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

**Assignment 4. Traffic with probe packets**

Name: Xingji Chen

Student ID: 101659554

E-mail: xingji.chen@aalto.fi

Task 1: Packet capture with probe packets

1. Use the subprocess module to run Ping commands to test network connectivity and response times for multiple destination addresses, recording the date and time stamp of each command. Write the results of the Ping test to a text file associated with each destination address. Information about the destination addresses and the output files is stored in a list called destinations. Perform a Ping test for each destination address and save the results to the appropriate file for subsequent analysis and monitoring of network performance.

import subprocess  
import time  
  
def ping\_and\_save(destination, output\_file):  
 current\_datetime = time.strftime("%Y-%m-%d %H:%M:%S")  
 with open(output\_file, 'a') as file:  
 file.write(current\_datetime + '\n')  
  
 try:  
 ping\_result = subprocess.run(['ping', '-n', '5', destination], capture\_output=True, timeout=5, text=True)  
 with open(output\_file, 'a') as file:  
 file.write(ping\_result.stdout + '\n')  
 except subprocess.TimeoutExpired:  
 with open(output\_file, 'a') as file:  
 file.write("Ping超时\n")  
  
destinations = [  
 {"name": "ok1.iperf.comnet-student.eu", "ip": "195.148.124.36"},  
 {"name": "blr1.iperf.comnet-student.eu", "ip": "142.93.213.224"},  
 {"name": "cbg-uk.ark.caida.org", "ip": "128.232.97.9"},  
 {"name": "bjl-gm.ark.caida.org", "ip": "196.46.233.22"},  
 {"name": "msy-isu.ark.caida.org", "ip": "non"}  
]  
  
output\_directory = "ping\\"  
  
for destination in destinations:  
 output\_file = output\_directory + destination["name"] + ".txt"  
 ping\_and\_save(destination["name"], output\_file)

1. Capture network packets from a WLAN interface and save them to a PCAP file. Use PowerShell to retrieve network statistics for the WLAN network adapter.

import subprocess  
  
dumpcap\_command = 'dumpcap -i "WLAN" -w "D:\\as4\\CHENXINJI2.pcap"'  
subprocess.run(dumpcap\_command, shell=True)  
  
powershell\_command = 'powershell Get-NetAdapterStatistics -Name "WLAN"'  
subprocess.run(powershell\_command, shell=True)

1. The iperf3 tool for network performance testing is used to connect to separate target servers with the option to specify custom ports.

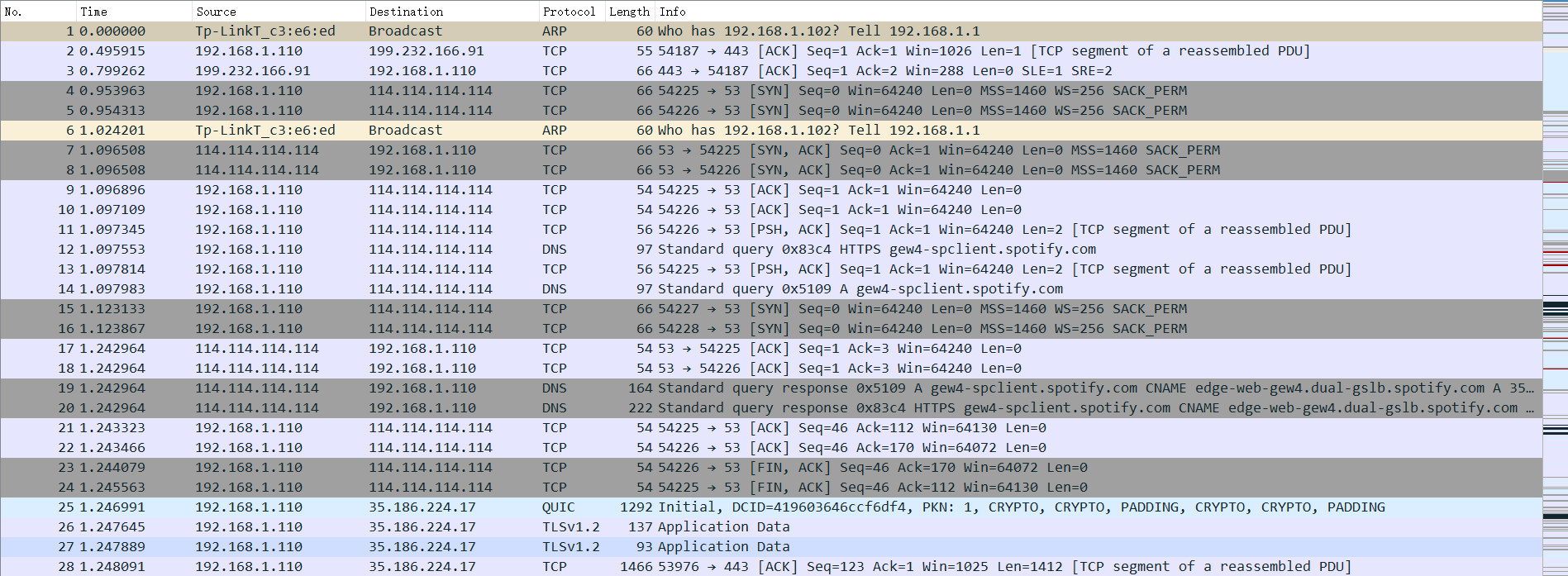
D:\iperf3.exe -c ok1.iperf.comnet-student.eu  
D:\iperf3.exe -c blr1.iperf.comnet-student.eu -p *5203*

1. The tshark tool, which is a command line version of Wireshark, was used to extract session statistics of IP and TCP packets from different PCAP files for network packet analysis and performance evaluation.

tshark -r icmp.pcap -q -z conv,ip > icmp.txt  
tshark -r iperf.pcap -q -z conv,tcp > iperf.txt

1. Sanity checks

|  |  |
| --- | --- |
| Size of trace file | 1102294535 bytes |
| Number of packets in trace file | 900712 |
| Total size of packets | 1102294535 bytes |

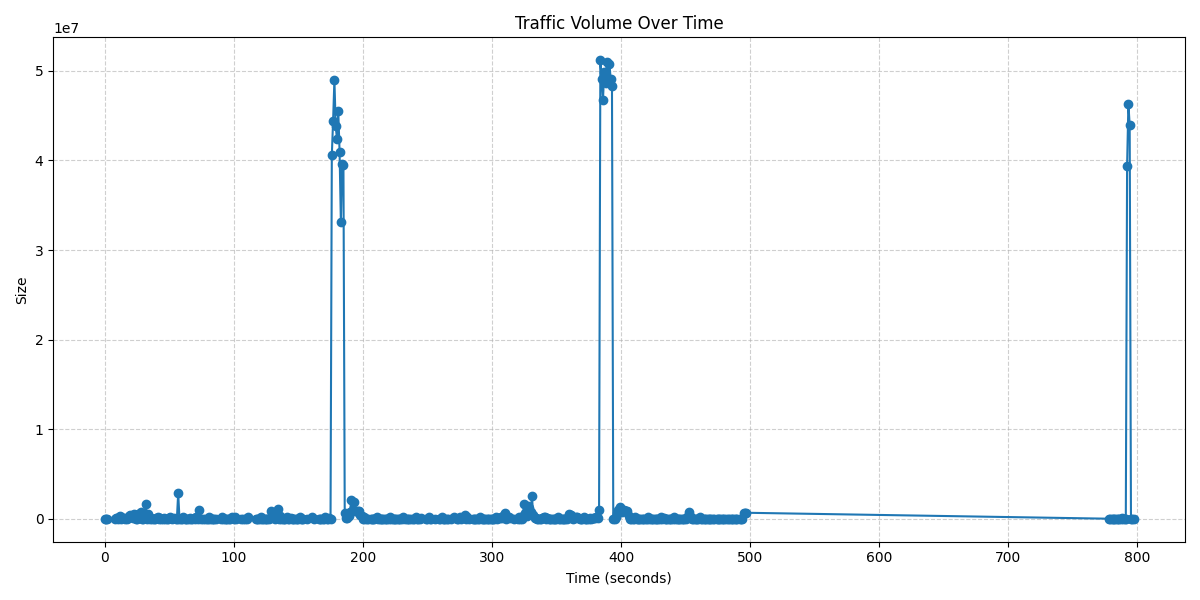


Capture file statistics are related to the file itself and provide general information about the capture, while counters are statistics related to the content and behavior of network traffic. We can see that values from interface counters to capture file has some differences.

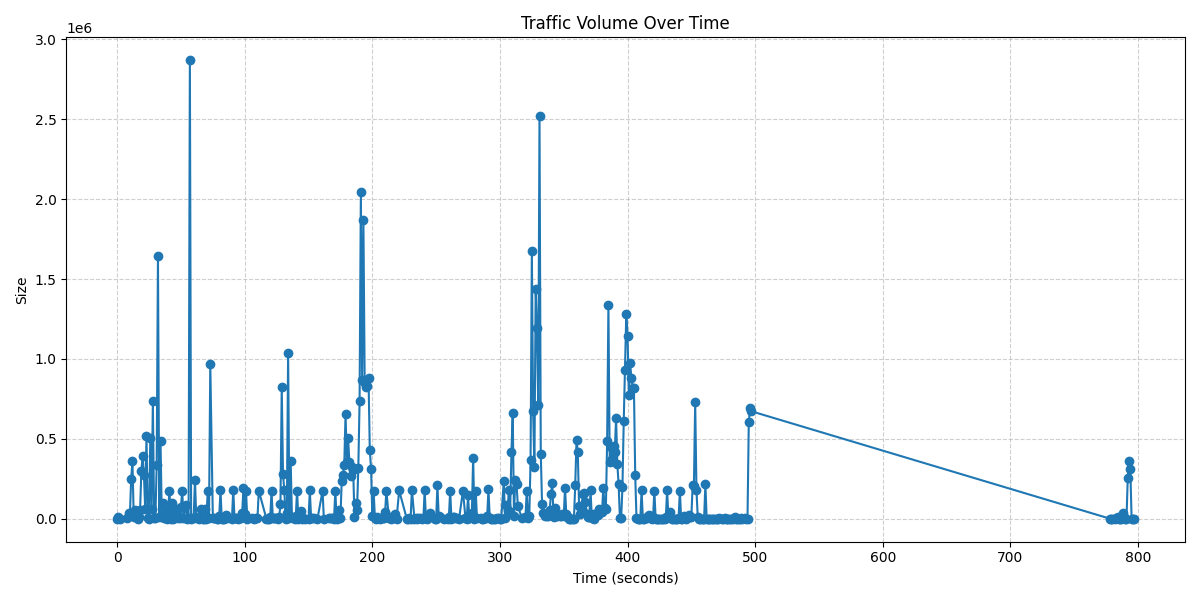
This is because capture file statistics are extracted and computed directly by analytics tools, such as Wireshark, based on file header information without the need to deeply analyze the content of the packets. Counters, on the other hand, require more detailed analysis and counting of each packet, and usually require more processing to obtain information about the packet contents.

Capture File Statistics is used to provide summary information about the capture file itself to help the user understand the overall characteristics of the data capture. Counters are used to analyze the actual content and behavior of network packets in order to gain a deeper understanding of the characteristics of network traffic, such as traffic distribution, error detection and protocol analysis.

1. Plot the traffic volume over time by considering all captured packets within the most appropriate time interval.

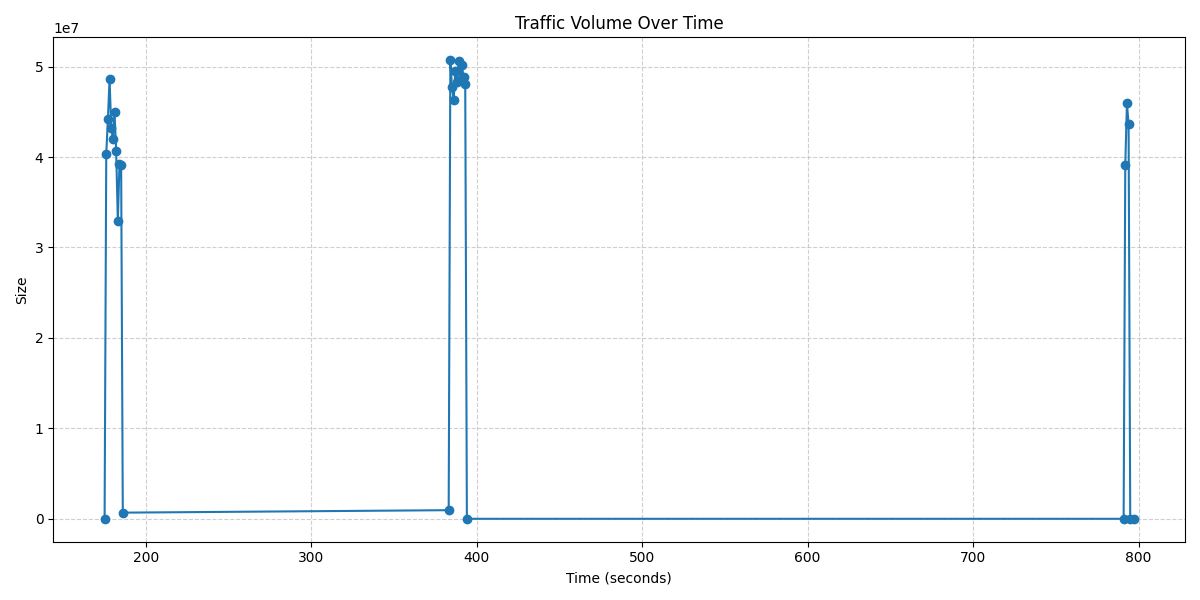


1. Plot the traffic volume without the ping packets and iperf3 packets over time (select the same interval selected in the previous plot).



1. Plot the traffic volume comparing the ping packets with the iperf3 packets over time (keeping the same interval).

iperf3:



ping:



1. Provide the average throughput.

|  |  |
| --- | --- |
| Average throughput | 1382196.1011820212 bytes per second |

1. Do you have any observations from the above plot of network traffic?

For the traffic volume over time by considering all captured packets within the most appropriate time interval, high traffic volume occurs only at certain moments, such as the 180th, 390th and 790th seconds, when the size of the traffic volume suddenly increases to 5\*10^7 bytes, while at other moments the flow fluctuates and is basically very small.

For the traffic volume without the ping packets and iperf3 packets over time, the trend of the traffic volume is not so clear, we can observe that it has been fluctuating and the peaks occur irregularly.

For the traffic volume with the iperf3 packets over time, similar to the traffic volume over time by considering all captured packets, peaks occur at some certain moments which are different. Peaks occur at 100th second, 380th second and 790th second, reaching 5\*10^7 bytes. And before or after these moments, the traffic volume stays at a very low level. For the ping packets, peaks occur more frequently, at 50th, 130th, 210th, 290th, 370th, 450th and 780th seconds. But the peak is much smaller than the former, and only reaches a maximum of 300 bytes, which indicates that the number of ping packets is very small.

Task 2: Compare active and passive measurements

1. This code performs the calculation and accumulation of packet lengths, separately calculating the total length of all packets, ICMP packets, and packets containing specific information.

with open('cxj.csv', 'r') as csvfile:  
 # Create a CSV reader  
 csvreader = csv.reader(csvfile)  
  
 # Skip the header row  
 next(csvreader)  
  
 # Initialize variables  
 total\_packet\_length = 0  
 icmp\_packet\_length = 0  
 custom\_info\_packet\_length = 0  
  
 # Iterate through each row  
 for row in csvreader:  
 # Get the length field  
 packet\_length = int(row[5])  
  
 # Calculate the total packet length across all packets  
 total\_packet\_length += packet\_length  
  
 # Calculate the packet length for ICMP protocol packets  
 if row[4] == 'ICMP':  
 icmp\_packet\_length += packet\_length  
  
 # Calculate the packet length for packets containing '5201' in the Info field  
 if ' 5201 ' in row[6]:  
 custom\_info\_packet\_length += packet\_length

1. The main function of this code is to perform byte to kilobyte unit conversion, and will be converted to the value and other relevant information returned to the Pandas Series.

ef convert\_to\_bytes(row):  
 units = {'bytes': 1, 'kb': 1024, 'mb': 1024\*\*2}  
  
 try:  
 ld\_bytes\_unit = str(row['ld\_bytes\_unit']).lower()  
 factor = units[ld\_bytes\_unit]  
 ld\_kb = row['ld\_bytes'] \* factor  
  
 rd\_bytes\_unit = str(row['rd\_bytes\_unit']).lower()  
 factor = units[rd\_bytes\_unit]  
 rd\_kb = row['rd\_bytes'] \* factor  
  
 total\_bytes\_unit = str(row['total\_bytes\_unit']).lower()  
 factor = units[total\_bytes\_unit]  
 total\_kb = row['total\_bytes'] \* factor  
  
 return pd.Series({'ld\_bytes': ld\_kb, 'rd\_bytes': rd\_kb, 'total\_bytes': total\_kb, 'server\_ip': row['second\_ip\_interface']})  
 except KeyError as e:  
 print(f"Error processing row {row}: {e}")  
 raise ValueError("Invalid unit. Supported units are 'bytes', 'kb', 'mb.")

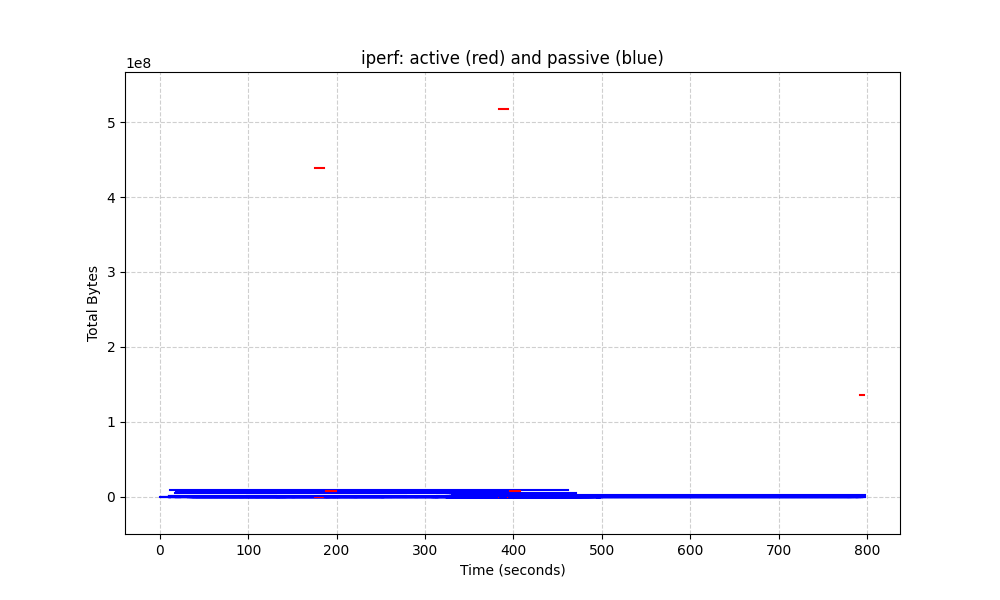
1. The purpose of this code is to draw horizontal lines of different colors based on the conditions of active and passive measurements, depending on whether server\_ip matches a particular IP address and port combination.

for (name1, name2, server\_ip), group in groups:  
 if server\_ip.strip() in {'195.148.124.36:5201', '142.93.213.224:5203'}:  
 ax.plot([group['start'], group['start'] + group['duration']], [group['total\_bytes'], group['total\_bytes'],], color='r')  
 else:  
 ax.plot([group['start'], group['start'] + group['duration']], [group['total\_bytes'], group['total\_bytes'],], color='b')

1. How much traffic was there that was not iperf or ping traffic?

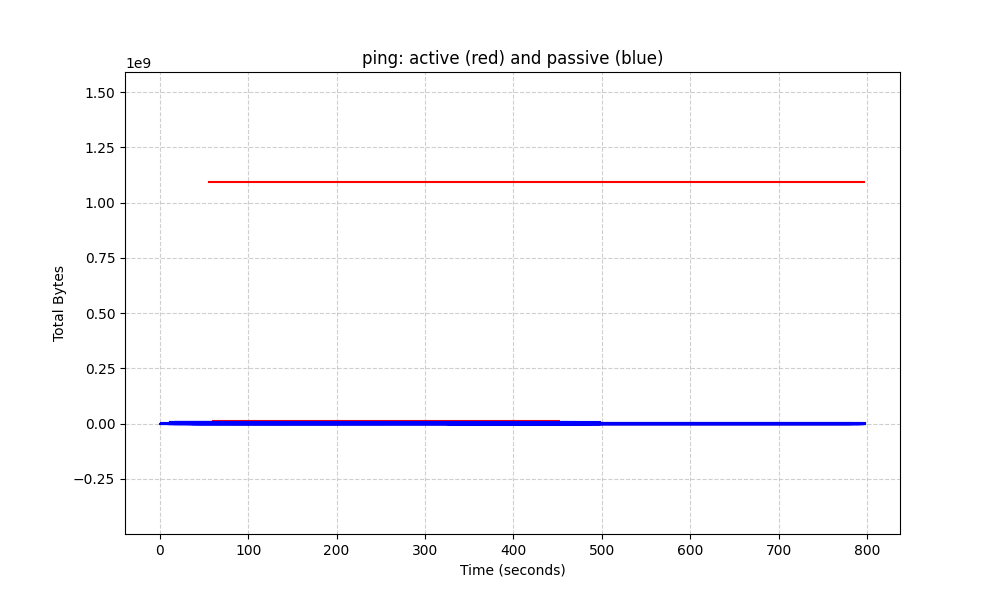
|  |  |
| --- | --- |
| all packets | 1102294535 bytes |
| packets use icmp | 18870 bytes |
| packets use 5201 port | 1034465646 bytes |
| traffic was there that was not iperf or ping traffic | 67810019 bytes |

1. Compare iperf results from active and passive measurements. Provide a table and plot a time series.



|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Ip | Left direction frames | Left direction bytes | Right direction frames | Right direction bytes | Total frames | Total bytes | Start | Duration | Server ip |
| 192.168.1.110:54405 <-> 195.148.124.36:5201 | 65565 | 3624960 | 324183 | 512753664 | 389748 | 516947968 | 383.924019 | 10.0548 | 195.148.124.36:5201 |
| 192.168.1.110:54405 <-> 195.148.124.36:5201 | 50298 | 2781184 | 275495 | 436207616 | 325793 | 438304768 | 175.960358 | 10.1746 | 195.148.124.36:5201 |
| 192.168.1.110:57283 <-> 195.148.124.36:5201 | 16527 | 913408 | 85246 | 134217728 | 101773 | 135266304 | 792.020745 | 5.0714 | 195.148.124.36:5201 |
| 192.168.1.110:54171 <-> 146.75.82.250:443 | 5511 | 8289280 | 2783 | 162816 | 8294 | 8452096 | 11.344449 | 450.2186 | 146.75.82.250:443 |
| 192.168.1.110:54269 <-> 208.80.154.240:443 | 5420 | 7857152 | 2756 | 221184 | 8176 | 8078336 | 18.438310 | 381.1625 | 208.80.154.240:443 |

1. Compare ping results from active and passive measurements. Provide a table and plot a time series.



|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Ip | Left direction frames | Left direction bytes | Right direction frames | Right direction bytes | Total frames | Total bytes | Start | Duration |
| 192.168.1.110 <-> 195.148.124.36 | 132457 | 7153k | 684989 | 1034M | 817446 | 1041M | 55.755887 | 741.3363 |
| 192.168.1.110 <-> 114.114.114.114 | 4743 | 394k | 6728 | 425k | 11471 | 819k | 0.953963 | 792.6291 |
| 192.168.1.110 <-> 142.93.213.224 | 1738 | 95k | 9248 | 13M | 10986 | 13M | 59.959400 | 391.3487 |
| 192.168.1.110 <-> 146.75.82.250 | 5511 | 8095k | 2783 | 159k | 8294 | 8254k | 11.344449 | 450.2186 |
| 192.168.1.110 <-> 208.80.154.240 | 5420 | 7673k | 2756 | 216k | 8176 | 7889k | 18.438310 | 381.1625 |

In active measurements, the measurer actively generates traffic that may interfere with the network to some extent. This interference may affect the true reflection of network performance. In passive measurements, there may be bias in which flows are selected for observation and analysis. The sample traffic selected may not be representative of the entire network traffic

Comparing iperf results from active and passive measurements, the red line indicates an active measurement, while the blue line indicates a passive measurement. Active measurements introduce a lot of flow, but the duration is short, usually only a few seconds. Passive measurements introduces very little flow, but lasts a long time, almost the entire measurement.

Compare ping results from active and passive measurements, the red line indicates an active measurement, while the blue line indicates a passive measurement. Similarly, passive measurements introduce very little flow and last for a long time, almost the entire measurement. However active measurements also introduce a lot of traffic, but last longer, unlike iperf results.

1. Comparisons of active and passive measurements

|  |  |  |
| --- | --- | --- |
|  | Active measurements | Passive measurements |
| Measurement Tools | Ping, Iperf, Traceroute | CoralReef, Wireshark, tcpdump |
| Measured Features | Latency (RTT), Packet Loss, Bandwidth Utilization, Real-time Performance | Availability, Bandwidth Utilization, Errors, Packet Drops, Traffic Analysis |
| Use of Tools | Ping is used to measure host reachability and latency | CoralReef is used for traffic analysis and extraction of traffic features |
| Iperf is used for measuring bandwidth, throughput, latency and network performance testing | Wireshark is used for packet analysis, troubleshooting, and traffic analysis |
| measuring bandwidth, throughput, and latency | tcpdump is used for packet capture and analysis |
| Issues and Limitations | Active measurement can introduce network interference and affect real performance | Passive measurement requires support from network devices and may not be applicable to all networks |
| Active measurement may not accurately simulate complex real-world traffic | The accuracy of passive measurement is influenced by sample selection and sampling rates |
| Active measurement is affected by network conditions and paths | Passive measurement may not capture encrypted and privacy-protected traffic |