**ELEC-E7130 Internet Traffic Measurements and Analysis**

**Assignment 3. User traffic**

Name: Xingji Chen

Student ID: 101659554

E-mail: xingji.chen@aalto.fi

Task 1: Introduction to the traffic data

1. What is the passive measurement in terms of network traffic? What kind of information does it provide, and what is its role or significance?

* Passive measurement of network traffic is the process of observing and collecting network traffic data without actively interfering with or modifying the traffic itself. This type of measurement is non-intrusive and relies on monitoring and analyzing the way network traffic naturally flows through the network. Passive measurements do not inject test traffic or generate additional network load.
* Passive measurement systems collect various types of data from network traffic. This data can include packet headers, payload content, flow statistics, timestamps, and more.
* Passive measurement is important because it provides a comprehensive and non-intrusive view of network behavior. It can help optimize network performance, enhance security, and ensure better quality of service for end users.

1. Please provide an explanation of the concepts of packet capture and flow data. What kind of information they can provide? Additionally, discuss the advantages, disadvantages, and importance of both packet capture and flow data in network analysis.

* Packet capture
* Packet capture is the process of intercepting and recording packets traveling over a network. It involves capturing individual network packets transmitted over the network infrastructure.
* Each packet typically contains a portion of the data being sent, as well as metadata such as source and destination IP addresses, ports, protocol information, and timestamps.
* Advantages:

1. Detail: provides the most detailed information allowing in-depth analysis of network traffic.
2. Security: detects network intrusions, malware and suspicious activity by examining packet contents.
3. Troubleshooting: helps identify and diagnose network problems, performance bottlenecks, and communication issues.

* Disadvantages:

1. Amount of data: capturing all packets may generate large amounts of data quickly, requiring large amounts of storage capacity.
2. Privacy concerns: packet capture can raise privacy concerns, especially when capturing sensitive or personal data.
3. Processing overhead: capturing, storing, and analyzing packet data may consume significant computational resources.

* Importance:

Packet capture is critical for monitoring and analyzing network traffic. It helps diagnose network problems, optimize network performance and identify security threats.

* Flow data
* Flow data represents a summarized view of network traffic.
* Flow data records information about traffic flows that share common attributes such as source and destination IP addresses, ports, and protocols. Flow data is typically generated by network devices such as routers and switches and sent to a collector for analysis.
* Advantages:

1. Efficiency: requires fewer storage and processing resources than packet capture due to data aggregation.
2. Scalability: suitable for analyzing large-scale networks with high traffic volumes.

* Disadvantages:

1. Limited level of detail: lack of detailed information makes it unsuitable for deep packet inspection.
2. Unable to capture payload: flow data does not capture the actual contents of the packet, including application data.

* Importance:

Flat data is important for monitoring network usage, identifying traffic patterns and detecting potential security threats, especially in large and complex network environments.

1. What is hashing? How does the hash algorithm work and what is the relation with the memory management in the large data analysis?

* Hashing is a mathematical function that maps input data of arbitrary length to output data of fixed length. The main purpose of the hashing algorithm is to create a fixed-size piece of data called a hash (or hash value) that should ideally be unique for different input data.
* Hash algorithms receive input data, which can be text, files, passwords, or numbers of arbitrary length. The algorithm takes the input data and converts it into a fixed-length binary string through mathematical operations, such as modulo, bitwise operations, and encryption. After processing, the algorithm generates a unique hash value, usually a fixed-length binary or hexadecimal string.
* In big data analytics, data is usually distributed across multiple nodes or servers. Through hashing algorithms, data can be sliced and distributed to different nodes according to a particular criterion to achieve distributed storage and processing of data.

Task 2: Analyse flow data

* Data acquisition

# Get network adapter information  
Get-NetAdapter  
# Capture network traffic using dumpcap  
Start-Process -FilePath "dumpcap" -ArgumentList "-i", "WLAN", "-w", "D:\data\ass3.pcap" -NoNewWindow -Wait

* First execute the command to get information about the network adapters on the computer. It is usually used to list the details of the network adapters installed on the computer, including name, status, speed, etc.
* Then executes the command to capture network packets, which is used in conjunction with the Wireshark network analysis tool. It specifies the network adapter ("WLAN") to be captured and saves the captured packets to the file "D:\data\ass3.pcap".
* Data processing

# Update package list  
sudo apt-get update  
# Install tshark  
sudo apt-get install tshark  
# Analyze the PCAP file and save flow data to flow\_data.txt  
tshark -r T2\_data.pcap -q -z conv,tcp > flow\_data.txt  
# Analyze the PCAP file with verbose output and save flow data to flow\_data1.txt  
tshark -r T2\_data.pcap -q -z conv,tcp -V > flow\_data1.txt

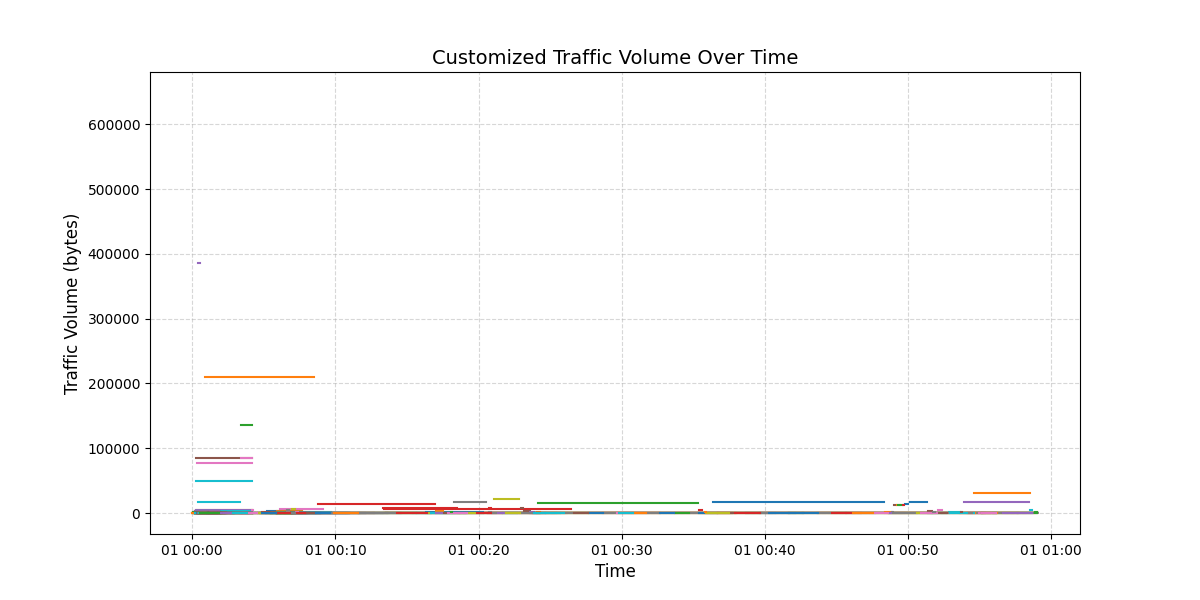
* First update system's package list to ensure that the latest version of the package can be installed. And install Tshark, which is the command line version of the Wireshark network analysis tool. Tshark is used to analyze network packets and can extract information about network traffic from PCAP files.
* Then use Tshark to extract the TCP network flow information from the PCAP file named "T2\_data.pcap" and output the result to a text file named "flow\_data.txt". More detailed information, including packet contents for each TCP session, is output to a text file named "flow\_data1.txt".

1. Provide basic statistics of flow data, including

* total number of flows,
* minimum, median, mean and maximum flow sizes in bytes and packets

|  |  |
| --- | --- |
| Item | Value |
| Total number of flows | 1522 |
| Minimum flow size (bytes) | 54 |
| Median flow size (bytes) | 756.5 |
| Mean flow size (bytes) | 159005.5959264126 |
| Maximum flow size (bytes) | 96468992 |
| Minimum flow size (packets) | 1 |
| Median flow size (packets) | 10.0s |
| Mean flow size (packets): | 345.65834428383704 |
| Maximum flow size (packets) | 95011 |

1. Plot the traffic volume (bytes) of the flow data file.



1. Please provide the top 5 most commonly used protocols, as well as the five most common source ports and five most common destination ports based on flows. Detail in a table for each one

* the number of flows
* the number of packets
* the amount of data (bytes)
* the application or usage

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Top 5 Protocols | | | | |
| Protocol | Flows | Packets | Bytes | Application |
| eth:ethertype:ip:udp:dtls | 782217 | 782217 | 632113440 | HTTP |
| eth:ethertype:ip:tcp:tls | 285111 | 285111 | 65997699 | HTTP |
| eth:ethertype:ip:tcp | 228697 | 228697 | 154422789 | HTTP |
| eth:ethertype:ip:udp:quic | 191186 | 191186 | 203161985 | HTTP |
| eth:ethertype:ip:udp:stun | 20518 | 20518 | 2567736 | HTTP |

|  |  |
| --- | --- |
| Top 5 Source Ports | |
| Port | Count |
| 443 | 453247 |
| 11599 | 247252 |
| 64150 | 125608 |
| 55021 | 103432 |
| 62415 | 52097 |

|  |  |
| --- | --- |
| Top 5 Destination Ports | |
| Port | Count |
| 443 | 252321 |
| 64150 | 247277 |
| 11599 | 125568 |
| 51969 | 107335 |
| 62415 | 103445 |

1. Which are the top-ten host pairs based on

* number of flows
* number of bytes

Are there the same pairs?

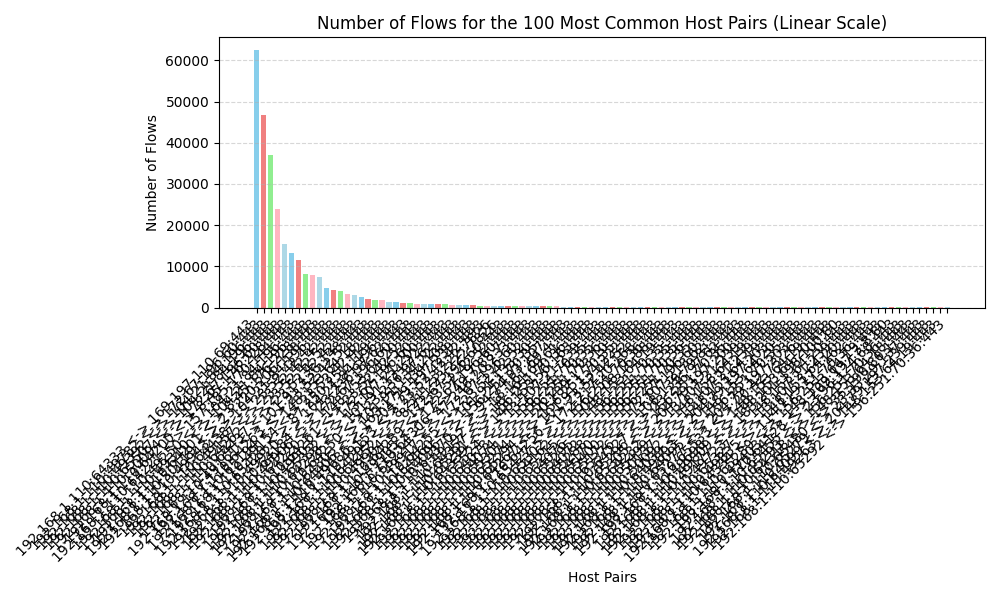
* No, there are different pairs.

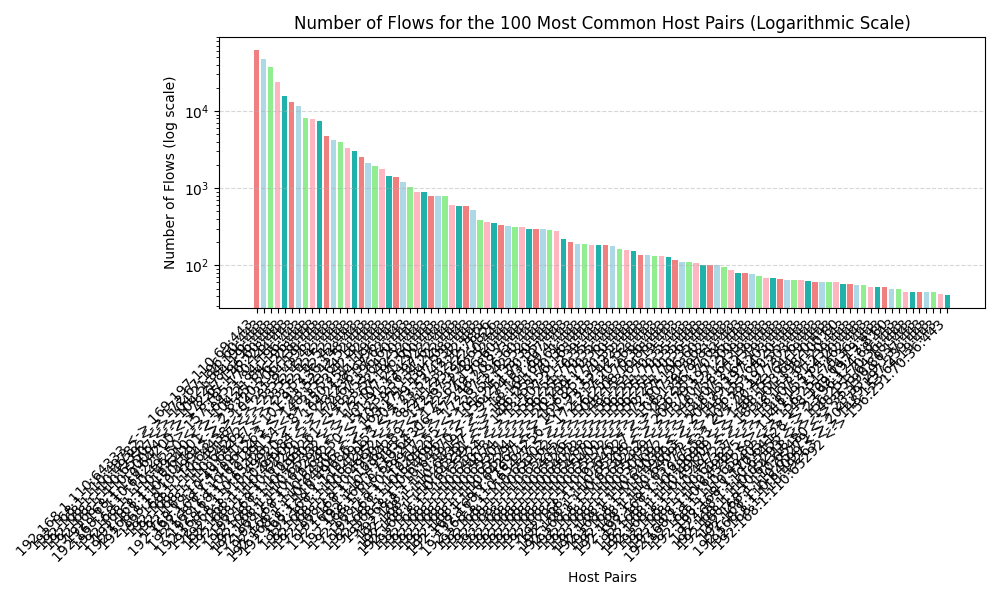
|  |  |  |
| --- | --- | --- |
| Top-ten host pairs based on number of flows | | |
| Pairs | Number of flows | Number of bytes |
| 192.168.1.110:64333 <-> 169.197.110.69:443 | 62457 | 94371840 |
| 192.168.1.110:49782 <-> 172.67.196.60:443 | 46611 | 6878208 |
| 192.168.1.110:49392 <-> 104.21.84.196:443 | 37049 | 6151168 |
| 192.168.1.110:65507 <-> 104.21.84.196:443 | 23885 | 3646464 |
| 192.168.1.110:50094 <-> 172.67.196.60:443 | 15445 | 2847744 |
| 192.168.1.110:64200 <-> 8.45.176.228:443 | 13224 | 18874368 |
| 192.168.1.110:64221 <-> 157.185.170.144:443 | 11478 | 17825792 |
| 192.168.1.110:49250 <-> 104.21.84.196:443 | 8225 | 1394688 |
| 192.168.1.110:49310 <-> 104.21.84.196:443 | 7891 | 1370112 |
| 192.168.1.110:64201 <-> 8.45.176.228:443 | 7340 | 10485760 |

|  |  |  |
| --- | --- | --- |
| Top-ten host pairs based on number of flows | | |
| Pairs | Number of flows | Number of bytes |
| 192.168.1.110:64333 <-> 169.197.110.69:443 | 62457 | 94371840 |
| 192.168.1.110:64200 <-> 8.45.176.228:443 | 13224 | 18874368 |
| 192.168.1.110:64221 <-> 157.185.170.144:443 | 11478 | 17825792 |
| 192.168.1.110:64201 <-> 8.45.176.228:443 | 7340 | 10485760 |
| 192.168.1.110:50119 <-> 23.220.206.138:443 | 4690 | 7142400 |
| 192.168.1.110:49782 <-> 172.67.196.60:443 | 46611 | 6878208 |
| 192.168.1.110:64438 <-> 23.52.42.42:443 | 4009 | 6181888 |
| 192.168.1.110:49392 <-> 104.21.84.196:443 | 37049 | 6151168 |
| 192.168.1.110:49157 <-> 23.52.42.42:443 | 3337 | 5089280 |
| 192.168.1.110:64262 <-> 23.52.42.52:443 | 3064 | 4714496 |

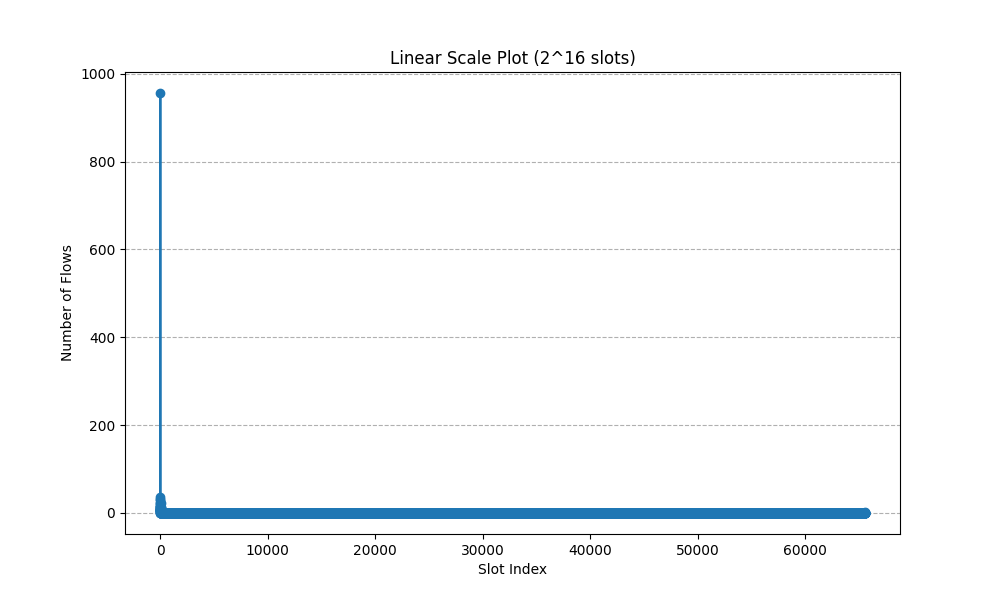
1. Plot the number of flows for the 100 most common pairs of hosts

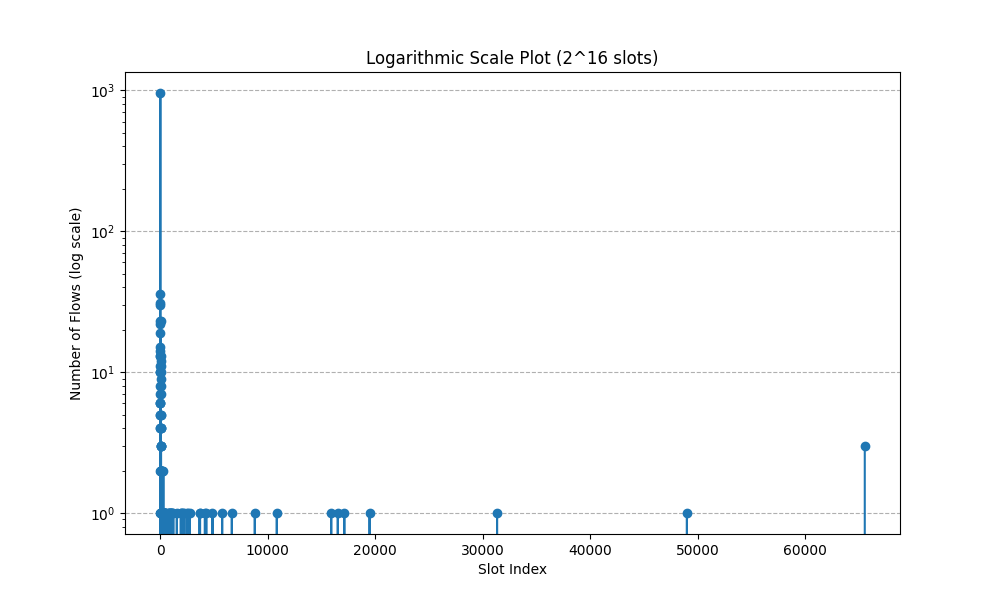
* Using linear scale
* Using logarithmic scale





1. Repeat the previous plot (both linear and logarithmic scale) using this time fixed size (216 slots) array approach (Network capture tutorial - Large data analysis, pp. 8 and solution #2, pp. 10). What can you say about the results?





* In linear scale, the number of flows is concentrated in the front slots.
* In logarithmic scale, the number of flows is concentrated in the front and back slots.

1. Is there a more efficient approach in terms of running time and memory consumption to accomplish this task?

* Use Tshark's filtering options to reduce the amount of data that needs to be processed. Use the -Y option to specify display filtering expressions to focus on specific types of traffic, protocols, or IP addresses. This can significantly reduce the number of packets Tshark needs to analyze.
* Create batch scripts or use a scripting language such as Python to automate the process. This makes it possible to process multiple files consecutively without manual intervention.
* Make sure that the system has enough RAM to handle the size of the PCAP files. Insufficient RAM may slow down the analysis or even cause the analysis tool to crash.
* Access to a multi-core processor system and explore parallel processing options to analyze PCAP files concurrently, which can significantly reduce overall processing time.

Task 3: Analyse packet capture (user traffic)

* Main function

# Function to convert bytes to kilobytes  
def bytes\_to\_kilobytes(bytes\_value, unit):  
 byte\_units = {'bytes': 1, 'kb': 1024, 'mb': 1024\*\*2}  
 return bytes\_value \* byte\_units.get(unit.lower(), 1)

* This function accepts two arguments: bytes\_value for the byte value to be converted and unit for the byte unit ('bytes', 'kilobytes', 'mb'). It will convert the byte value to kilobytes (KB) according to the byte unit and return the result.

def geocode(address, attempt=1, max\_attempts=5):  
 try:  
 geolocator = Nominatim(user\_agent="geoip\_app")  
 return geolocator.geocode(address)  
 except GeocoderTimedOut:  
 if attempt <= max\_attempts:  
 return do\_geocode(address, attempt=attempt+1)

* This function is used to geocode by address, i.e. to convert an address to geographic coordinates (longitude and latitude). This function uses recursion to handle exceptions and retries if the geocoding request times out.

source\_ips = df['source\_interface'].apply(lambda x: x.split(':')[0]).unique()  
dest\_ips = df['destination\_interface'].apply(lambda x: x.split(':')[0]).unique()

* This code is used to extract the unique IPv4 address from the data, which can be used for further analysis or to count the number of different IPv4 addresses.

1. How many IPv4 hosts (and IPv6, if any) are communicating?

* The number of IPv4 hosts is 162.

1. Top 5 host countries (e.g. GeoIP)

|  |  |
| --- | --- |
| Top 5 Host Countries | |
| Country | Count |
| العراق | 19 |
| United States | 7 |
| Türkiye | 7 |
| Česko | 4 |
| Canada | 3 |

1. Top 15 hosts by byte counts.

|  |  |
| --- | --- |
| Top 15 Hosts by Byte Counts | |
| IP | Bytes |
| 192.168.1.110:64333 | 96468992 |
| 192.168.1.110:64200 | 19922944 |
| 192.168.1.110:64221 | 17825792 |
| 192.168.1.110:49782 | 12582912 |
| 192.168.1.110:64201 | 11534336 |
| 192.168.1.110:49392 | 10485760 |
| 192.168.1.110:50119 | 7266304 |
| 192.168.1.110:65507 | 6728704 |
| 192.168.1.110:64438 | 6290432 |
| 192.168.1.110:49157 | 5196800 |
| 192.168.1.110:64262 | 4799488 |
| 192.168.1.110:50094 | 4628480 |
| 192.168.1.110:64437 | 3939328 |
| 192.168.1.110:64263 | 2999296 |
| 192.168.1.110:49250 | 2282496 |

1. Top 15 hosts by packet counts. Were there any differences between the top 15 hosts in terms of byte counts and packet counts?

* Yes.

|  |  |
| --- | --- |
| Top 15 Hosts by Packet Counts | |
| IP | Bytes |
| 192.168.1.110:49782 | 95011 |
| 192.168.1.110:64333 | 91367 |
| 192.168.1.110:49392 | 76099 |
| 192.168.1.110:65007 | 48991 |
| 192.168.1.110:50094 | 31331 |
| 192.168.1.110:64200 | 19474 |
| 192.168.1.110:64221 | 17122 |
| 192.168.1.110:49250 | 16517 |
| 192.168.1.110:49310 | 15905 |
| 192.168.1.110:64201 | 10844 |
| 192.168.1.110:49943 | 8797 |
| 192.168.1.110:50119 | 6660 |
| 192.168.1.110:64438 | 5761 |
| 192.168.1.110:49157 | 4862 |
| 192.168.1.110:64262 | 4283 |

1. Top 10 TCP and top 5 UDP port numbers (by packet count).

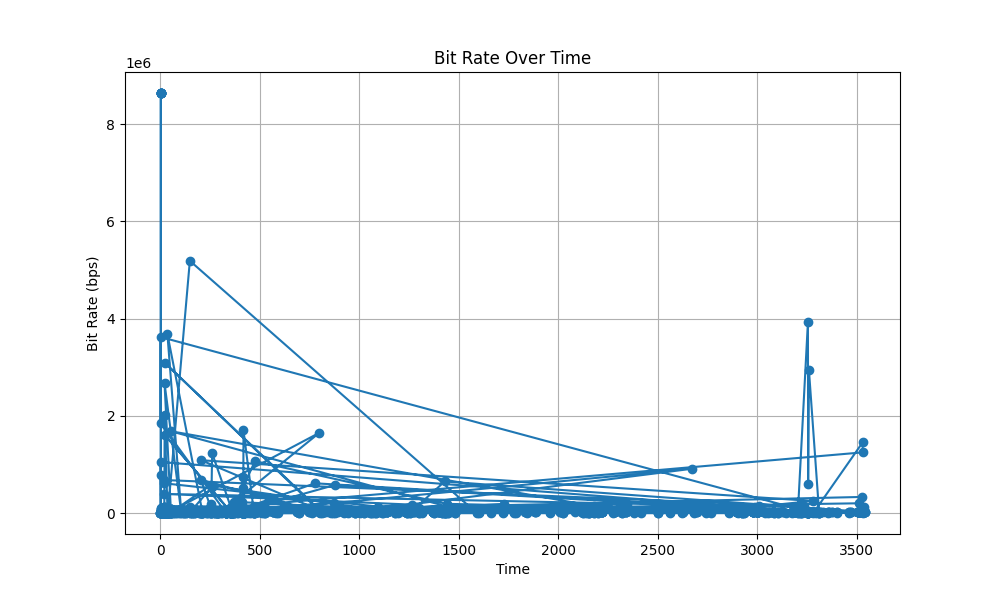
* UDP is not used.

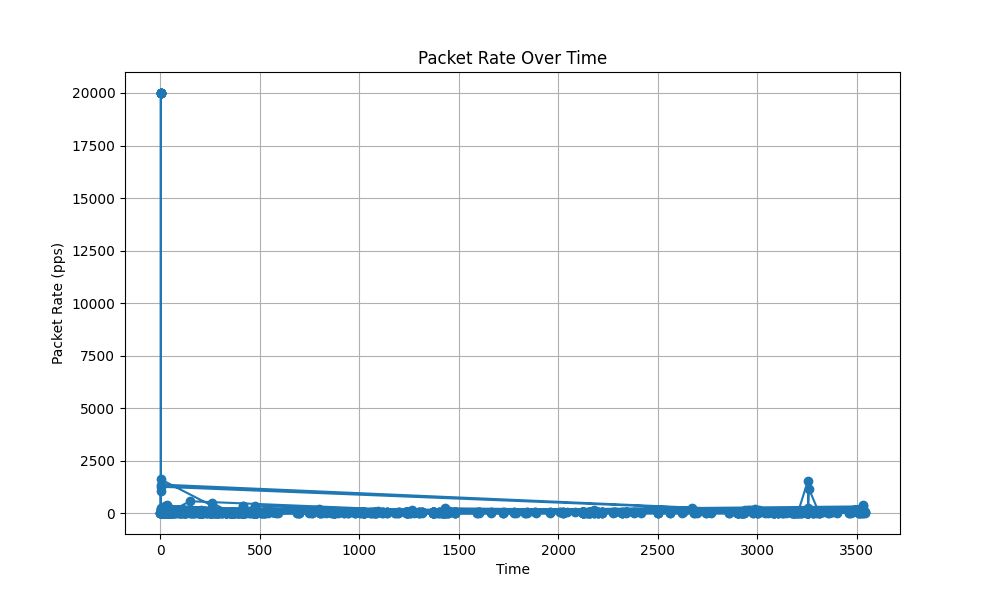
|  |  |
| --- | --- |
| Top 10 TCP Port Numbers (by Packet Count) | |
| Port | Count |
| 443 | 513920 |
| 53 | 9716 |
| 7826 | 1686 |
| 80 | 482 |
| 1900 | 120 |
| 63984 | 26 |
| 63985 | 26 |
| 63989 | 26 |
| 64110 | 11 |
| 64123 | 11 |

1. Top 10 fastest TCP connections

|  |  |  |
| --- | --- | --- |
| Top 10 Fastest TCP Connections | | |
| First IP interface | Second IP interface | Speed |
| 192.168.1.110:64047 | 49.4.45.146:443 | 1.080000e+06 |
| 192.168.1.110:64031 | 49.4.17.138:443 | 1.080000e+06 |
| 192.168.1.110:64003 | 49.4.17.138:443 | 1.080000e+06 |
| 192.168.1.110:64004 | 49.4.17.138:443 | 1.080000e+06 |
| 192.168.1.110:64032 | 49.4.17.138:443 | 1.080000e+06 |
| 192.168.1.110:64010 | 47.246.20.231:443 | 1.080000e+06 |
| 192.168.1.110:64014 | 49.4.17.138:443 | 1.080000e+06 |
| 192.168.1.110:64002 | 49.4.17.138:443 | 1.080000e+06 |
| 192.168.1.110:64406 | 163.171.134.108:443 | 6.484586e+05 |
| fe80::cad9:c2c0:7c6c:9cd9:50105 | fe80::3e6a:48ff:fec3:e6ed:1900 | 4.911538e+05 |

1. Bit and packet rate over time (e.g. tcpstat, capinfos)





1. How many hosts were tried to contact to, but communication failed for a reason or another? Can you identify different subclasses of failed communications?

* The number of hosts that tried to communicate but failed is 0, which means there is no failed communication.
* Summarize
* In this task, I initially captured the data packets transmitted by the computer over a specific time frame, resulting in the acquisition of a pcap (packet capture) file. Subsequently, I proceeded to transform this pcap file into flow data and subsequently conducted a comprehensive analysis of the captured information.
* After analyzing the flow data, I extracted basic statistical information regarding the flow data, host pairs, flow count, and data volume. This information helps us gain a deeper understanding of the characteristics and trends in network activity.
* The packet capture was subjected to analysis, from which we extracted valuable information including host count, IP addresses, and TCP-related data.