפן כרום (ניכנס לסתם אתרים).

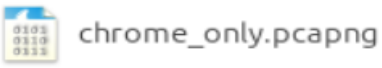
נעצור את ההקלטה ונשמור אותה בשם קובץ pcapng.



| | Time | Source | Destination | Protocol | Length | Info |
|---|---|---|---|---|---|---|
| 8605 | 16.092086069 | 192.168.126.132 | 142.250.75.98 | QUIC | 842 | Protected Payload (KP0), DCID=f89a86409042e5cd |
| 8606 | 16.093273488 | 192.168.126.132 | 142.250.75.67 | QUIC | 75 | Protected Payload (KP0), DCID=fa42c3801a02acb8 |
| 8607 | 16.093395526 | 192.168.126.132 | 142.250.75.67 | QUIC | 75 | Protected Payload (KP0), DCID=fa42c3801a02acb8 |
| 8608 | 16.098925582 | 142.250.75.34 | 192.168.126.132 | QUIC | 1014 | Protected Payload (KP0) |
| 8609 | 16.098926698 | 142.250.75.34 | 192.168.126.132 | QUIC | 179 | Protected Payload (KP0) |
| 8610 | 16.098926721 | 142.250.75.34 | 192.168.126.132 | QUIC | 69 | Protected Payload (KP0) |
| 8611 | 16.103974278 | 142.250.75.98 | 192.168.126.132 | QUIC | 70 | Protected Payload (KP0) |
| 8612 | 16.109992928 | 192.168.126.132 | 142.250.75.34 | QUIC | 75 | Protected Payload (KP0), DCID=eb91d8c1ab87111e |
| 8613 | 16.110160595 | 192.168.126.132 | 142.250.75.34 | QUIC | 75 | Protected Payload (KP0), DCID=eb91d8c1ab87111e |
| 8614 | 16.110511171 | 192.168.126.132 | 146.75.121.44 | TLSv1.3 | 126 | Application Data |
| 8615 | 16.110776421 | 146.75.121.44 | 192.168.126.132 | TCP | 60 | 443 → 35958 [ACK] Seq=4829 Ack=2833 Win=64240 Len |
| 8616 | 16.115177533 | 142.250.75.67 | 192.168.126.132 | QUIC | 66 | Protected Payload (KP0) |
| 8617 | 16.115177758 | 35.201.67.47 | 192.168.126.132 | QUIC | 66 | Protected Payload (KP0) |
| 8618 | 16.135685350 | 192.168.126.132 | 142.250.75.34 | QUIC | 74 | Protected Payload (KP0), DCID=eb91d8c1ab87111e |
| 8619 | 16.135846593 | 192.168.126.132 | 142.250.75.98 | QUIC | 74 | Protected Payload (KP0), DCID=f89a86409042e5cd |

```
Frame 1: 100 bytes on wire (800 bits), 100 bytes captured (800 bits) on interf
Ethernet II, Src: VMware_9e:c6:8d (00:0c:29:9e:c6:8d), Dst: VMware_ed:23:0c (6
Internet Protocol Version 4, Src: 192.168.126.132, Dst: 192.168.126.2
User Datagram Protocol, Src Port: 49502, Dst Port: 53
Domain Name System (query)
```

```
0000  00 50 56 ed 23 0c 00 0c  29 9e c6 8d 08 00
0010  00 56 19 5e 00 00 40 11  e3 61 c0 a8 7e 84
0020  7e 02 c1 5e 00 35 00 42  7e 2b 11 07 01 00
0030  00 00 00 00 00 01 0e 63  6c 69 65 6e 74 73
0040  76 69 63 65 73 0a 67 6f  6f 67 6c 65 61 70
0050  03 63 6f 6d 00 00 01 00  01 00 00 29 05 c0
0060  00 00 00 00
```

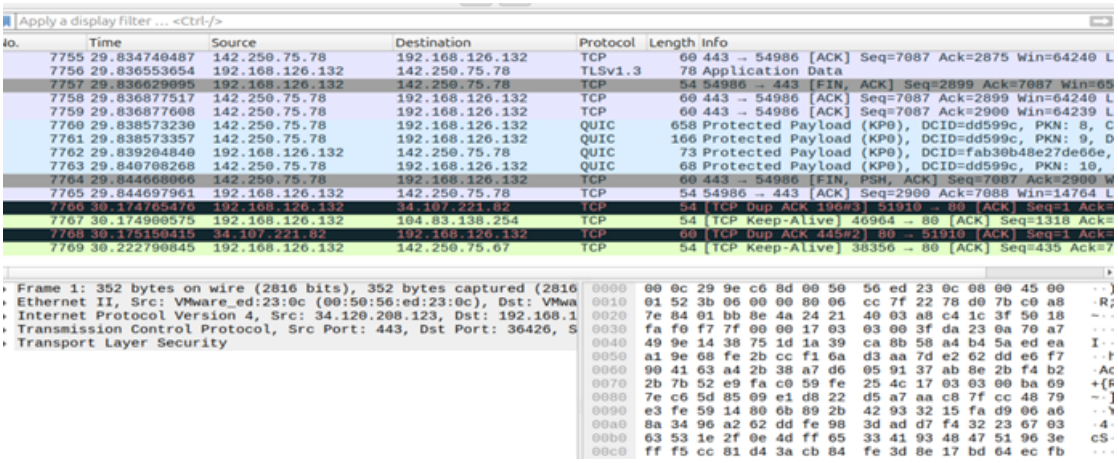



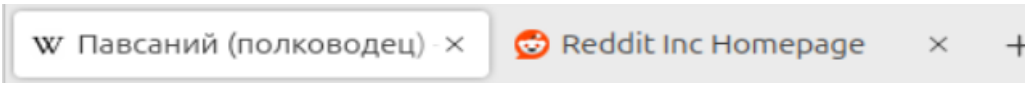
## הקלטה 2- Firefox Browsing

כמו שעשינו עם הגלישה בכרום, כך נעשה גם בדפדפן פיירפוקס באופן דומה.

נפתח הקלטה חדשה ב-Wireshark ונגלוש בפיירפוקס (ניכנס לסתם אתרים).

אחר כך נעצור את ההקלטה ונשמור אותה בשם קובץ pcapng.



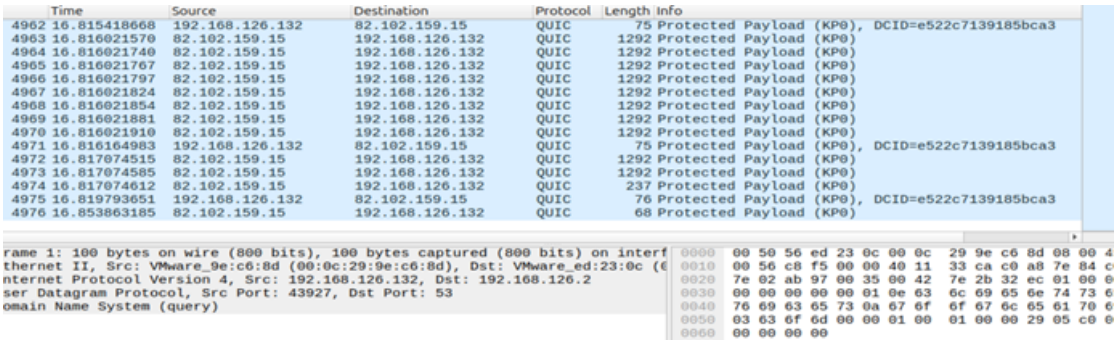




## הקלטה 3- Audio Streaming

אנחנו בחרנו להשתמש ב-YouTube Music בתור מקור ה- Audio Streaming שאותו נקליט.

נתחיל הקלטה חדשה ונגלוש ב-YouTube Music (נכנס ברנדומליות ל-שירים, בלי סרטונים רק אודיו)

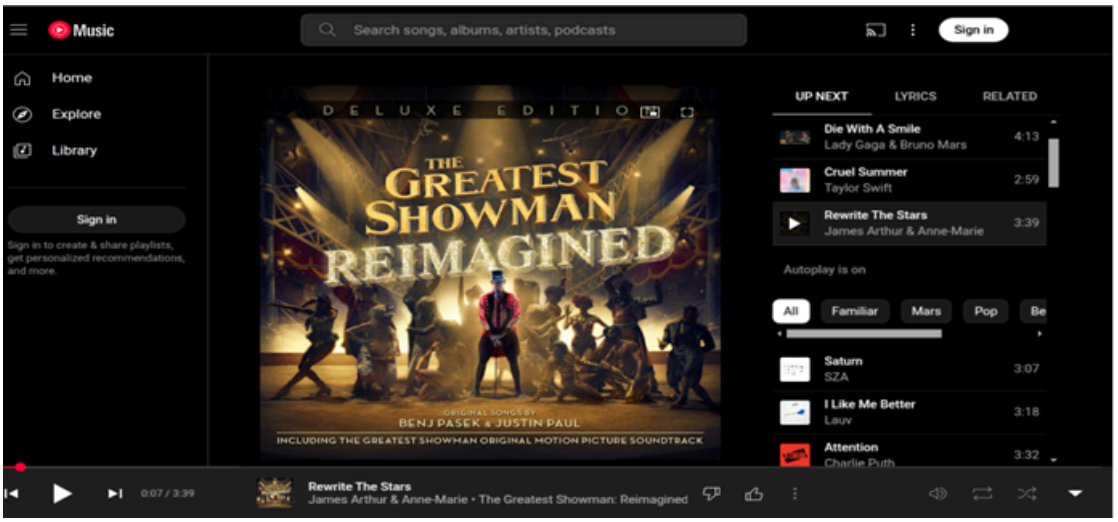
לאחר מכן נעצור את ההקלטה ונשמור אותה בתור קובץ pcapng.



| | Time | Source | Destination | Protocol | Length | Info |
|---|---|---|---|---|---|---|
| 4962 | 16.815418668 | 192.168.126.132 | 82.102.159.15 | QUIC | 75 | Protected Payload (KP0), DCID=e522c7139185bca3 |
| 4963 | 16.816021570 | 82.102.159.15 | 192.168.126.132 | QUIC | 1292 | Protected Payload (KP0) |
| 4964 | 16.816021740 | 82.102.159.15 | 192.168.126.132 | QUIC | 1292 | Protected Payload (KP0) |
| 4965 | 16.816021767 | 82.102.159.15 | 192.168.126.132 | QUIC | 1292 | Protected Payload (KP0) |
| 4966 | 16.816021797 | 82.102.159.15 | 192.168.126.132 | QUIC | 1292 | Protected Payload (KP0) |
| 4967 | 16.816021824 | 82.102.159.15 | 192.168.126.132 | QUIC | 1292 | Protected Payload (KP0) |
| 4968 | 16.816021854 | 82.102.159.15 | 192.168.126.132 | QUIC | 1292 | Protected Payload (KP0) |
| 4969 | 16.816021881 | 82.102.159.15 | 192.168.126.132 | QUIC | 1292 | Protected Payload (KP0) |
| 4970 | 16.816021910 | 82.102.159.15 | 192.168.126.132 | QUIC | 1292 | Protected Payload (KP0) |
| 4971 | 16.816164983 | 192.168.126.132 | 82.102.159.15 | QUIC | 75 | Protected Payload (KP0), DCID=e522c7139185bca3 |
| 4972 | 16.817074515 | 82.102.159.15 | 192.168.126.132 | QUIC | 1292 | Protected Payload (KP0) |
| 4973 | 16.817074585 | 82.102.159.15 | 192.168.126.132 | QUIC | 1292 | Protected Payload (KP0) |
| 4974 | 16.817074612 | 82.102.159.15 | 192.168.126.132 | QUIC | 237 | Protected Payload (KP0) |
| 4975 | 16.819793651 | 192.168.126.132 | 82.102.159.15 | QUIC | 76 | Protected Payload (KP0), DCID=e522c7139185bca3 |
| 4976 | 16.853863185 | 82.102.159.15 | 192.168.126.132 | QUIC | 68 | Protected Payload (KP0) |

```
rame 1: 100 bytes on wire (800 bits), 100 bytes captured (800 bits) on interf
thernet II, Src: VMware_9e:c6:8d (00:0c:29:9e:c6:8d), Dst: VMware_ed:23:0c (0
nternet Protocol Version 4, Src: 192.168.126.132, Dst: 192.168.126.2
ser Datagram Protocol, Src Port: 43927, Dst Port: 53
omain Name System (query)
```

```
0000  00 50 56 ed 23 0c 00 0c  29 9e c6 8d 08 00 4
0010  00 56 c8 f5 00 00 40 11  33 ca c0 a8 7e 84 c
0020  7e 02 ab 97 00 35 00 42  7e 2b 32 ec 01 00 0
0030  00 00 00 00 00 01 0e 63  6c 69 65 6e 74 73 6
0040  76 69 63 65 73 0a 67 6f  6f 67 6c 65 61 70 6
0050  03 63 6f 6d 00 00 01 00  01 00 00 29 05 c0 0
0060  00 00 00 00
```



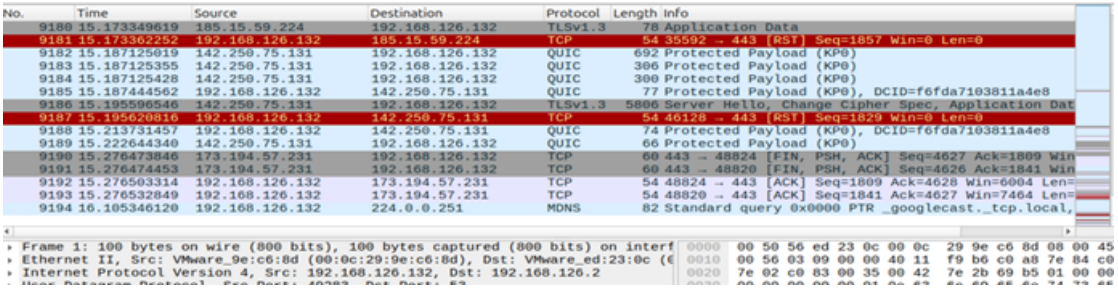

youtubeMusic_audioStreaming.pcapng

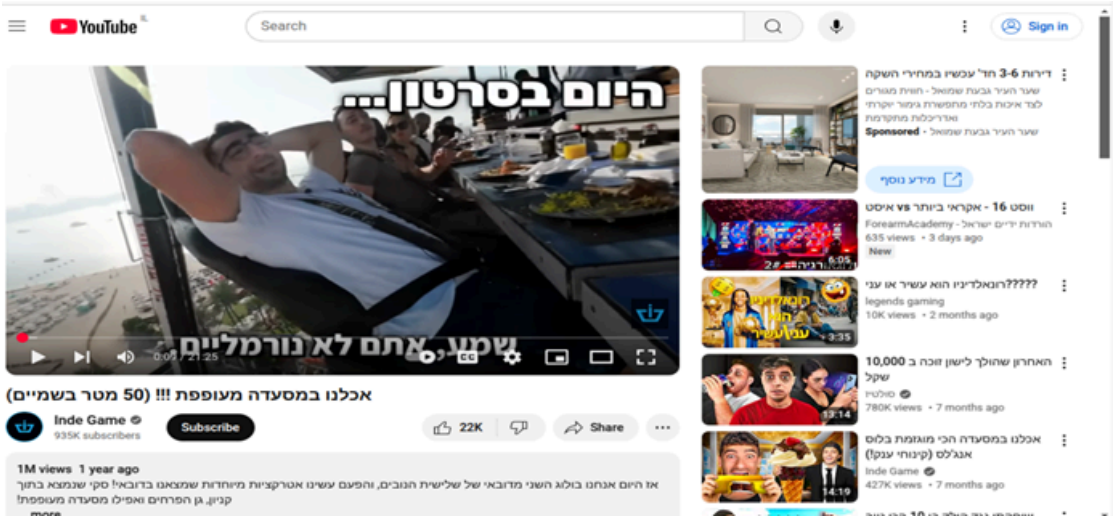
## הקלטה 4- Video Streaming

מקור ה-Video Streaming שלנו יהיה YouTube (הרגיל, לא YouTube Music).

נתחיל הקלטה חדשה ונתחיל לגלוש ב-YouTube ( ניכנס ל- סרטונים ברנדומליות).

לאחר מכן נעצור את ההקלטה ונשמור אותה בתור קובץ pcapng.



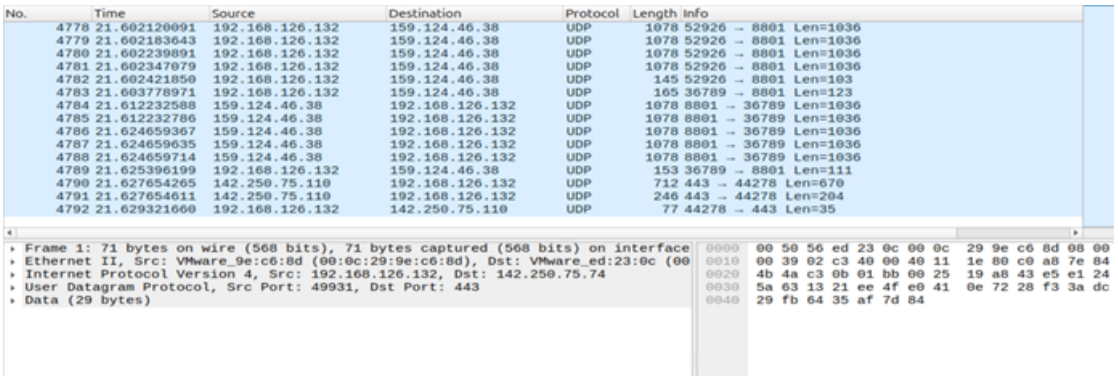

youtube_videoStreaming.pcapng

## הקלטה 5- Video Conferencing

מקור ה-Video Conferencing שלנו יהיה Zoom.

נתחיל הקלטה חדשה וניכנס לשיחת Zoom.

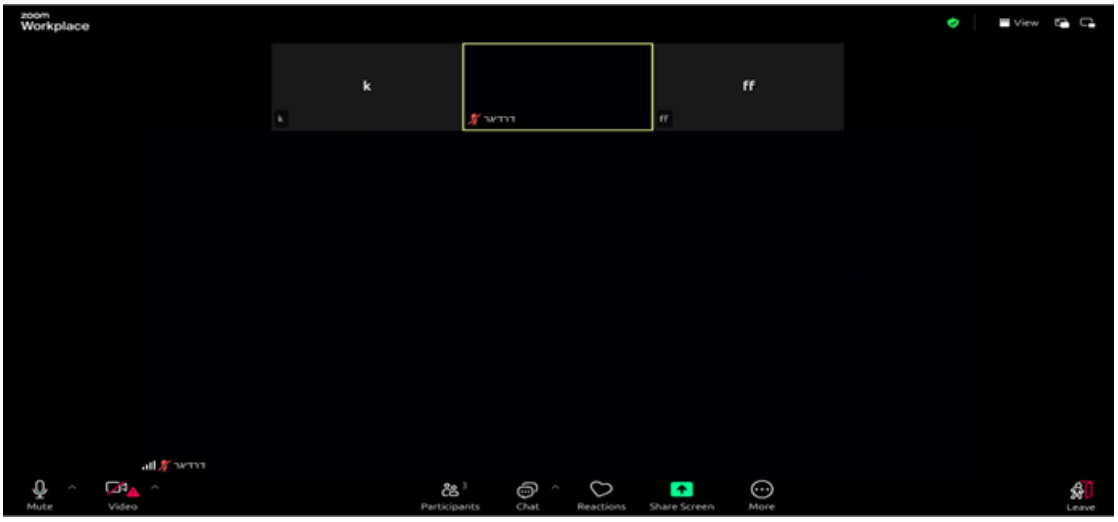
לאחר מכן נעצור את ההקלטה ונשמור אותה בתור קובץ pcapng.



| No. | Time | Source | Destination | Protocol | Length | Info |
|---|---|---|---|---|---|---|
| 4778 | 21.602120091 | 192.168.126.132 | 159.124.46.38 | UDP | 1078 | 52926 → 8801 Len=1036 |
| 4779 | 21.602183643 | 192.168.126.132 | 159.124.46.38 | UDP | 1078 | 52926 → 8801 Len=1036 |
| 4780 | 21.602239891 | 192.168.126.132 | 159.124.46.38 | UDP | 1078 | 52926 → 8801 Len=1036 |
| 4781 | 21.602347079 | 192.168.126.132 | 159.124.46.38 | UDP | 1078 | 52926 → 8801 Len=1036 |
| 4782 | 21.602421850 | 192.168.126.132 | 159.124.46.38 | UDP | 145 | 52926 → 8801 Len=103 |
| 4783 | 21.603778971 | 192.168.126.132 | 159.124.46.38 | UDP | 165 | 36789 → 8801 Len=123 |
| 4784 | 21.612232588 | 159.124.46.38 | 192.168.126.132 | UDP | 1078 | 8801 → 36789 Len=1036 |
| 4785 | 21.612232786 | 159.124.46.38 | 192.168.126.132 | UDP | 1078 | 8801 → 36789 Len=1036 |
| 4786 | 21.624659367 | 159.124.46.38 | 192.168.126.132 | UDP | 1078 | 8801 → 36789 Len=1036 |
| 4787 | 21.624659635 | 159.124.46.38 | 192.168.126.132 | UDP | 1078 | 8801 → 36789 Len=1036 |
| 4788 | 21.624659714 | 159.124.46.38 | 192.168.126.132 | UDP | 1078 | 8801 → 36789 Len=1036 |
| 4789 | 21.625396199 | 192.168.126.132 | 159.124.46.38 | UDP | 153 | 36789 → 8801 Len=111 |
| 4790 | 21.627654265 | 142.250.75.110 | 192.168.126.132 | UDP | 712 | 443 → 44278 Len=670 |
| 4791 | 21.627654611 | 142.250.75.110 | 192.168.126.132 | UDP | 246 | 443 → 44278 Len=204 |
| 4792 | 21.629321660 | 192.168.126.132 | 142.250.75.110 | UDP | 77 | 44278 → 443 Len=35 |

```
▸ Frame 1: 71 bytes on wire (568 bits), 71 bytes captured (568 bits) on interface
▸ Ethernet II, Src: VMware_9e:c6:8d (00:0c:29:9e:c6:8d), Dst: VMware_ed:23:0c (00
▸ Internet Protocol Version 4, Src: 192.168.126.132, Dst: 142.250.75.74
▸ User Datagram Protocol, Src Port: 49931, Dst Port: 443
▸ Data (29 bytes)

0000  00 50 56 ed 23 0c 00 0c  29 9e c6 8d 08 00
0010  00 39 02 c3 40 00 40 11  1e 80 c0 a8 7e 84
0020  4b 4a c3 0b 01 bb 00 25  19 a8 43 e5 e1 24
0030  5a 63 13 21 ee 4f e0 41  0e 72 28 f3 3a dc
0040  29 fb 64 35 af 7d 84
```




zoom_videoConferencing.pcapng

## 2,3. ניתוח הנתונים של ההקלטות עם גרפים + הסברים

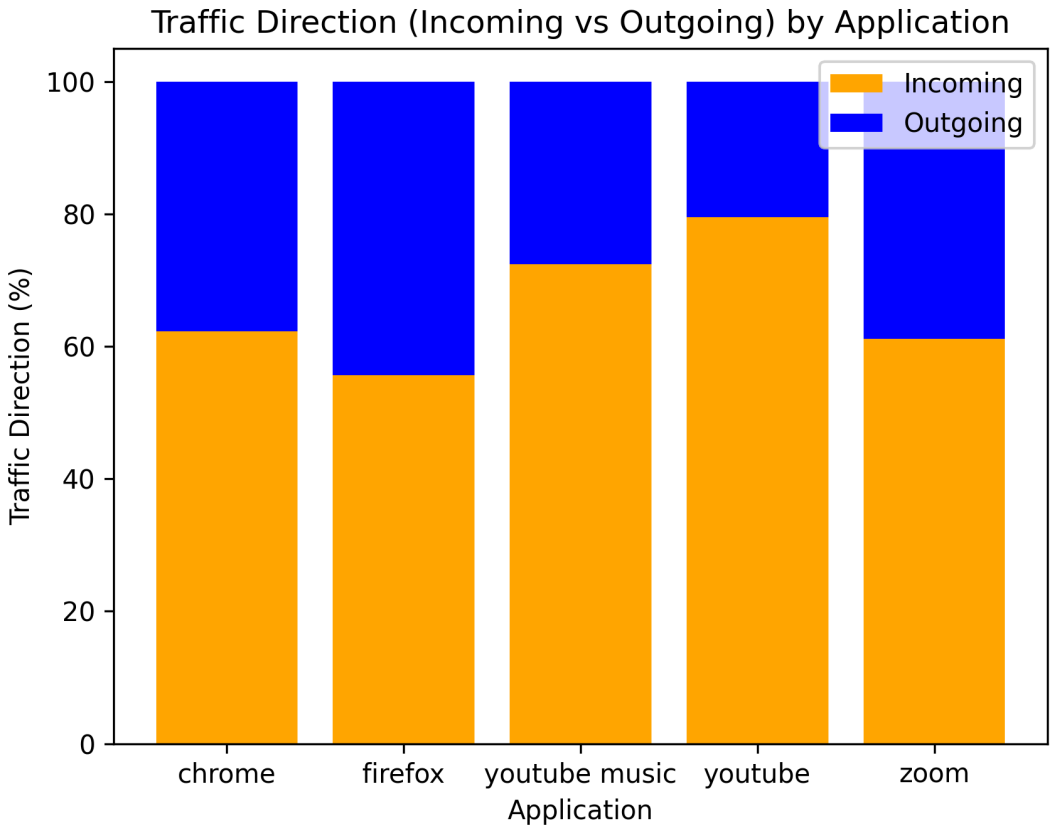
כל הגרפים נוצרו באמצעות סקריפטים של Python לצורך ניתוח נתוני התעבורה שהוקלטו בקובצי pcapng.

כתבנו סקריפטים ב-Python המשתמשים בספריות של פייתון לצורך ניתוח הנתונים והצגתם בצורה גרפית.

הסקריפטים מחשבים מדדים שונים כגון גודל חבילה ממוצע, זמן הגעה בין חבילות, יחס תעבורה נכנסת/יוצאת, פרוטוקולים בשימוש ועוד, ויוצרים גרפים המשווים בין האפליקציות שנבדקו.

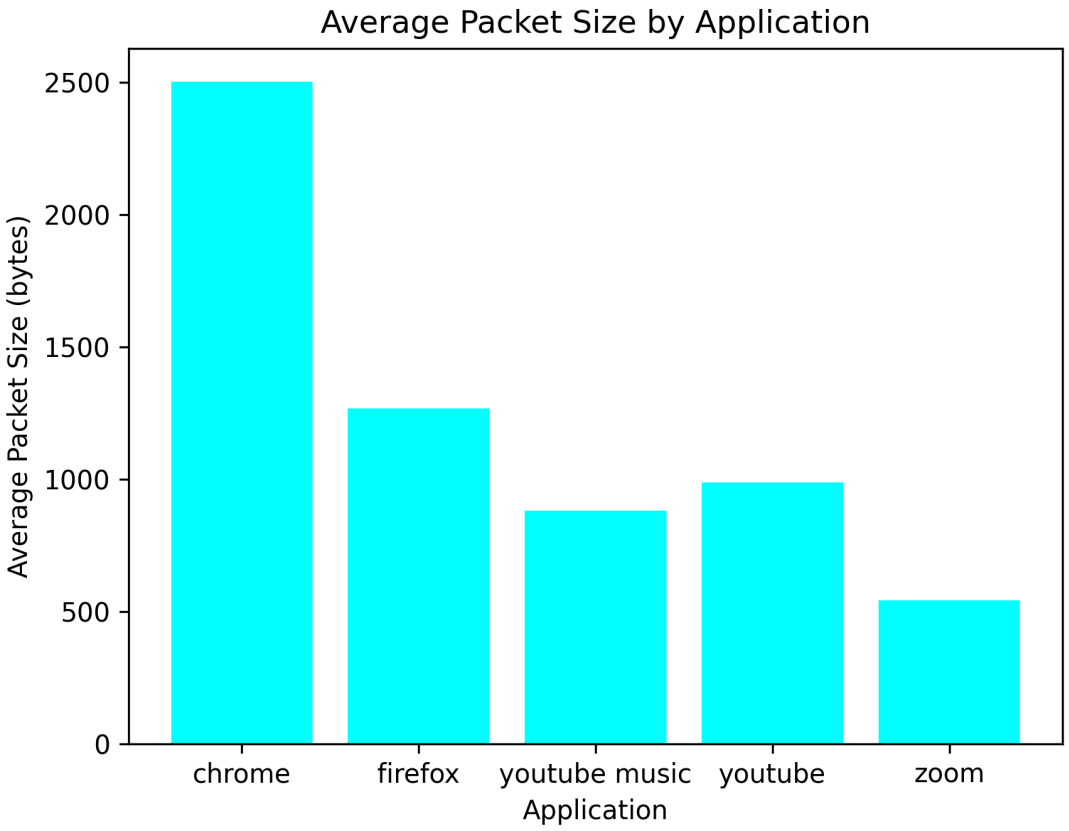
רשימת ספריות ה-Python שהשתמשנו בהן:

1. **pyshark**: קריאה וניתוח של קובצי pcapng, ושליפת נתוני חבילות: גודל, זמן הגעה, כתובת IP, TTL, סוג פרוטוקול, דגלי TCP וכו'.
2. **matplotlib**: יצירת גרפים ותרשימים להצגת הנתונים שנאספו מה-PCAP, הצגת הבדלים בין אפליקציות לפי מאפייני התעבורה שלהן.
3. **numpy**: חישובים מספריים על נתוני התעבורה. לדוגמה חישוב יחס תעבורה נכנסת/יוצאת, זמני הגעה ממוצעים בין חבילות ועוד.
4. **statistics**: חישובי סטטיסטיקה: ממוצעים, חציון, שונות, זמן הגעה ממוצע בין חבילות וכו'.
5. **collections.Counter**: ספירת הופעות של פרוטוקולים, דגלי TCP, פורטים ועוד.
6. **os**: יצירת ושמירת קובצי פלט (של הגרפים) בתיקיית פלט מוגדרת.
7. **pandas**: קריאה וניתוח של קובצי CSV עם נתוני תעבורה, וביצוע פילטור, ספירה והשוואה בין אפליקציות שונות.
8. **seaborn**: שיפור ויזואלי של הגרפים.
9. **re:** שליפת פורטי מקור ויעד מתוך עמודת ה-Info של ה-PCAP.



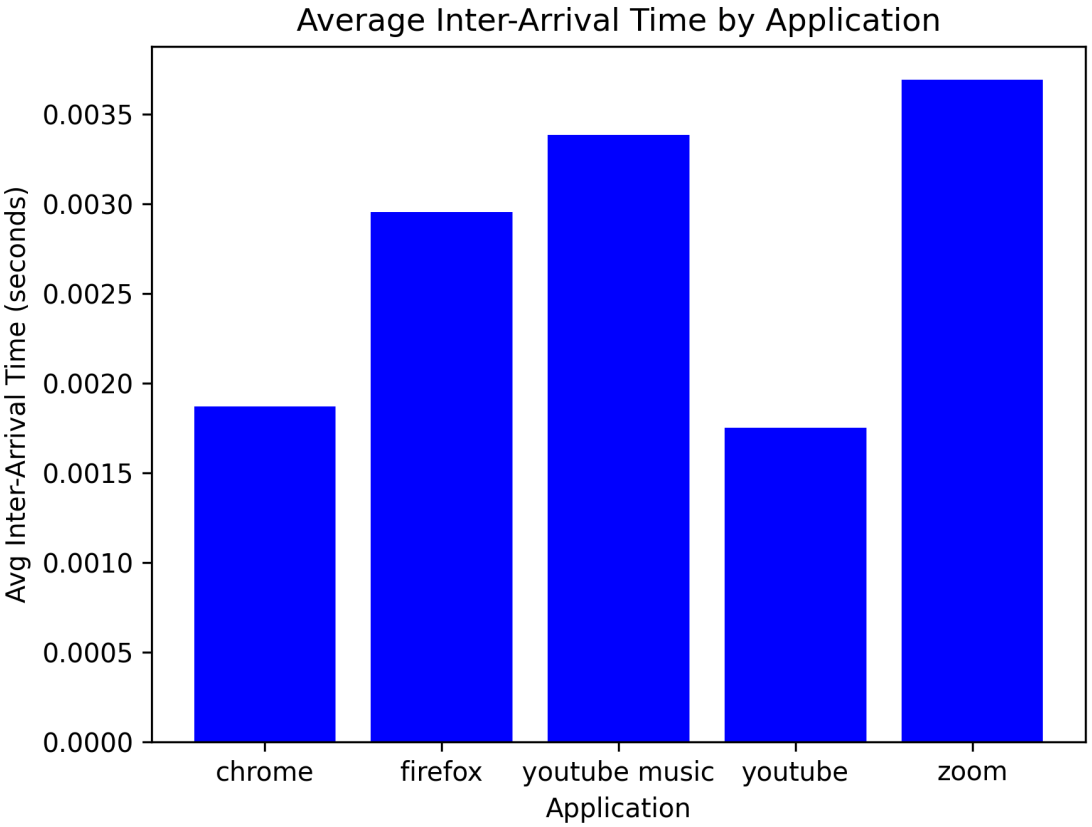

הגרף מציג את חלוקת התעבורה הנכנסת (Incoming) והיוצאת (Outgoing) עבור אפליקציות שונות.

- **Chrome:** רוב התעבורה נכנסת (~60%).
- **Firefox:** דומה ל-Chrome אך עם מעט פחות תעבורה נכנסת (~55%).
- **YouTube Music:** רוב מוחלט של התעבורה היא נכנסת (~70%).
- **YouTube:** התעבורה הנכנסת הגבוהה ביותר (~80%).
- **Zoom:** יחס מאוזן יותר (~60% נכנסת, ~40% יוצאת).



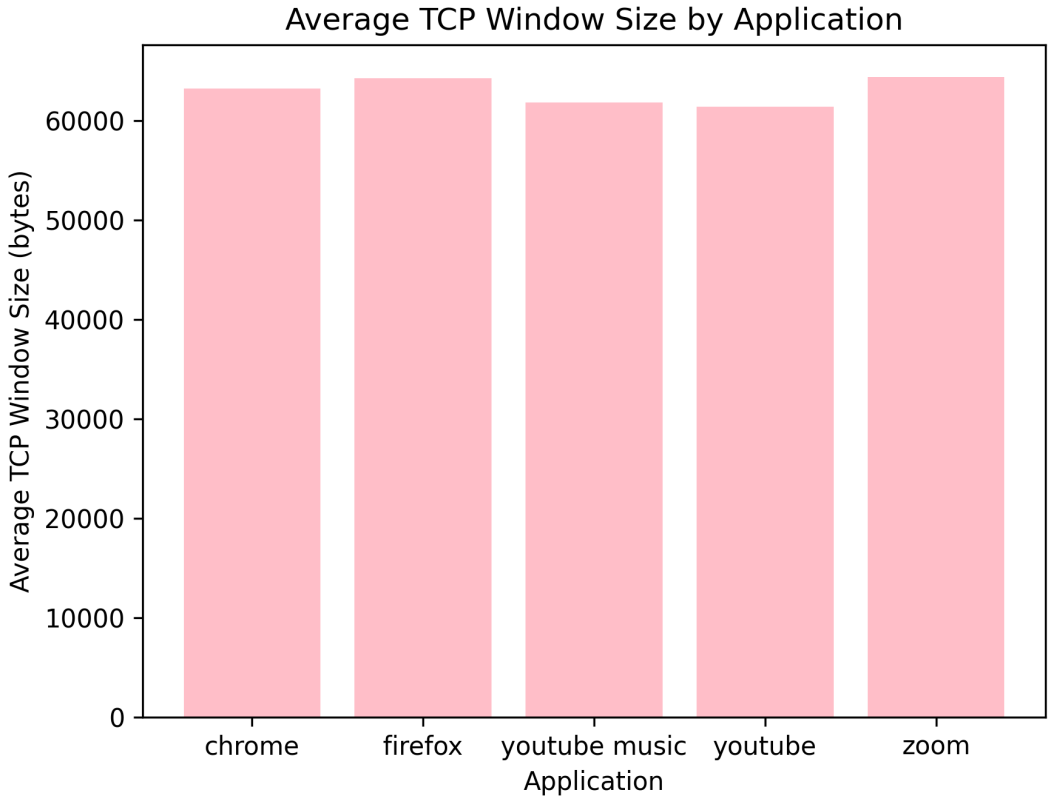

הגרף מציג את גודל החבילה הממוצע (Average Packet Size) בכל אחת מהאפליקציות.

- **Chrome:** גודל החבילה הממוצע הגבוה ביותר (~2500 בייט).
- **Firefox:** קטן משמעותית מ- Chrome (ב- ~1300 בייטים).
- **YouTube Music:** גודל חבילה קטן יחסית (~900 בייט).
- **YouTube:** חבילות מעט גדולות יותר (~1000 בייט) מאשר YouTube Music, אך עדיין די קטנות.
- **Zoom:** גודל החבילה הממוצע הקטן ביותר (~500 בייט).



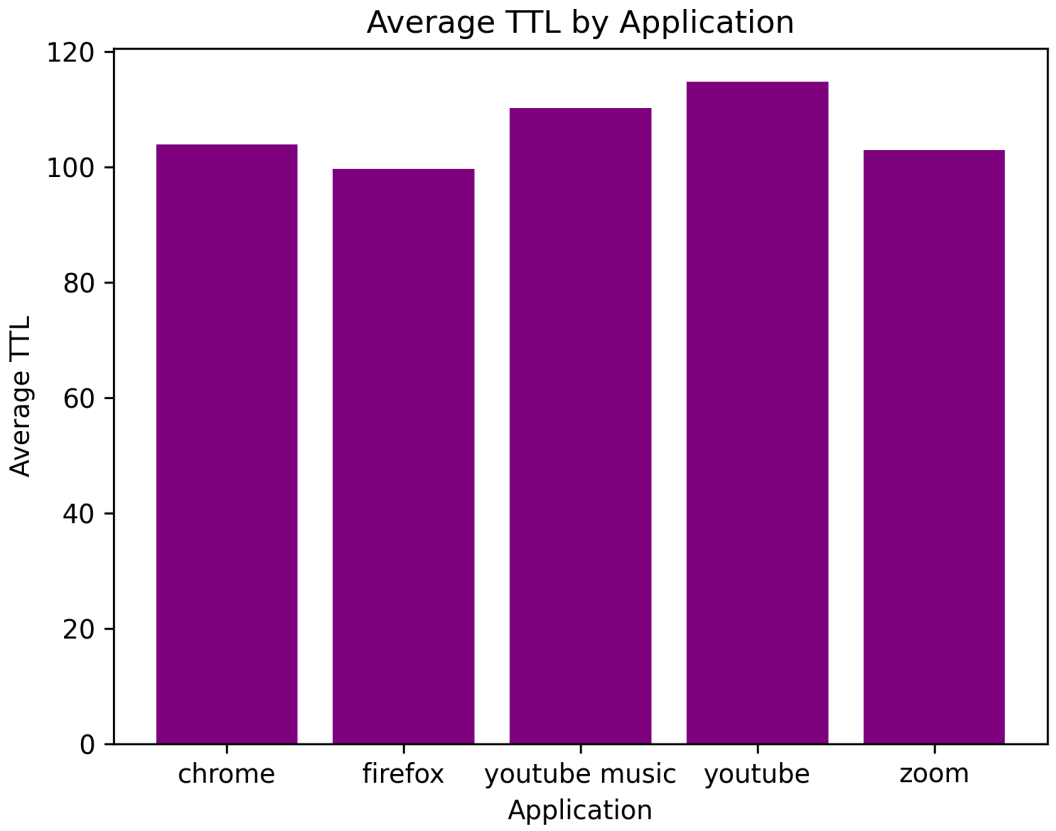

הגרף מציג את זמן ההגעה הממוצע בין חבילות (Average Inter Arrival Time) לכל אפליקציה.

- **Chrome:** זמן הגעה נמוך (~0.0018 שניות).
- **Firefox:** מעט גבוה יותר (~0.003 שניות).
- **YouTube Music:** זמן הגעה בינוני (~0.003 שניות).
- **YouTube:** נמוך יחסית (~0.002 שניות).
- **Zoom:** הזמן הגבוה ביותר (~0.0035 שניות).



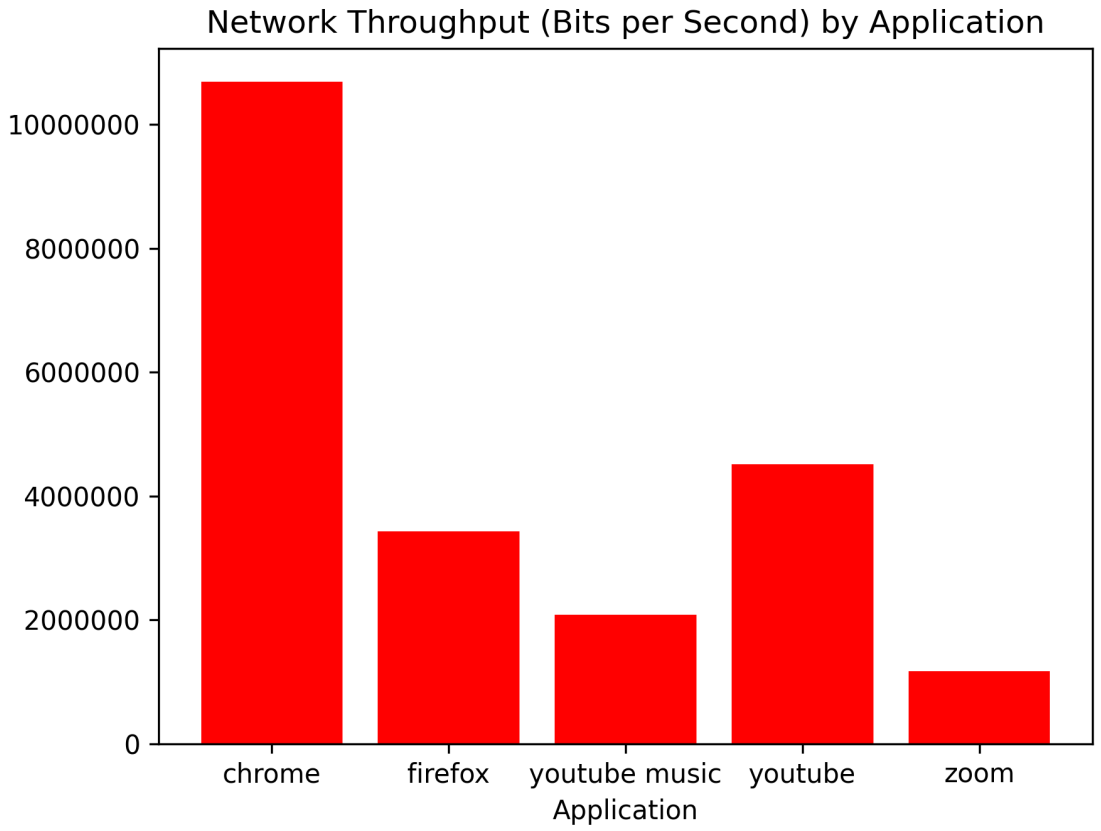

הגרף מציג את גודל חלון ה-TCP הממוצע (Average TCP Window Size) לכל אפליקציה.

- **Chrome:** גודל חלון TCP גבוה (~65,000 בייט).
- **Firefox:** מעט גבוה יותר מכרום (~67,000 בייט).
- **YouTube Music:** גודל חלון מעט נמוך יותר (~64,000 בייט).
- **YouTube:** דומה ליוטיוב מיוזיק (~63,000 בייט).
- **Zoom:** הגבוה ביותר (~68,000 בייט).



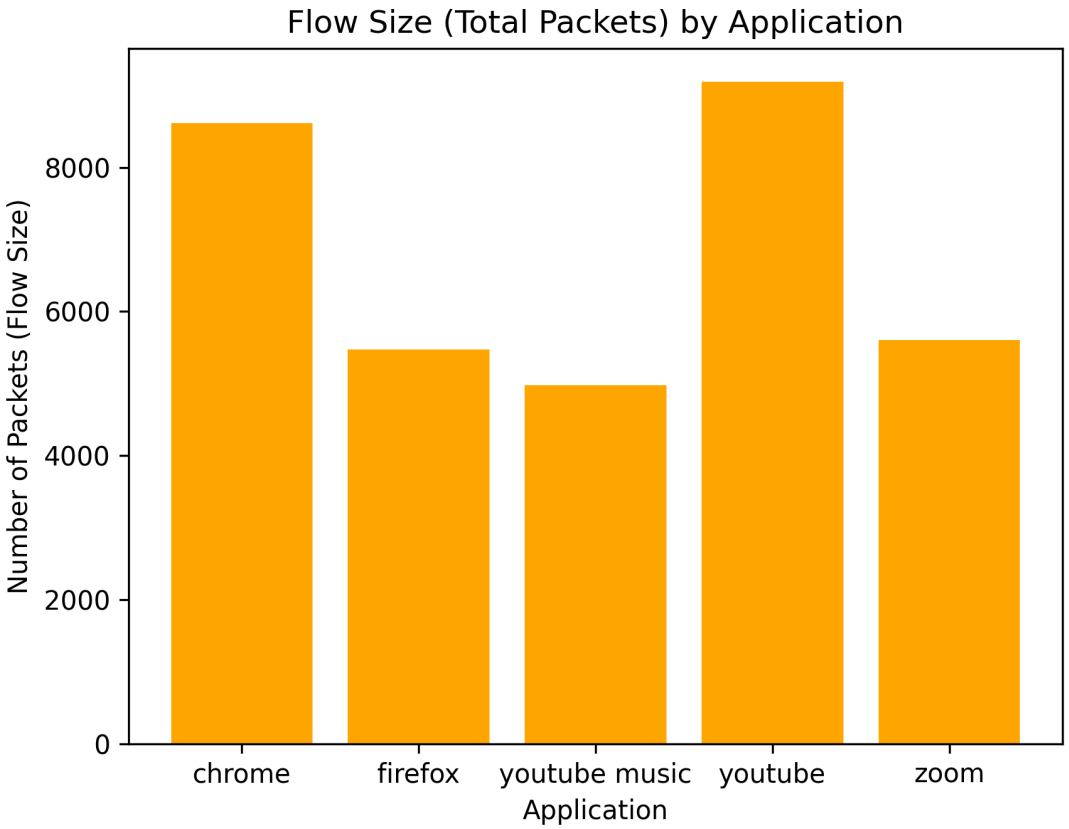

הגרף מציג את ה-TTL (ה- Time To Live) הממוצע לכל אפליקציה, שמייצג כמה ראוטרים (hops) יכולות החבילות לעבור לפני שהן נמחקות.

- **Chrome:** ה-TTL בממוצע ~105.
- **Firefox:** מעט נמוך יותר (~100).
- **YouTube Music:** ה-TTL גבוה יחסית (~110).
- **YouTube:** הגבוה ביותר (~115).
- **Zoom:** ה-TTL בינוני (~105).



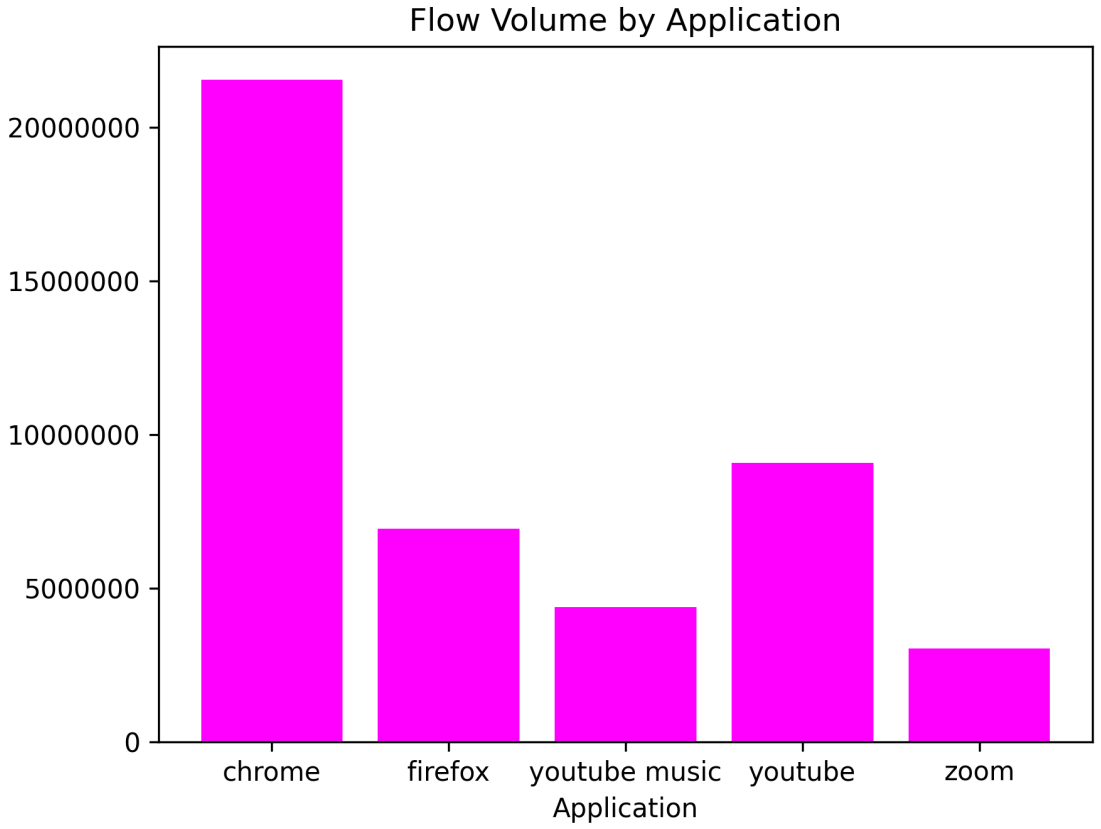

הגרף מציג את קצב העברת הנתונים (Throughput) ברשת לכל אפליקציה, נמדד בביטים לשנייה (bps).

- **Chrome:** קצב ההעברה הגבוה ביותר (~10,000,000 bps).
- **Firefox:** נמוך משמעותית (~3,500,000 bps).
- **YouTube Music:** קצב נמוך יחסית (~2,000,000 bps).
- **YouTube:** גבוה יותר (~5,000,000 bps).
- **Zoom:** הקצב הנמוך ביותר (~1,000,000 bps).



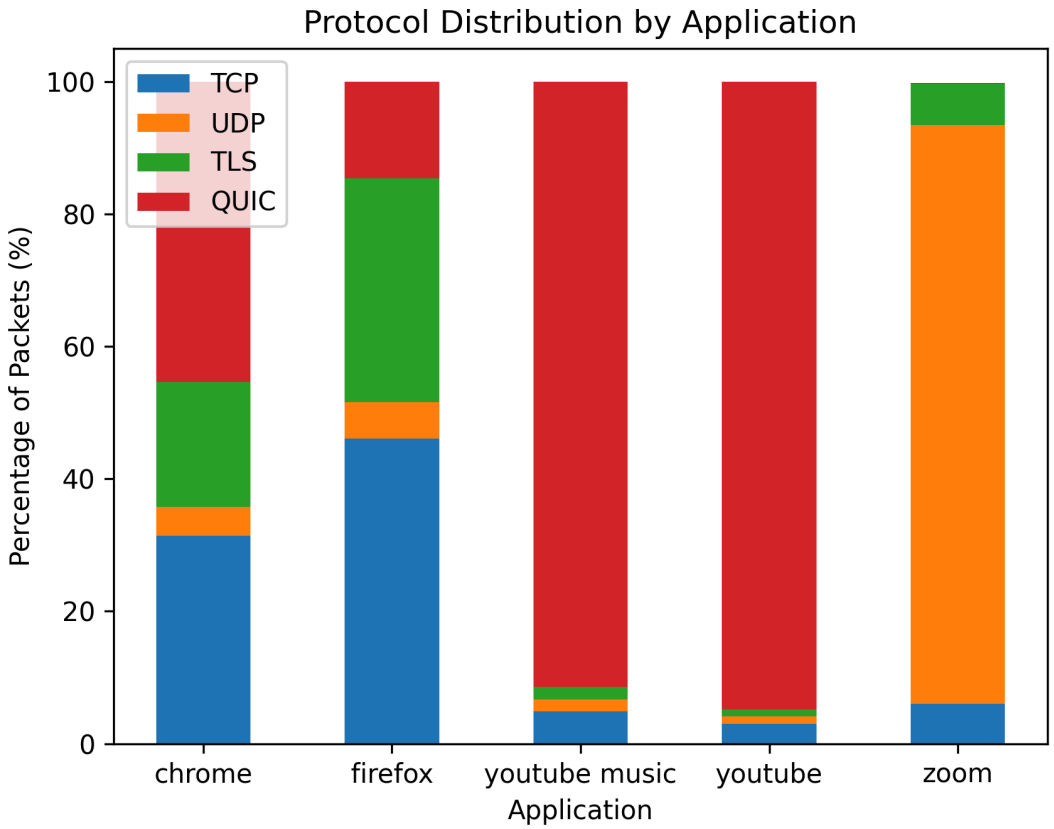

הגרף מציג את גודל הזרימה (Flow Size), כלומר מספר החבילות הכולל שנשלח עבור כל אפליקציה.

- **Chrome:** מספר חבילות גבוה (~9000).
- **Firefox:** נמוך יותר (~6000).
- **YouTube Music:** מספר חבילות קטן יחסית (~5000).
- **YouTube:** מספר החבילות הגבוה ביותר (~9500).
- **Zoom:** מספר חבילות בינוני (~6200).



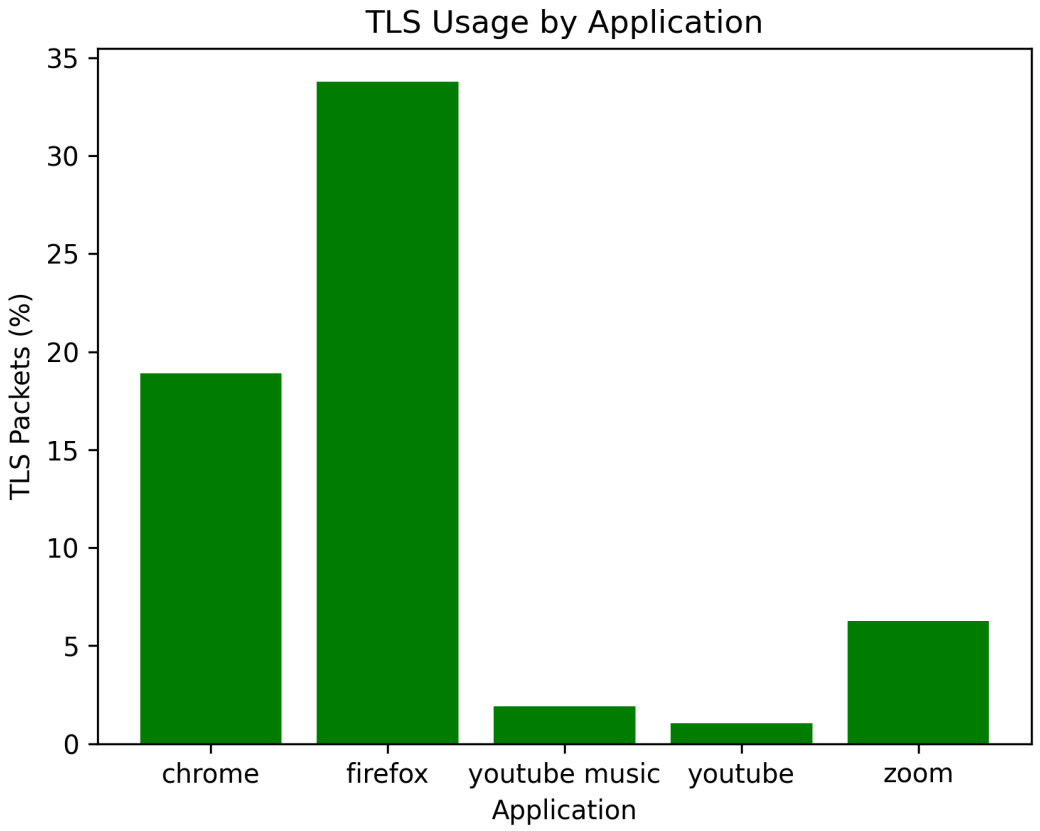

הגרף מציג את נפח הזרימה (Flow Volume), כלומר כמות הנתונים הכוללת שהועברה (בבייטים) לכל אפליקציה.

- **Chrome:** נפח הנתונים הגבוה ביותר (~21,000,000 בייט).
- **Firefox:** נמוך יותר (~7,000,000 בייט).
- **YouTube Music:** יחסית נמוך (~4,500,000 בייט).
- **YouTube:** נפח גבוה (~12,000,000 בייט).
- **Zoom:** הנפח הנמוך ביותר (~3,000,000 בייט).



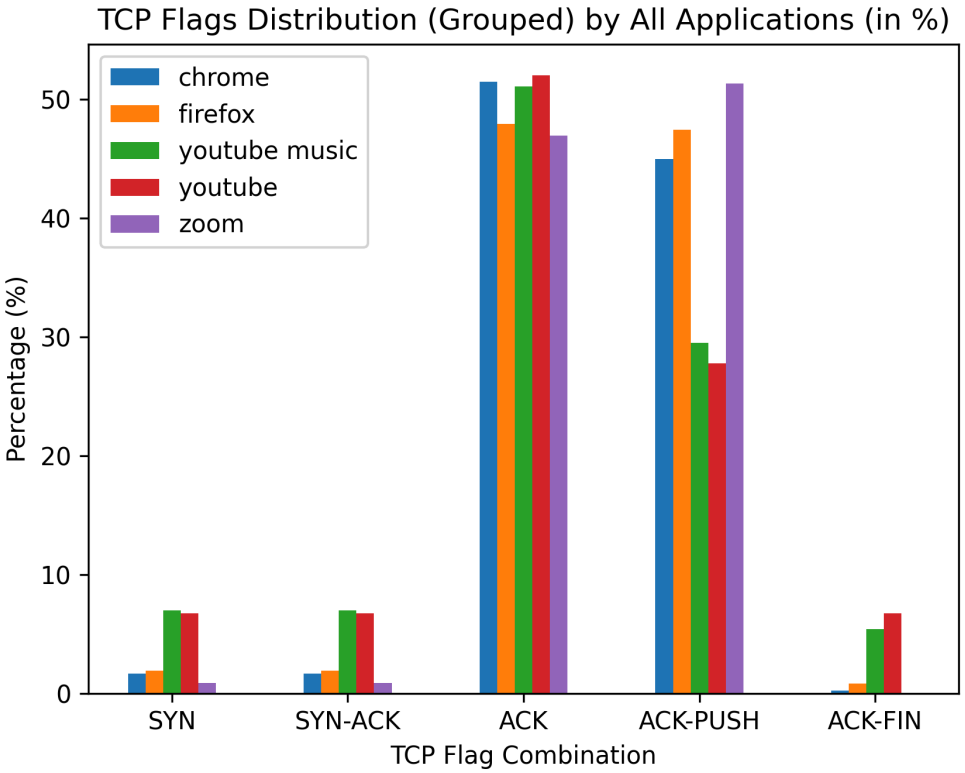

הגרף מציג את ההתפלגות (באחוזים) של פרוטוקולי התקשורת (TCP, UDP, TLS, QUIC) לכל אפליקציה.

- **Chrome:** רוב החבילות מתחלקות בין (כ-20%) TLS (~30%), TCP, ו- (~50%) QUIC.
- **Firefox:** הפרוטוקול העיקרי הוא (~50%) TCP, עם (~30%) TLS, ו- (~10%) UDP.
- **YouTube Music:** כמעט כולו QUIC (כ- 95%).
- **YouTube:** גם כאן QUIC שולט (~95%).
- **Zoom:** הדומיננטי הוא UDP (כ-85%), עם מעט (~10%) TCP ו-TLS.



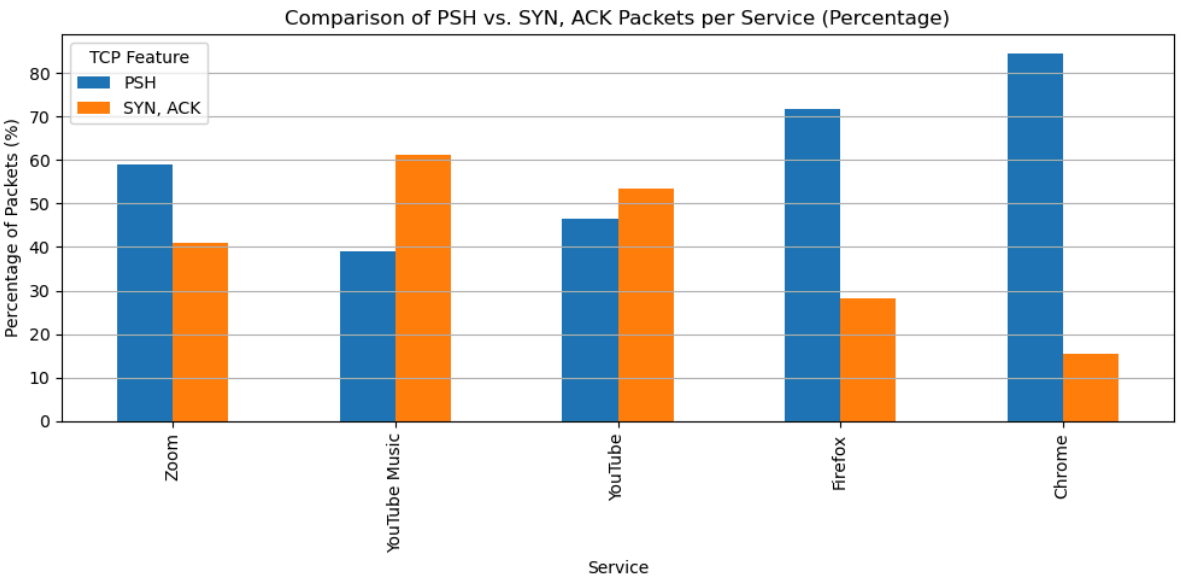

הגרף מציג את אחוז החבילות שמשתמשות ב-TLS בכל אפליקציה, כלומר כמות החבילות המוצפנות.

- **Chrome:** כ-20% מהחבילות משתמשות ב-TLS.
- **Firefox:** האחוז הגבוה ביותר (~34%).
- **YouTube Music:** שימוש נמוך מאוד (~2%).
- **YouTube:** כמעט אפסי (~1%).
- **Zoom:** אחוז נמוך (~6%).



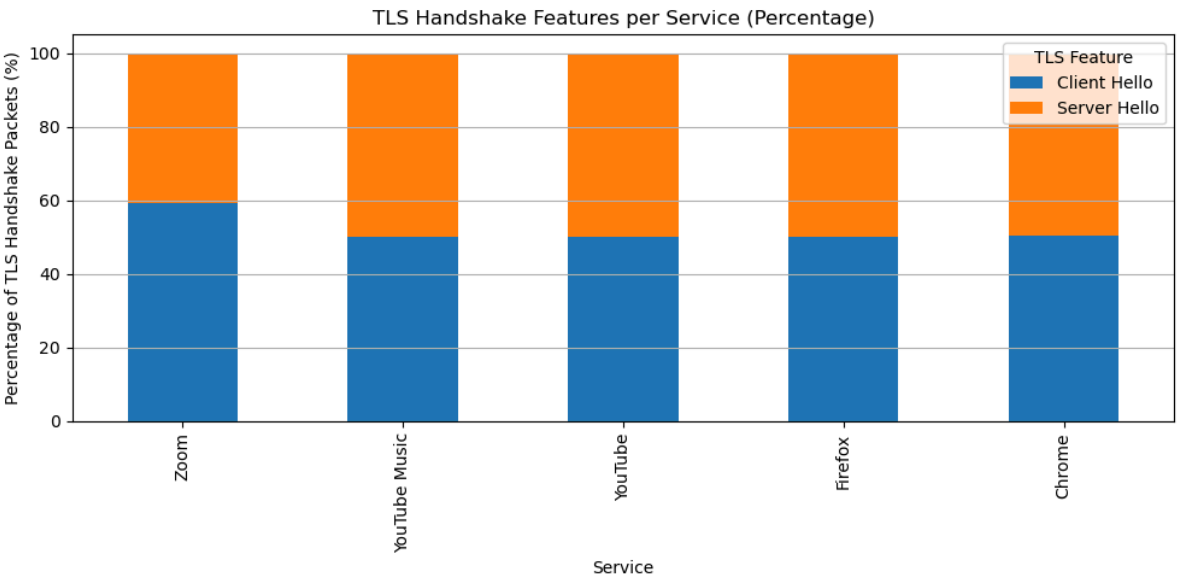

הגרף מציג את התפלגות דגלי ה-TCP בכל האפליקציות, כלומר איך כל אפליקציה משתמשת בסוגים שונים של חבילות TCP.

- **SYN:** אחוז נמוך בכל האפליקציות (~3-5%).
- **SYN-ACK:** גם הוא באחוזים נמוכים בכל האפליקציות (~2-4%).
- **ACK:** הדגל הנפוץ ביותר (~50-55% בכל האפליקציות).
- **ACK-PUSH:** גבוה מאוד (~40-50%), בעיקר ב-Zoom ובדפדפנים.
- **ACK-FIN:** אחוזים נמוכים בכל האפליקציות (~5-7%).



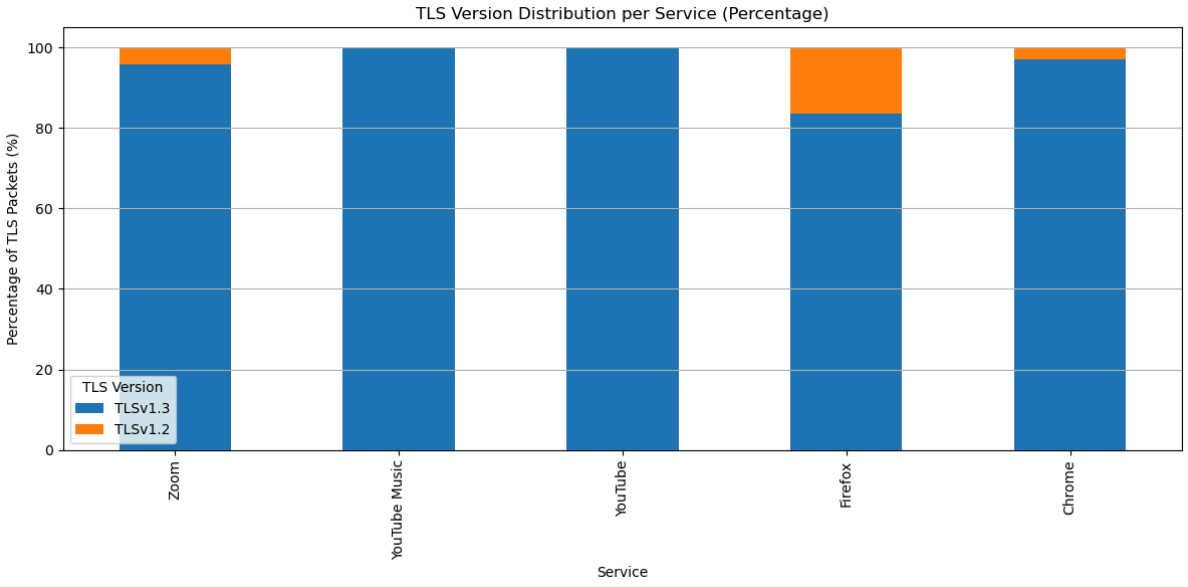

הגרף מציג את ההתפלגות האחוזית של PSH לעומת SYN/ACK.

- **Chrome:**
  חבילות PSH מגיעות לכ-85%, בעוד ש-SYN/ACK רק 15%.
- **Firefox:**
  כאן שיעור PSH הוא כ-70%. לעומתו, חבילות SYN/ACK מהוות כ-30%.
- **YouTube:**
  היחס נע בין 45% PSH ל-55% SYN/ACK.
- **YouTube Music:**
  חבילות SYN/ACK מהוות כ-60% לעומת 40% PSH.
- **Zoom:**
  כאן נקבל 60% PSH, לעומת 40% SYN/ACK.



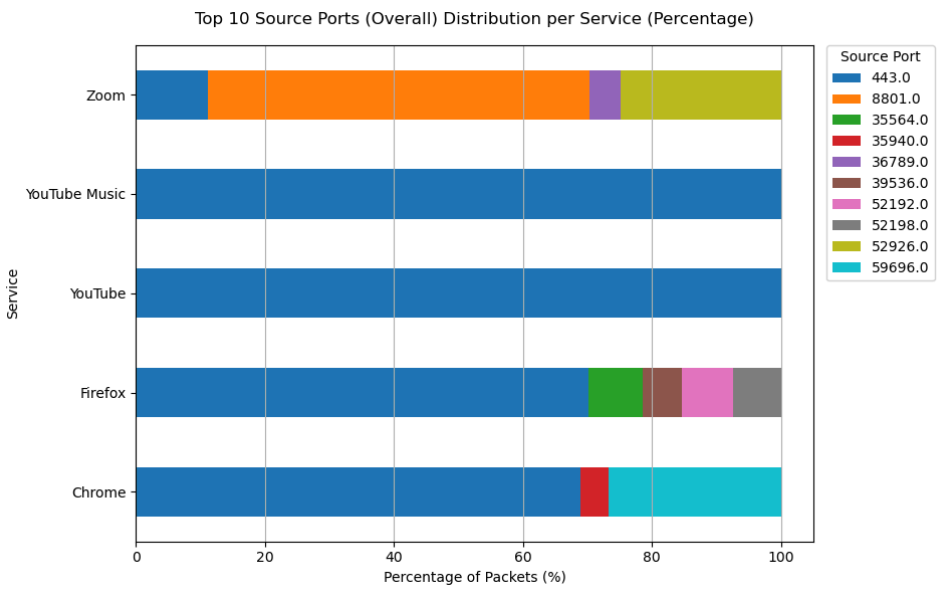

הגרף מציג את ההתפלגות האחוזית של הודעות Client Hello ו-Server Hello בכל שירות, חלק מה-tls handshake:

- **Zoom:** כ-60% Client Hello, לעומת 40% Server Hello.
- **YouTube Music:** כ-50% Client Hello, לעומת 50% Server Hello.
- **YouTube:** דומה ליוטיוב מיוזיק, 50% Client Hello, לעומת 50% Server Hello.
- **Firefox:** כ-52% Client Hello, לעומת 48% Server Hello
- **Chrome:** כ-53% Client Hello, לעומת 47% Server Hello.



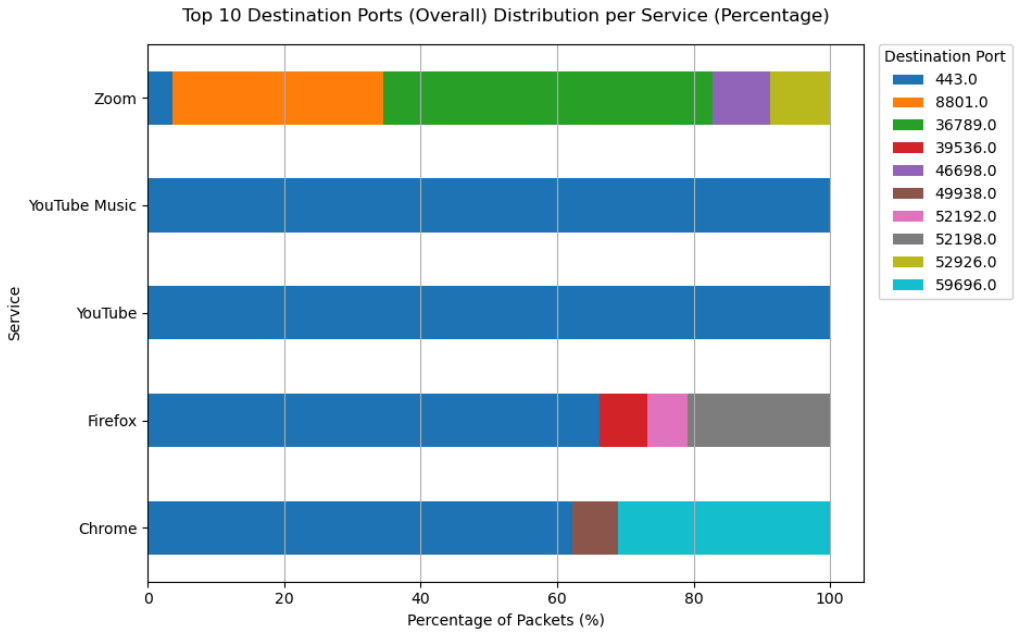

הגרף מציג את החלוקה האחוזית בין גרסאות TLSv1.3 ו-TLSv1.2 בכל שירות, וממחיש את ההעדפה לפרוטוקולים מאובטחים יותר:

- **Zoom:**
  100% מהתעבורה מבוססת על TLSv1.3.
- **YouTube Music:**
  גם כאן, 100% TLSv1.3.
- **YouTube:**
  תומך באופן מלא ב-TLSv1.3 (עם 100%).
- **Firefox:**
  כ-85% TLSv1.3 מול 15% TLSv1.2.
- **Chrome:**
  כ-95% TLSv1.3 לעומת 5% TLSv1.2.




הגרף מציג את התפלגות פורטי המקור (Source Ports) לכל שירות באחוזים.

- **Chrome:** רוב התעבורה מגיעה מפורט 443 (HTTPS), עם שימוש מועט בפורטים אקראיים כגון 35940 ו-59696.
- **Firefox:** גם כאן 443 דומיננטי, אך קיימת פעילות מסוימת בפורטים 52198, 52192 ו-39536.
- **YouTube Music:** יש 100% פורט 443.
- **YouTube:** מצב דומה ל-YouTube Music, כל התקשורת מגיעה מפורט 443.
- **Zoom:** מגוון רחב של פורטים - 443, 8801, 36789, 52926.




- **Chrome:** רוב מוחלט מהתעבורה עוברת דרך פורט 443 (HTTPS), עם מעט פעילות בפורטים אקראיים נוספים.
- **Firefox:** גם כאן פורט 443 דומיננטי, אך יש גם שימוש בפורט 52198 ובכמה פורטים אקראיים נוספים.
- **YouTube Music:** מאה אחוז מהתעבורה היא בפורט 443.
- **YouTube:** מצב דומה ל-YouTube Music, כל התעבורה בפורט 443.
- **Zoom:** מגוון גדול יותר של פורטים - 443, 8801, 36789, ופורטים נוספים.

# 4. דימוי תוקף

**דימוי תוקף בצורה ידנית- ע"י הנתונים שאספנו מהניתוח והגרפים שהפקנו מקודם.**

דימוי תוקף באופן ידני מתבצע על בסיס הנתונים שהופקו מהניתוח והגרפים שנבנו בשלבים הקודמים. תהליך זה מתבסס על אומדנים הנגזרים מהמדדים שהתקבלו, ומטרתו לזהות דפוסים אופייניים לכל סוג תעבורה בהתאם למאפיינים שנבדקו.

**חשוב לציין** כי הנתונים עשויים להשתנות בין הקלטות שונות עקב גורמים כגון משך ההקלטה, תנאי הסביבה (כמו רעשי רקע), עומס רשת משתנה ועוד. עם זאת, היחסים בין סוגי האפליקציות השונות צפויים להישמר, כך שניתן יהיה לזהות הבדלים עקביים בין סוגי תעבורה שונים (למשל, דפדפן לעומת וידאו סטרימינג) גם בתנאי בדיקה שונים.

המספרים המדויקים אליהם אנחנו מתייחסים בתהליך זה הם המספרים שהתקבלו מההקלטות והנתונים שאספנו בשלבים הקודמים. גם אם בהקלטות אחרות הערכים יהיו מעט שונים, השוני לא יהיה גדול מספיק כדי לערער את המסקנות או להרוס את התוצאות שהתקבלו. למעשה, השינויים בין ההקלטות אינם משמעותיים מספיק כדי לשנות את היחסים בין סוגי התעבורה, ולכן הניתוח שלנו נשאר יציב ומהימן גם במקרים של שונות קלה בנתונים.

תהליך זה מאפשר לבצע ניתוח השוואתי ולהסיק מסקנות מבוססות לגבי מאפייני התעבורה של כל סוג אפליקציה, תוך שמירה על עקביות בין הניסויים השונים.

## תרחיש 1: התוקף יודע את גודל החבילה, חותמת הזמן ואת ה-hash של ה-tuple-4

קיבלנו את הנתונים, כעת נראה איך ניתן לזהות את האפליקציה.

נתחיל מלעבור על נתוני ה-tuple:

1. **בודקים כתובת IP ופורט יעד (לפי ה-hash של ה-tuple):**
   - Google CDN (כלומר שירות של Google) ← כנראה YouTube / YouTube Music / Chrome
   - UDP בפורטים 8801, 36789, 3478-3481 ← כנראה Zoom.
   - מספר חיבורים קצרים למגוון אתרים בלי כתובת קבועה ← כנראה דפדפן (Chrome / Firefox).

אם יש כתובת IP ברורה של שירות ספציפי, ניתן לזהות במדויק. אם לא בטוחים, נמשיך לבדיקת הפרוטוקולים. בעזרת הנתונים מה-tuple ניתן להסיק באיזה פרוטוקול משתמשים, מה שיספק רמזים קריטיים על סוג האפליקציה בשימוש.

2. **בודקים את הפרוטוקול הדומיננטי:**
   - ~95% QUIC ← כנראה YouTube / YouTube Music / Chrome.
   - ~85% UDP ← כנראה Zoom.
   - ~80% TCP/TLS דומיננטי ← כנראה דפדפן (Firefox / Chrome).

כעת נבדוק את היחס בכיוון התעבורה:

3. **בודקים יחס תעבורה נכנסת/יוצאת:**
   - ~80% נכנסת, 20% יוצאת ← כנראה YouTube
   - ~70% נכנסת, 30% יוצאת ← כנראה YouTube Music
   - ~60% נכנסת, 40% יוצאת ← כנראה Chrome או Zoom.
   - ~55% נכנסת, 45% יוצאת ← כנראה Firefox.

אם יש יחס גבוה של תעבורה נכנסת, מדובר בסטרימינג (YouTube / YouTube Music). אם היחס יותר מאוזן, מדובר בדפדפן או שיחת וידאו (Zoom / Firefox / Chrome). כעת נבדוק את גודל החבילות בממוצע:

4. **בודקים גודל חבילות ממוצע:**
   - +~1300 בייטים ← כנראה דפדפן (Chrome / Firefox)
   - ~900-1000 בייטים ← כנראה סטרימינג (YouTube / YouTube Music)
   - ~500 בייטים ← כנראה Zoom

לבסוף, בעזרת חתימות הזמן של החבילות, ננתח לפי כמות זמן ממוצעת בין 2 חבילות:

5. **בודקים זמני הגעה בין חבילות (Inter-Arrival Time):**
   - ~0.0018 שניות ← כנראה Chrome.
   - ~0.002 שניות ← כנראה YouTube.
   - ~0.003 שניות ← כנראה YouTube Music או Firefox.
   - ~0.0035 שניות ← כנראה Zoom.

אם הזמן נמוך מאוד, כנראה מדובר בChrome, אם הזמן גבוה מאוד, כנראה מדובר בZoom. ואם הזמן בינוני-גבוה, מדובר בסטרימינג או Firefox.

### סיכום לתרחיש 1:

בתרחיש זה, התוקף מחזיק במידע רחב יחסית, הכולל את גודל החבילה, חותמת הזמן ואת ה-hash של ה-tuple-4, המאפשר לו להסיק מסקנות מדויקות למדי על סוג האפליקציה. תחילה, ניתן לבצע ניתוח של כתובות ה-IP והפורטים כדי לנסות לשייך את התעבורה לשירותים מוכרים כמו YouTube, Chrome או Zoom. לאחר מכן, בדיקה של הפרוטוקול הדומיננטי יכולה להצביע על כך שהתקשורת מתבצעת באמצעות QUIC, UDP או TCP, מה שמספק רמזים קריטיים על סוג השירות. יחס התעבורה הנכנסת מול היוצאת מוסיף מידע נוסף על אופיו של השימוש – כאשר סטרימינג מאופיין בתעבורה נכנסת דומיננטית, ואילו שירותים כמו דפדפנים ושיחות וידאו מציגים תעבורה מאוזנת יותר. בנוסף, גודל החבילות הממוצע וזמני ההגעה ביניהן מחזקים את התמונה, כאשר שירותי גלישה מתאפיינים בחבילות גדולות יחסית וזמני תגובה מהירים, סטרימינג בחבילות בגודל בינוני וזמנים ממוצעים, ושיחות וידאו בחבילות קטנות עם זמני הגעה ארוכים יותר. השילוב של כל הנתונים הללו מאפשר **זיהוי ברמת דיוק גבוהה מאוד**, ולעיתים אף זיהוי חד-משמעי של האפליקציה הספציפית.

### בתרחיש זה, איך ניתן למנוע את ההתקפה?

בתרחיש זה, התוקף מחזיק במידע רחב יחסית הכולל את גודל החבילה, חותמת הזמן ואת ה-hash של ה-tuple-4, ולכן יש להתמקד בהסתרת נתונים חיוניים כמו כתובת ה-IP, הפורטים והפרוטוקולים כדי למנוע זיהוי מדויק. דרכי ההגנה האפשריות כוללות:

- **הסתרת כתובת ה-IP:** שימוש ב-VPN, Proxy וכו' כדי למנוע מהתוקף לקשר את כתובת ה-IP לשירותים מסוימים כמו YouTube, Chrome או Zoom.
- **Padding של חבילות:** הוספת חבילות אקראיות או שינוי גודל החבילות כדי לטשטש דפוסים אופייניים לסוגי אפליקציות שונות, כך שהתוקף לא יוכל להסיק מסקנות מדויקות על האפליקציה בשימוש.
- **ערבוב תעבורה:** יצירת חיבורים מקבילים למספר שירותים במקביל (כגון סטרימינג בזמן גלישה) כדי לבלבל את התוקף ולמנוע ממנו לקבוע אילו חבילות קשורות לאיזו אפליקציה.
- **שינוי דינמי של פורטים:** שימוש בפורטים דינמיים או שרתים מתווכים שיקשו על זיהוי סוג האפליקציה על בסיס מספרי הפורטים הסטנדרטיים.
- **שימוש בפרוטוקולים מאובטחים ומוצפנים:** מעבר לשימוש בפרוטוקולים מוצפנים שיטשטשו את סוג התעבורה וימנעו מהתוקף לזהות את הפרוטוקולים המקוריים.
- **יצירת תעבורה רנדומלית:** שליחת חבילות "ריקות" או שינויים אקראיים במבנה התעבורה כדי לטשטש את היחס בין תעבורה נכנסת ויוצאת, ולהקשות על הסקת מסקנות מדויקות.

## תרחיש 2: התוקף יודע רק את גודל החבילה ואת חותמת הזמן

גם פה, קיבלנו את הנתונים, כעת נראה איך ניתן לזהות את האפליקציה.

נתחיל מלבדוק גודל חבילות ממוצע:

1. **בודקים גודל חבילות ממוצע:**
   - ~1300+ בייטים ← כנראה דפדפן (Chrome / Firefox)
   - ~900-1000 בייטים ← כנראה סטרימינג (YouTube / YouTube Music)
   - ~500 בייטים ← כנראה Zoom

לכן אנחנו מסיקים שאם החבילות קטנות מאוד בממוצע, כנראה מדובר בזום. אם החבילות בינוניות, כנראה מדובר בסטרימינג, ואם החבילות גדולות או גדולות מאוד, כנראה מדובר בדפדפן.

כעת בעזרת חתימות הזמן של החבילות, ננתח לפי כמות זמן ממוצעת בין כל 2 חבילות:

2. **בודקים זמני הגעה בין חבילות (Inter-Arrival Time):**
   - ~0.0018 שניות ← כנראה Chrome.
   - ~0.002 שניות ← כנראה YouTube.
   - ~0.003 שניות ← כנראה YouTube Music או Firefox.
   - ~0.0035 שניות ← כנראה Zoom.

אם הזמן נמוך מאוד, כנראה מדובר בChrome, אם הזמן גבוה מאוד, כנראה מדובר בZoom. ואם הזמן בינוני-גבוה, מדובר בסטרימינג או Firefox.

### סיכום לתרחיש 2:

בתרחיש זה, המידע שבידי התוקף מוגבל יותר, שכן הוא יודע רק את גודל החבילה ואת חותמת הזמן, ללא מידע על היעד, הפורטים או הפרוטוקולים בשימוש. כתוצאה מכך, הזיהוי הופך לפחות מדויק, אך עדיין ניתן לבצע סיווג כללי של סוג השירות. גודל החבילות מספק הבחנה ראשונית בין סוגי אפליקציות, כאשר חבילות קטנות אופייניות לשיחות וידאו, חבילות בגודל בינוני מתאימות לסטרימינג, וחבילות גדולות מצביעות על גלישה בדפדפן. זמני ההגעה בין החבילות מסייעים להבחנה נוספת, כאשר שירותים שונים מציגים דפוסים ייחודיים של זמני תגובה. עם זאת, קיימת חפיפה בין חלק מהשירותים, מה שמקשה על קביעת זיהוי חד-משמעי ומוביל לבלבול בין סוגי תעבורה מסוימים, כמו סטרימינג ודפדפנים.

### בתרחיש זה, איך ניתן למנוע את ההתקפה?

בתרחיש זה, שבו התוקף יודע רק את גודל החבילות ואת חותמת הזמן, יכולת הזיהוי שלו פחות מדויקת מאשר בתרחיש הראשון. עם זאת, עדיין ניתן לטשטש את דפוסי התעבורה כדי להקשות עליו להסיק מסקנות ברורות. דרכי ההגנה האפשריות כוללות:

- **הרצת מספר אפליקציות במקביל:** הפעלת מספר שירותים בו-זמנית (כגון השמעת מוזיקה תוך כדי גלישה) כדי ליצור תעבורה מעורבת ולמנוע זיהוי חד-משמעי של סוג השירות.
- **יצירת תעבורה מזויפת:** שליחת חבילות אקראיות בזמן שאינו קשור לפעילות בפועל, כך שהתוקף לא יוכל לזהות דפוסים מבוססי גודל חבילה או זמני הגעה.
- **שימוש בפרוטוקולים עם תעבורה אחידה:** מעבר לשימוש בפרוטוקולים שמצפינים ומאחדים את גודל החבילות והמרווחים ביניהן, כמו VPN, כדי למנוע הבחנה בין סוגי התעבורה.

## סיכום לשני התרחישים:

ניתוח שני התרחישים מצביע על כך שכמות המידע הזמינה לתוקף היא הגורם הקריטי בדיוק הזיהוי. כאשר התוקף מחזיק במידע רחב הכולל נתוני tuple-4, הפרוטוקול וגודל החבילה, ניתן לזהות אפליקציות בדיוק גבוה ואף לקשר תעבורה לשירותים ספציפיים. לעומת זאת, כאשר המידע מוגבל רק לגודל החבילות ולזמני ההגעה, הזיהוי הופך למעורפל יותר, אך עדיין ניתן לבצע סיווג כללי לסוג השירות. באופן כללי, אפליקציות סטרימינג, גלישה ושיחות וידאו מציגות דפוסי תעבורה שונים מספיק כדי שניתן יהיה להבדיל ביניהן, אך כאשר המידע חלקי, הגבולות בין הסוגים הללו מטושטשים יותר.

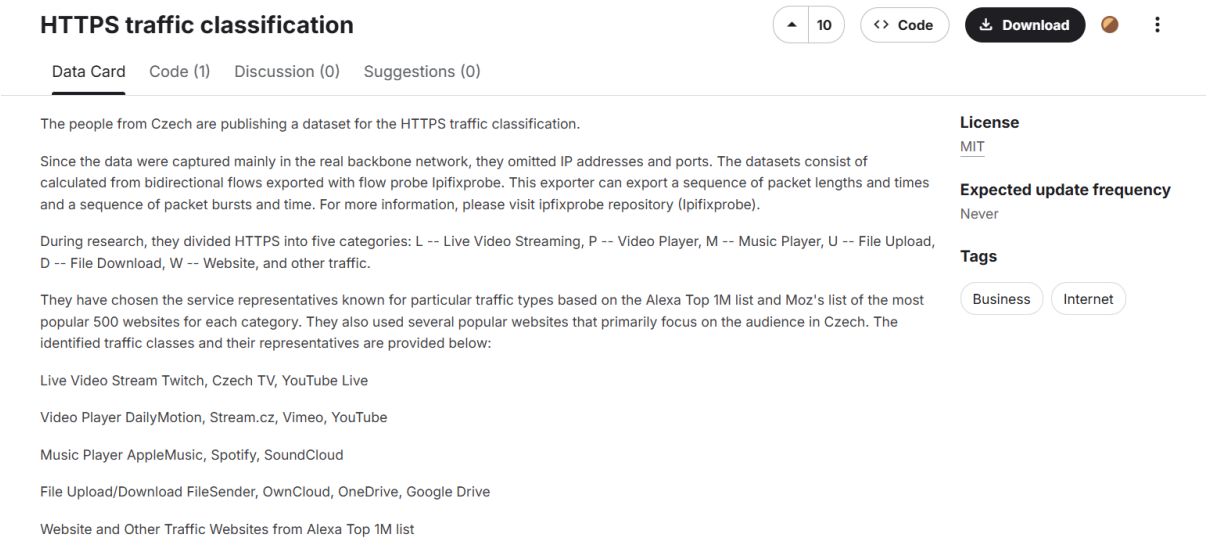
## דימוי תוקף בצורה אוטומטית באמצעות למידת מכונה

כיוון בתרחיש השני זיהוי התעבורה ידנית קשה יותר משמעותית, רצינו לבדוק האם נוכל לקבל תוצאות סיווג טובות יותר אם נשתמש במודל למידת מכונה.
ולכן, יצרנו מודל למידת מכונה המסוגל לסווג תעבורת רשת לפי סוג השירות בו השתמש המשתמש.
המטרה היא שוב, לדמות תוקף שמנסה לזהות את סוג האפליקציה שמשתמשים בה, על סמך גודל החבילות וחותמות הזמן בלבד, מבלי לראות את תוכן התעבורה עצמו.

## הקדמה:

השתמשנו בדאטה סט מהאתר Kaggle (אושר ע"י המרצה בפורום).

קישור לדאטה סט: https://www.kaggle.com/datasets/inhngcn/https-traffic-classification/data



HTTPS traffic classification

10 | Code | Download

Data Card | Code (1) | Discussion (0) | Suggestions (0)

The people from Czech are publishing a dataset for the HTTPS traffic classification.

Since the data were captured mainly in the real backbone network, they omitted IP addresses and ports. The datasets consist of calculated from bidirectional flows exported with flow probe Ipifixprobe. This exporter can export a sequence of packet lengths and times and a sequence of packet bursts and time. For more information, please visit ipfixprobe repository (Ipifixprobe).

During research, they divided HTTPS into five categories: L -- Live Video Streaming, P -- Video Player, M -- Music Player, U -- File Upload, D -- File Download, W -- Website, and other traffic.

They have chosen the service representatives known for particular traffic types based on the Alexa Top 1M list and Moz's list of the most popular 500 websites for each category. They also used several popular websites that primarily focus on the audience in Czech. The identified traffic classes and their representatives are provided below:

Live Video Stream Twitch, Czech TV, YouTube Live

Video Player DailyMotion, Stream.cz, Vimeo, YouTube

Music Player AppleMusic, Spotify, SoundCloud

File Upload/Download FileSender, OwnCloud, OneDrive, Google Drive

Website and Other Traffic Websites from Alexa Top 1M list

License
MIT

Expected update frequency
Never

Tags
Business | Internet

הדאטה סט מכיל תעבורה מוקלטת של מספר שירותים שונים.

שלב ראשון היה למחוק את השירותים שלא רלוונטיים לנו- File Upload ו- File Download.

אחרי זה שינינו את עמודת type שלא תכיל קיצורים כמו P בשביל Video Streaming או M בשביל Audio Streaming אלא את השמות Video Streaming ו- Audio Streaming עצמם כדי להקל על קריאת הדאטה.

בנוסף, הכמות מכל שירות לא הייתה שווה, מה שהיה יכול ליצור bias (הטייה) בהליך הסיווג של המודל לשירותים השונים. לכן דאגנו למחוק רשומות מכל השירותים כדי שהכמות שלהם תהיה שווה, ובכך להבטיח את אמינות הסיווג.

# בניית המודלים

כתבנו סקריפט בפייתון שקורא את הדאטה סט ומעבד את הנתונים. לאחר מכן יוצר מודלים ומאמן אותם על הדאטה, ולבסוף משתמש במודלים אלה לחיזוי השירות בו נעשה שימוש + בדיקת רמת הדיוק.

ספריות הפייתון בהן השתמשנו בקוד:

**עיבוד נתונים**

- **pandas** - קריאה של קובץ ה-CSV, עיבוד הנתונים וטיפול בטבלאות.
- **numpy** - ביצוע חישובים מספריים וניהול מערכים נומריים.

**למידת מכונה**

- **sklearn** - ספרייה ללמידת מכונה המכילה:
  - **חלוקת נתונים** - train_test_split מחלק את הדאטה לסט אימון וסט בדיקה.
  - **נרמול הנתונים** - StandardScaler מבצע נרמול (סטנדרטיזציה) של הנתונים.
  - **חישוב דיוק** - accuracy_score מחשב את הדיוק של המודלים.
  - **יצירת מודלים** - LogisticRegression, SVM, ו- RandomForestClassifier.
- **imblearn** - איזון מחודש של הנתונים באמצעות SMOTE, כדי להתמודד עם חוסר איזון בין הקטגוריות.
- **xgboost** - שימוש במודל XGBClassifier, שהוא אלגוריתם חזק לסיווג נתונים.

# תהליך האימון והחיזוי

1. **טעינת הדאטה סט**
   - קריאת הקובץ והסרת עמודות שאינן רלוונטיות.
2. **הכנת הנתונים**
   - בחירת **תכונות רלוונטיות בלבד** (כלומר רק העמודות המתייחסות לגודל פאקטה והזמן שלה)
   - קידוד המשתנה TYPE (סוג השירות) לערכים מספריים.
   - נרמול הנתונים כדי לוודא שסקאלת המספרים לא תשפיע על הביצועים.
   - איזון הנתונים באמצעות SMOTE (בנוסף למחיקה הידנית שעשינו)
3. **אימון המודלים**
   - אימון ארבעה מודלים שונים על סט אימון (20% מהדאטה):
     - **Logistic Regression**
     - **Support Vector Machine (או SVM)**
     - **XGBoost Classifier**
     - **Random Forest**
4. **בדיקת הדיוק של המודלים**
   - המודלים נבחנו על סט בדיקה (80% מהדאטה).
   - לכל מודל- חישוב דיוק כללי ודיוק לכל סוג שירות בנפרד.

## תוצאות המודלים

```
--- Support Vector Machine ---
Per-Class Accuracy:
  Audio Streaming: 0.5346
  Video Conferecing: 0.2690
  Video Streaming: 0.3794
  Web Surfing: 0.9329
Overall Model Accuracy: 0.5290
```

```
--- Logistic Regression ---
Per-Class Accuracy:
  Audio Streaming: 0.5806
  Video Conferecing: 0.3465
  Video Streaming: 0.5789
  Web Surfing: 0.9736
Overall Model Accuracy: 0.6199
```

```
--- XGBoost Classifier ---
Per-Class Accuracy:
  Audio Streaming: 0.7901
  Video Conferecing: 0.8841
  Video Streaming: 0.8790
  Web Surfing: 0.9942
Overall Model Accuracy: 0.8868
```

```
--- Random Forest ---
Per-Class Accuracy:
  Audio Streaming: 0.8026
  Video Conferecing: 0.8861
  Video Streaming: 0.8867
  Web Surfing: 0.9946
Overall Model Accuracy: 0.8925
```

## מסקנות

1. המודלים הלינאריים, SVM ו- Logistic Regression, הציגו ביצועים חלשים (52-62% דיוק). לעומתם, המודלים מבוססי העצים, XGBoost ו-Random Forest, נתנו את התוצאות הכי טובות (כ-89% דיוק).
2. Web Surfing מזוהה כמעט תמיד נכון (~99% הצלחה), כי יש לו דפוס מאוד ברור של תעבורה.
3. Video Conferencing התקשה מאוד במודלים הלינאריים, אך השתפר משמעותית במודלים מבוססי העצים (מ-34% Logistic Regression ו-26% ב-SVM, ל-88% ב-XGBoost ו-Random Forest).
4. סך הכל הקטגוריות הכי בעייתיות לזיהוי היו הסטרימינג- Audio Streaming לעומת Video Streaming, מכיוון שהדימיון ביניהם הקשה על המודלים להפריד ביניהם.

**התוצאות שקיבלנו כעת, לאחר השימוש במודלים השונים, מציגות כי למרות שלתוקף יש כעת רק נתונים בסיסיים ביותר, ניתן להגיע לסיווג טוב יותר משמעותית.**

# שאלת הבונוס

## ניתוח תרחיש: שימוש בשתי אפליקציות בו-זמנית

כאשר משתמש מפעיל אפליקציה עיקרית באופן רציף (למשל, שירות סטרימינג או שיחת וידאו) ובמקביל משתמש מדי פעם באפליקציה נוספת (למשל, גלישה באינטרנט או שליחת אימיילים), תעבורת הרשת משתנה, וניתן לנתח זאת כדי לזהות דפוסים המעידים על סוג הפעילות.

מה קורה ברשת כאשר משתמש עובד עם שתי אפליקציות במקביל?

כאשר משתמש מפעיל אפליקציה ראשית, נוצרת תבנית קבועה יחסית של חבילות רשת. דפוס זה תלוי באופי האפליקציה הראשית. לדוגמה:

- וידאו סטרימינג שולח חבילות נתונים גדולות בתדירות יחסית קבועה (Buffering מראש והזרמת נתונים).
- שיחת וידאו משתמשת בחבילות קטנות יותר עם תדירות קבועה כדי להבטיח תקשורת רציפה.
- אפליקציות אודיו (כגון מוזיקה) שולחות חבילות בינוניות בקצב קבוע.

כאשר האפליקציה המשנית נכנסת לפעולה לזמן קצר, היא יוצרת שינוי רגעי בתבנית התעבורה, אותו ניתן לזהות במספר דרכים:

1. גודל חבילות משתנה – למשל, אם האפליקציה הנוספת שולחת מידע קטן בתדירות גבוהה (כמו צ'אט) או מידע גדול בפרצי תעבורה (כמו הורדת קובץ).
2. זמן הגעה בין חבילות (Inter-arrival Time) – עשוי להשתנות עקב שילוב של שתי זרימות נתונים נפרדות.
3. יחס בין תעבורה נכנסת ליוצאת – עשוי להשתנות באופן זמני, למשל, כאשר אפליקציה נוספת דורשת מידע נוסף מהשרת.
4. עלייה רגעית במספר החיבורים החדשים – ייתכן שייפתח חיבור נוסף לפרוטוקול אחר.

## ניתוח תוקף – כיצד ניתן לזהות שימוש בשתי אפליקציות?

### תרחיש 1: לתוקף יש גישה ל- tuple-4, גודל חבילות וחותמות זמן

כאשר התוקף מסוגל לראות למי נשלחות החבילות ובאילו פורטים, יש לו מידע מדויק יותר:

- הוא יכול לבדוק אם יש שני זרמים שונים של תעבורה המגיעים מכתובת IP אחת.
- אם יש שינוי פתאומי במספר היעדים, הוא יכול להסיק שאפליקציה חדשה הופעלה.
- אם מתגלים שני פורטים שונים פועלים במקביל (למשל, פורט אחד ל-Spotify ופורט נוסף לדפדפן), ניתן להסיק שהמשתמש משתמש בשתי אפליקציות בו-זמנית.

למשל אם האפליקציה הראשית היא שירות סטרימינג, אך לפתע מתחילים להופיע חיבורים לכתובות IP של שירותי מייל, ניתן להניח שהמשתמש שלח אימייל.

איך ניתן להקשות על זיהוי כזה?

1. שימוש ב-VPN – יטשטש את כתובות ה-IP, כך שהתוקף לא יוכל לזהות שירותים שונים בקלות.
2. שימוש בפרוטוקולים שמצפינים נתונים על מנת להקשות על זיהוי סוג האפליקציה.
3. יצירת חיבורים נוספים רנדומליים – אפשר להפעיל חיבורים פיקטיביים כדי לבלבל את התוקף.

### מסקנה על תרחיש 1:

בתרחיש זה, דיוק הזיהוי של התוקף הוא **טוב למדי,** מכיוון שהוא יכול להצליב בין כתובות IP, פורטים ופרוטוקולים כדי להסיק אילו שירותים נמצאים בשימוש. כאשר שתי אפליקציות רצות במקביל, ניתן לראות שינוי בנתונים כגון פתיחת חיבור נוסף, שימוש בפורטים שונים, או שינוי ביחס בין תעבורה נכנסת ליוצאת. התוקף יכול לזהות לא רק שקיימות שתי אפליקציות פעילות, אלא גם אילו שירותים ספציפיים הם. עם זאת, שימוש ב-VPN או Proxy עשוי להפחית את דיוק הזיהוי.

### תרחיש 2: לתוקף יש רק גודל חבילות וחותמות זמן

כאשר לתוקף אין מידע על ה-tuple-4, הוא יכול לנתח את דפוסי גודל החבילות וזמני ההגעה ביניהן כדי להסיק אם יש שימוש בשתי אפליקציות בו-זמנית.

כיצד התוקף יכול לזהות שינוי בהתנהגות הרשת?

1. הפרעות בדפוס הרגיל של החבילות – אם תעבורת הרשת הייתה קבועה ואז הופיעו שינויים בלתי מוסברים, זה עשוי להעיד על אפליקציה נוספת שפועלת.
2. עלייה פתאומית בגודל חבילות מסוימות – למשל, אם תעבורה רגילה של אודיו מתבצעת בחבילות קטנות אך לפתע מופיעות חבילות גדולות יותר, יכול להיות שהמשתמש פתח דפדפן והחל לגלוש.
3. שינוי בקצב שליחת החבילות – שירות סטרימינג שולח חבילות בקצב יחסית אחיד, אך כאשר אפליקציה חדשה מופעלת, ייתכנו תקופות של Burst Data (התפרצויות נתונים).
4. יחס משתנה בין זמני הגעה – אם יש פתאום חבילות עם זמני הגעה גבוהים יותר, זה עשוי להצביע על עומס נוסף באפליקציה חדשה.

איך ניתן להקשות על זיהוי כזה?

1. יצירת דפוס אחיד של חבילות – למשל, אפליקציות יכולות להשתמש באותו גודל חבילות באופן מלאכותי כדי למנוע הבחנה בין סוגי תעבורה.
2. Padding של חבילות (מילוי נתונים ריקים) – שליחת חבילות בגודל אחיד כדי שלא יהיה ניתן לזהות שינוי בין אפליקציות שונות.
3. שליחת חבילות "דמי" בזמנים אקראיים – כך שהתוקף יתקשה לזהות מתי באמת בוצעה פעולה חדשה.
4. שימוש בפרוטוקולים מאובטחים עם רנדומיזציה – למשל, שימוש בפרוטוקולים שמאחדים את גודל החבילות באופן דינמי כדי למנוע זיהוי לפי דפוסים קבועים.

### מסקנה על תרחיש 2:

בתרחיש זה, **דיוק הזיהוי נמוך יותר**, אך עדיין ניתן לזהות במקרים שבהם האפליקציות הנמצאות בשימוש מייצרות דפוסי תעבורה שונים. כאשר האפליקציה המשנית גורמת לשינוי ברור בגודל החבילות או בתדירות שליחתן, ניתן לזהות כי פעילות נוספת מתבצעת. עם זאת, כאשר שתי האפליקציות דומות באופיין, כמו דפדפן ושירות וידאו, הזיהוי הופך לפחות חד-משמעי. עיכובים טבעיים ברשת או שימוש בטכניקות כמו Padding ורנדומיזציה של החבילות עשויים להוריד את דיוק הזיהוי משמעותית ולהקשות על התוקף לזהות אם קיימת פעילות כפולה.

### מסקנה כללית:

יכולת הזיהוי של התוקף תלויה בעיקר במידע שברשותו. כאשר יש גישה ל-tuple-4, הדיוק טוב בהרבה, ומאפשר לתוקף לזהות את מספר האפליקציות הפועלות ואת סוגיהן באופן מוצלח למדי. לעומת זאת, כאשר המידע מוגבל רק לגודל וזמני החבילות, הזיהוי הופך למבוסס יותר על אומדנים והשוואות עקיפות, מה שמפחית את הדיוק ומאפשר לזהות פעילות כפולה **בתנאים מסוימים וספציפיים**. טכניקות כמו VPN, Padding, ורנדומיזציה של חבילות עשויות לשבש את ניתוח הנתונים ולצמצם את יכולת הזיהוי משמעותית.