**CSCD58**

**OVERVIEW:**

Tools used:

* Wireshark
  + For collecting dumps of the PCAP file in CSV format
  + For reading the PCAP file
* Python
  + Libraries used:
    - numpy and matplotlib for generating graphs

For additional information, consult README.md for more detailed descriptions of code and usage

CSV dumps:

* univ1\_trace.csv
  + Extracted using Wireshark by “File > Export Packet Dissections > Export as CSV”
  + This is considered to be the “main” CSV
* Modified “dump” CSVs
  + Add additional columns in Wireshark (information from packet headers)
    - Right click on columns and click “Column Preferences”
    - Example – add columns “TCP Header” and “IP Header” with field “tcp.hdr\_len” and “ip\_hdr\_len” respectively

**PER-PACKET STATISTICS – PACKET TYPES:**

Data collection strategy:

* We consider the **PROTOCOL** given from Wireshark to determine the “type” of packet
* Layer distribution is done by grouping the protocols which we discussed in class
  + Transport Layer – TCP, UDP
  + Network Layer – IPv4, ICMPv6, ICMP
  + Link Layer – ARP
  + Everything else is grouped in “Other”

Script overview (**packet\_type.py**):

* Packet dissections is extracted from Wireshark in CSV format
* Script distributes packets into groups based on the criteria defined above

Results:

|  |  |  |
| --- | --- | --- |
| **DISTRIBUTION OF ALL PACKETS** | | |
| **PROTOCOL** | **NUMBER OF PACKETS** | **PACKET SIZE** |
| IGRP | 1016 | 834896 |
| CDP | 6 | 2691 |
| TELNET | 2299 | 1007480 |
| RIP | 28 | 2520 |
| NBSS | 97398 | 100995673 |
| MySQL | 16 | 1898 |
| CVSPSERVER | 380 | 315054 |
| Intel ANS probe | 2002 | 136136 |
| LPD | 195 | 45659 |
| VNC | 1065 | 452653 |
| SSL | 8886 | 5356851 |
| TCP | 477958 | 316114599 |
| ISAKMP | 4323 | 918926 |
| VRRP | 791 | 50624 |
| DHCPv6 | 13 | 1430 |
| LLC | 20894 | 2381796 |
| PIMv0 | 567 | 43092 |
| OSPF | 881 | 283290 |
| NBNS | 4723 | 479700 |
| RSL | 1 | 81 |
| IGMPv0 | 235 | 15040 |
| ESP | 29147 | 19779022 |
| PPTP | 430 | 336763 |
| MDNS | 80 | 8378 |
| SMTP | 761 | 852870 |
| MS NLB | 1340 | 1829040 |
| 0x200e | 14 | 1232 |
| Gryphon | 116 | 22320 |
| SSH | 7453 | 3245383 |
| DNS | 33854 | 5734077 |
| DSI | 75 | 6500 |
| LLMNR | 144 | 10656 |
| ICMP | 22552 | 1449590 |
| ICMPv6 | 2 | 244 |
| ARP | 65594 | 4200316 |
| UDP | 191650 | 134281370 |
| NBDS | 527 | 145071 |
| NCP | 26 | 3419 |
| NTP | 219 | 20586 |
| NCS | 1214 | 118972 |
| BOOTP | 70 | 32803 |
| Syslog | 4686 | 761994 |
| GRE | 39 | 4634 |
| UDPENCAP | 36 | 2304 |
| IPv4 | 2362 | 1018776 |
| SRVLOC | 13 | 1079 |
| IPX | 6 | 384 |

|  |  |  |  |
| --- | --- | --- | --- |
| **LAYER DISTRIBUTION** | | | |
| **LAYER** | **NUMBER OF PACKETS** | **PERCENTAGE** | **NUMBER OF BYTES** |
| Link Layer | 65594 | 6.65% | 4200316 |
| Network Layer | 24916 | 2.53% | 2468610 |
| Transport Layer | 669608 | 67.91% | 450395969 |
| Other | 225969 | 22.92% | 146242977 |

Analysis:

The vast majority of packets are sent with some type of transport protocol. The Link and Network protocols are associated with less than 10% of the total packets. A small but interesting detail is how the Network layer’s byte composition from packets is greater than half of the Link layer’s composition, despite being less than half of the Link layer’s representation signifying more bytes transferred per packet.

**PER-PACKET STATISTICS – SIZE OF PACKETS:**

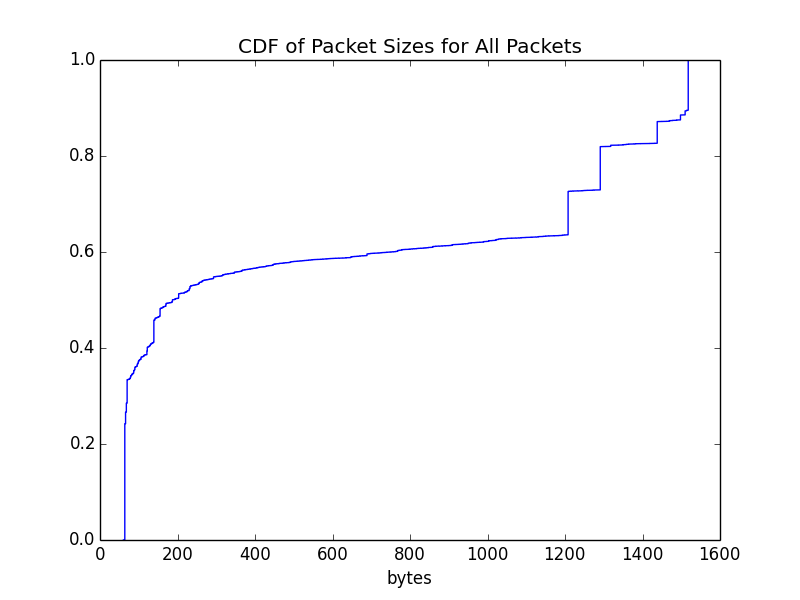
Data collection strategy:

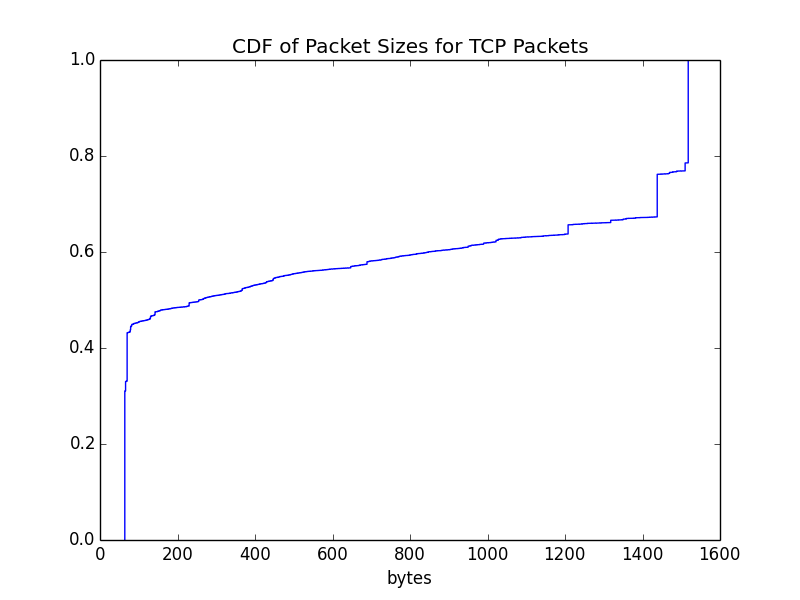
* Using Wireshark, we add the field “tcp.len” to the CSV
  + This is the payload length noted in the TCP header
    - Adding this column to the CSV is described in the (**Overview**) section of this report
* We separate the four groups as follows:
  + TCP packets – packets using the TCP protocol
  + UDP packets – packets using the UDP protocol
  + IP packets – packets using either TCP or UDP
  + Non-IP packets – anything else
* Header calculations are done as follows:
  + TCP header = frame size – payload size
  + UDP header = 8 (UDP headers are fixed at 8 bytes)
  + IP header – calculations are same as above

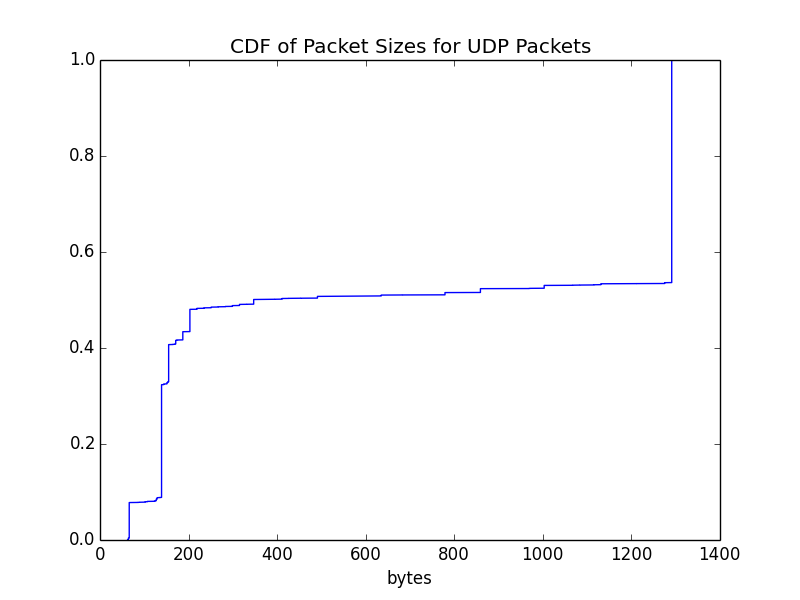
Script overview (**packet\_size.py**):

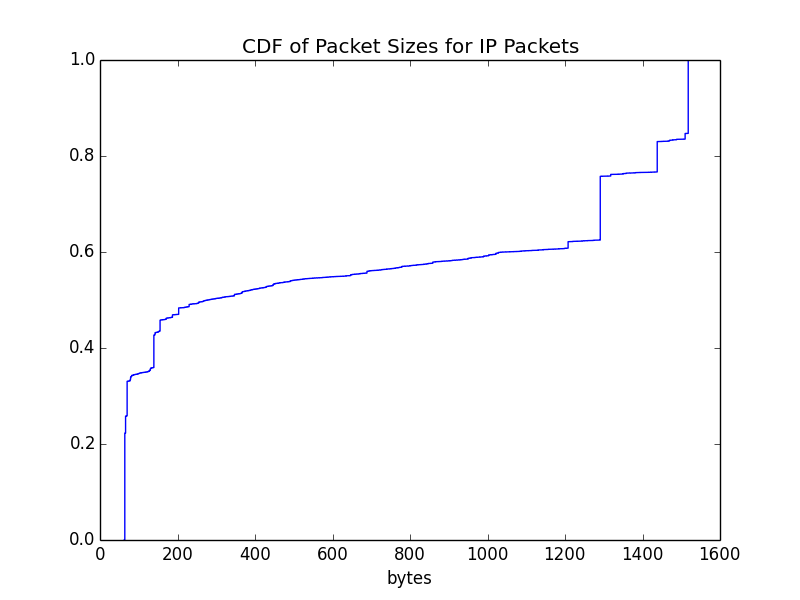
* Packet dissections extracted from Wireshark in CSV format
* Script collects packet lengths and header lengths (of appropriate packets) according to criteria defined above
* Plots CDF graphs

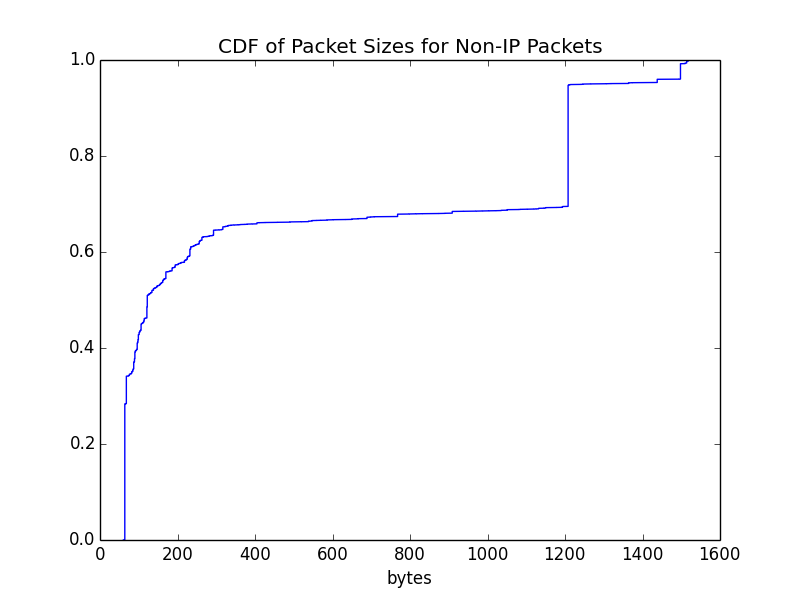
Packet size results:











Analysis:

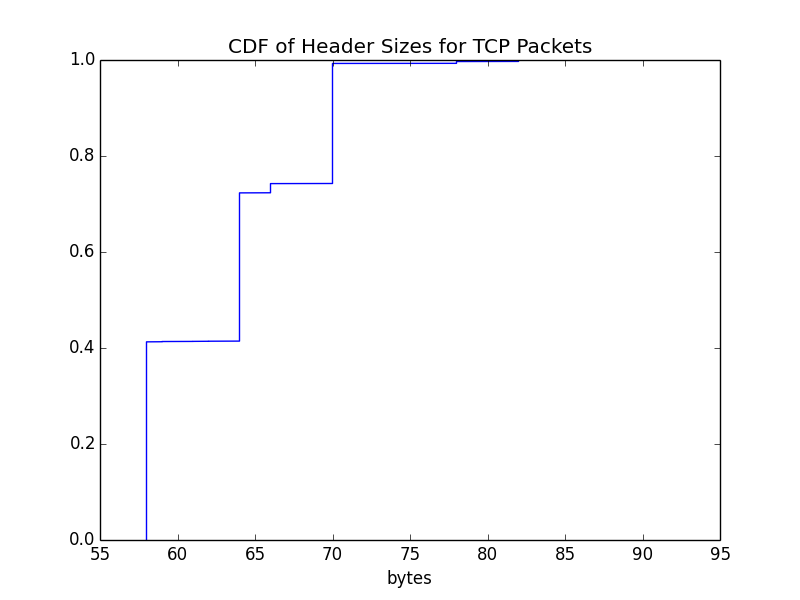
It’s worth noting that based on our total packet CDF, about half of all the packets captured and analyzed were 200 bytes or less. This trend of small packet sizes is apparent in all the individual graphs as well.

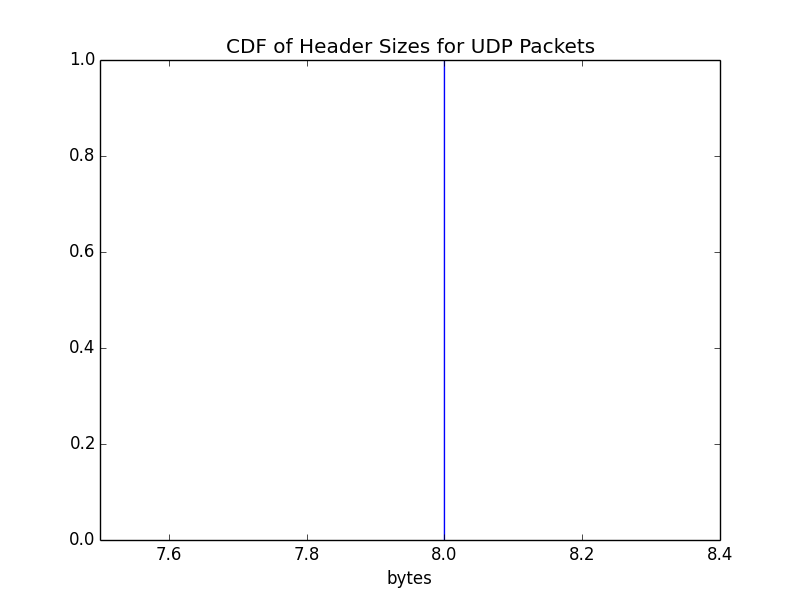
Another similar observation is that the other major percentage of packets are greater than or equal to 1200 bytes. Again, this persists across all the various protocols.

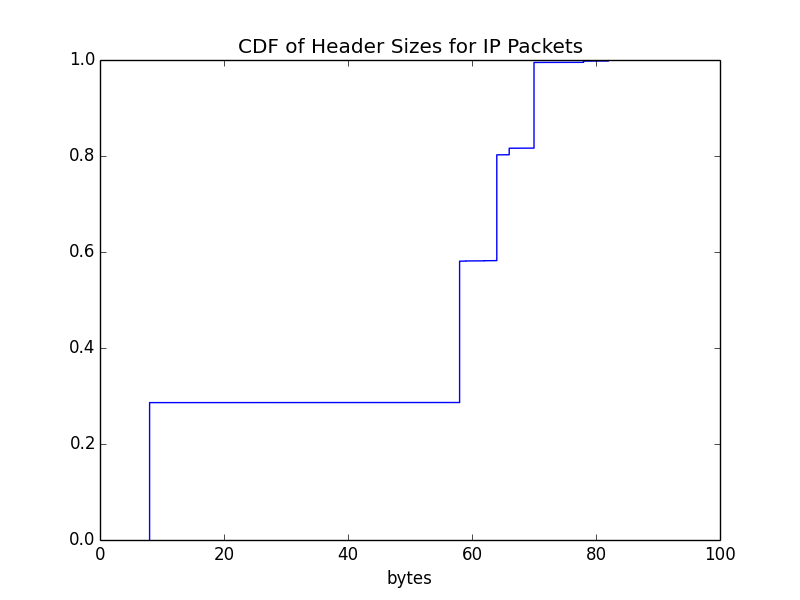
One major difference is after the 1200 byte mark, there is a sudden noticeable difference among the distributions of packets. For example, the UDP and Non-IP packets take sudden climbs towards 100% whereas the TCP and IP packets show a more gradual, stepping climb towards the total mark.

This stands to show there are some major similarities in packet size distributions across the various protocols when at or under 1200 bytes. Beyond that, it becomes easier to identify and categorize each protocol’s distribution style.

Header size results:







Analysis:

In general, TCP packets seem to contribute to the majority of the packet size which can be seen with how similar the CDF of TCP and all packets are. In addition to this, TCP packets seem to make up the most of IP packets as well.

TCP packets are also generally larger than UDP packets in terms of length. This is most likely due to the fact that UDP packets have a fixed header length of 8 bytes and TCP packets have a minimum header length of 20 bytes and a maximum length of 60 bytes. With the headers alone, TCP is larger than UDP and with an additional payload the byte length will only rise.

The header length of UDP packets is uninteresting as UDP packets have a fixed header length of 8 bytes. On the other hand, TCP packets have much more variance in their header lengths since there are opportunities to add optional options in the header. As a result, the majority of the variance in the CDF of IP headers stems from the variance in the TCP header length.

**FLOW TYPES:**

Script overview (**flow\_count.py**):

* Define TCP/UDP flows as a set of packets that have the same:
  + Source / Destination IP
    - Or this pair reversed
  + Source / Destination Port
  + Protocol
* We do not consider maximum packet inter-arrival time being less than 90 minutes since the packet inter-arrival time between all packets is around 3 minutes
* Script collects and counts all groups of packets of these criteria and returns the result

Results:

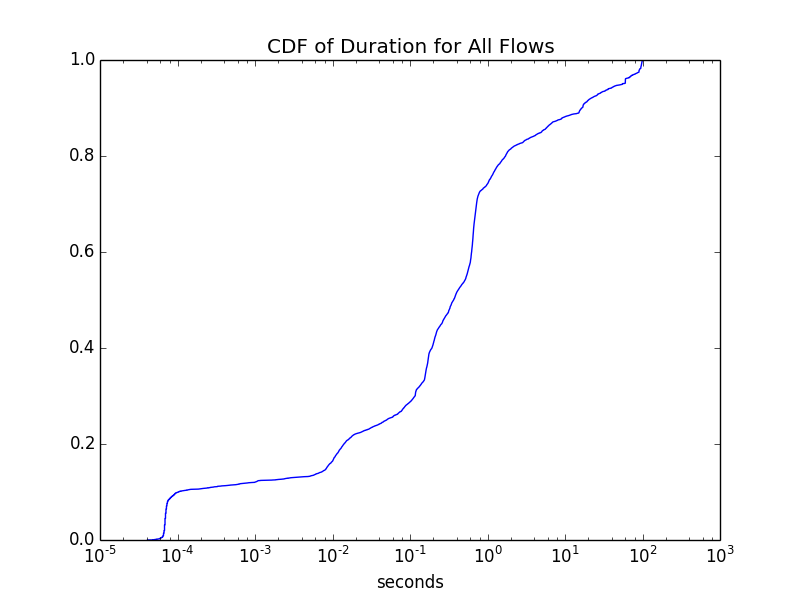
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **FLOW COUNT** | | | | |
| **PROTOCOL** | **# OF FLOWS** | **% OF TOTAL** | **PACKETS IN FLOW** | **BYTES IN FLOW** |
| TCP | 8825 | 96.51% | 477958 | 316114599 |
| UDP | 319 | 3.49% | 191650 | 134281370 |

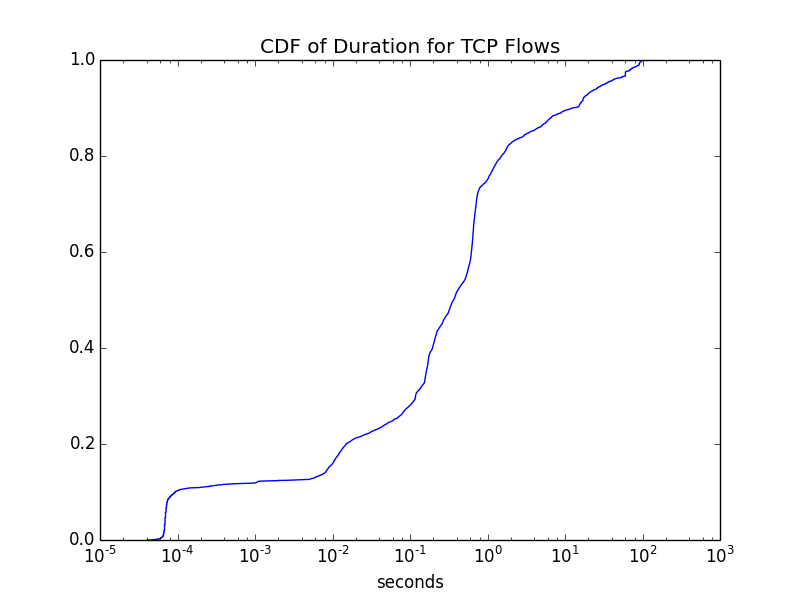
**FLOW DURATION:**

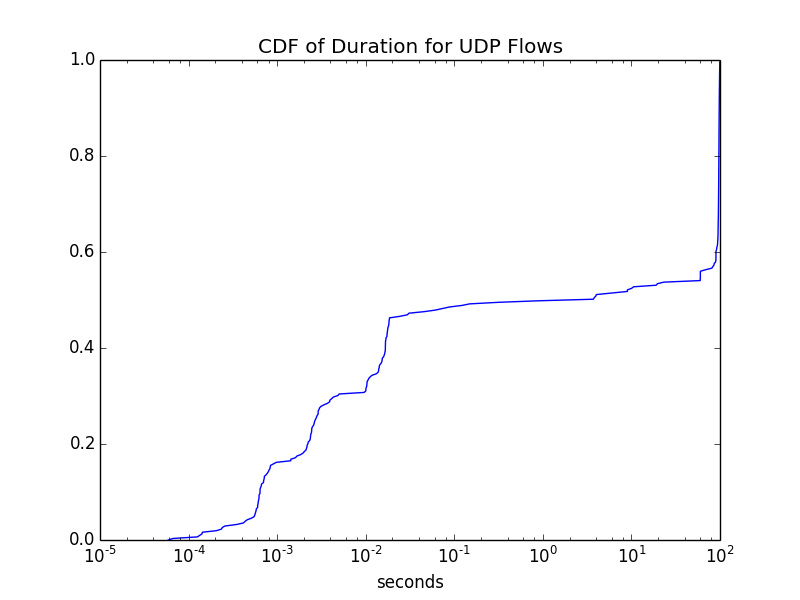
Script overview (**flow\_duration.py**):

* Using the same flow finding algorithm as seen in (**flow\_count.py**), compute all of the TCP/UDP flows
* Extract all of the durations of the flows (arrival time of last packet minus arrival time of the first packet)
  + We don’t consider flows that have the same final arrival and initial arrival time as we consider those such flows as being a single packet sent
* Separate the flows by protocol

Results:







Analysis:

We choose a logarithmic scale for the x-axis for these charts because there is a skewness of the variables. There are many values clustered at both extremes of the CDFs so a logarithmic scale can make it easier to portray the subtleties of the CDF.

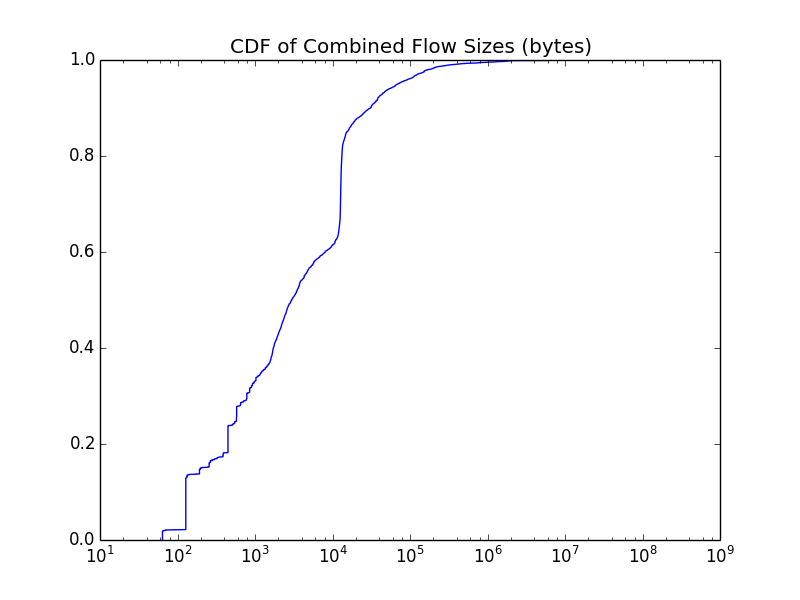
In general, UDP flows tend to have shorter durations in comparison to TCP flows. This can be evidently seen as the probability of a UDP flow to have a duration of around seconds is much higher than a TCP flow being of the same duration. The reasoning behind this might be because a TCP flow requires acknowledgement whereas UDP can continually send without receiving any acknowledgement from the sender which may result in a lower duration.

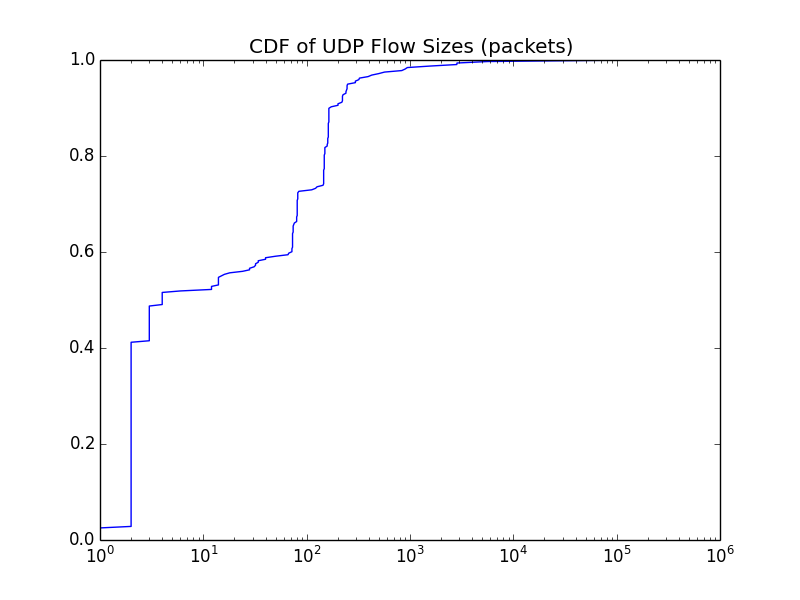
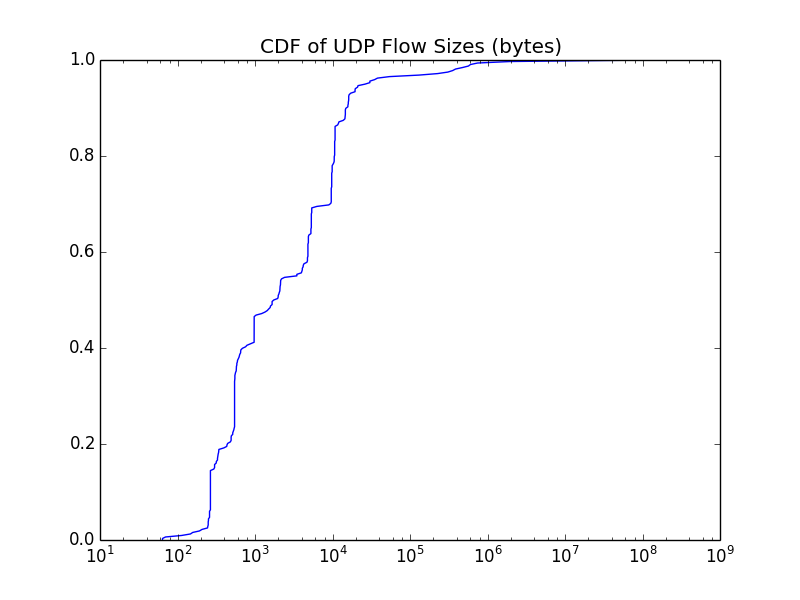
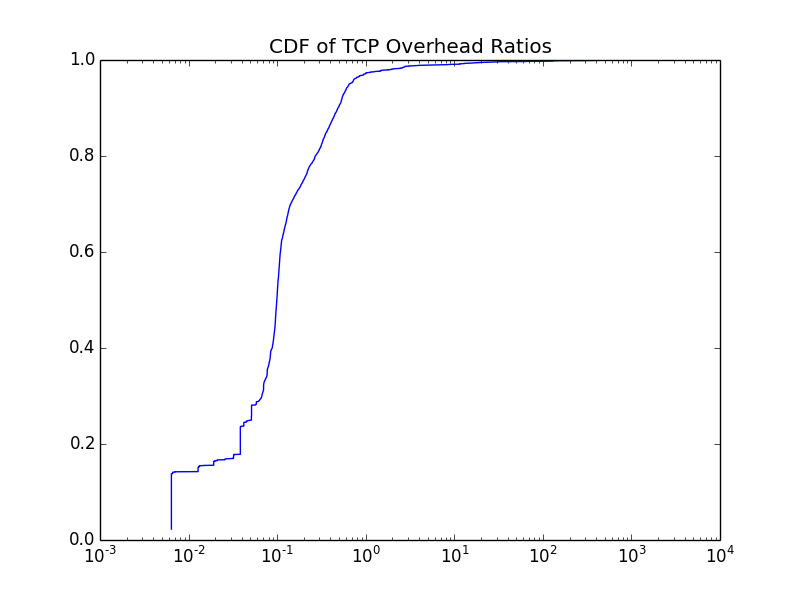
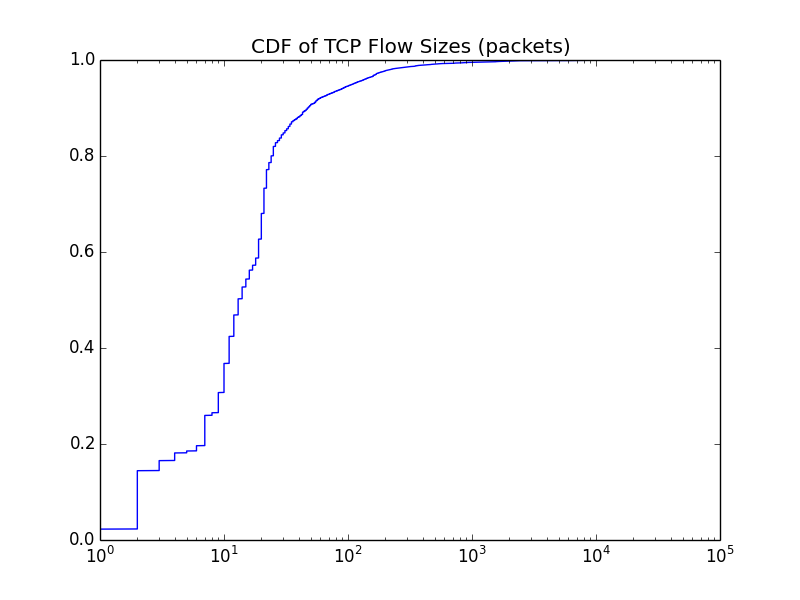
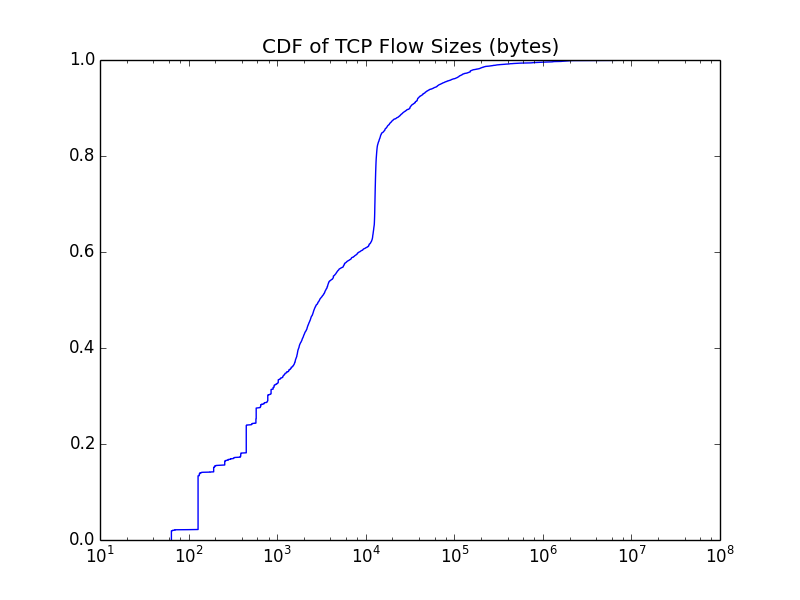
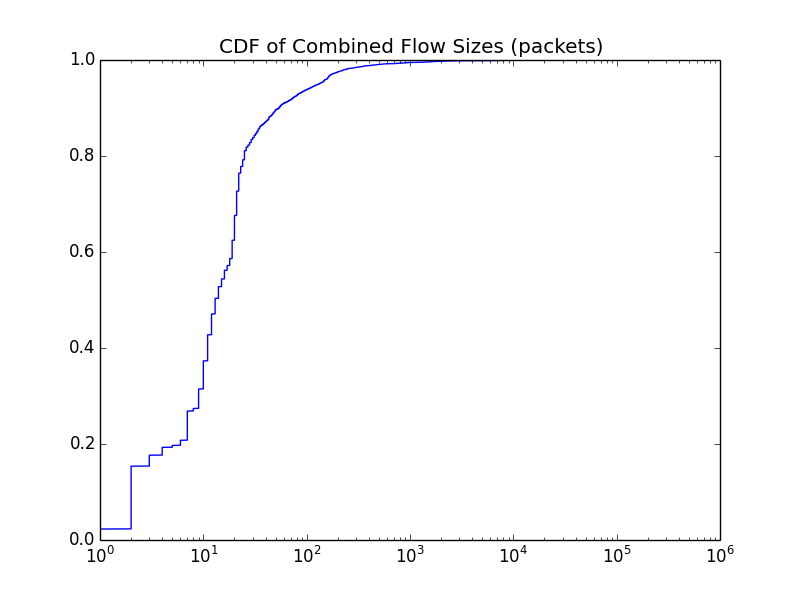
**FLOW SIZE:**

Script overview (**flow\_size.py**):

* Using the same flow finding algorithm as seen in (**flow\_count.py**), compute all of the TCP/UDP flows
* Extract the necessary information and calculate the TCP overhead ratio

Results:





Analysis:

**What is the difference between TCP and UDP?**

At a glance, there doesn’t appear to be much variation in between the two graphs. Both TCP and UDP appear to have approximately 60% of their flow sizes fall at or under 10000 bytes and they have a similar minimum observed flow size at around 100 bytes. The most noticeable difference is in their flow sizes by packets. About 50% of UDP flows have at least 10 packets, whereas about 20% of TCP flows have at least that many. What does this mean? That UDP flows are generally smaller when it comes to packet count, but because the byte sizes of the two flows match so similarly it can be surmised that the UDP flows generally send more bytes per payload than their TCP counterparts.

**What is the difference between packet count vs byte sum?**

See the above conclusion. Expect a more gradual change in sizes through byte sum but a more sudden shift from packet count.

**What can you say about the TCP overhead base on the chart?**

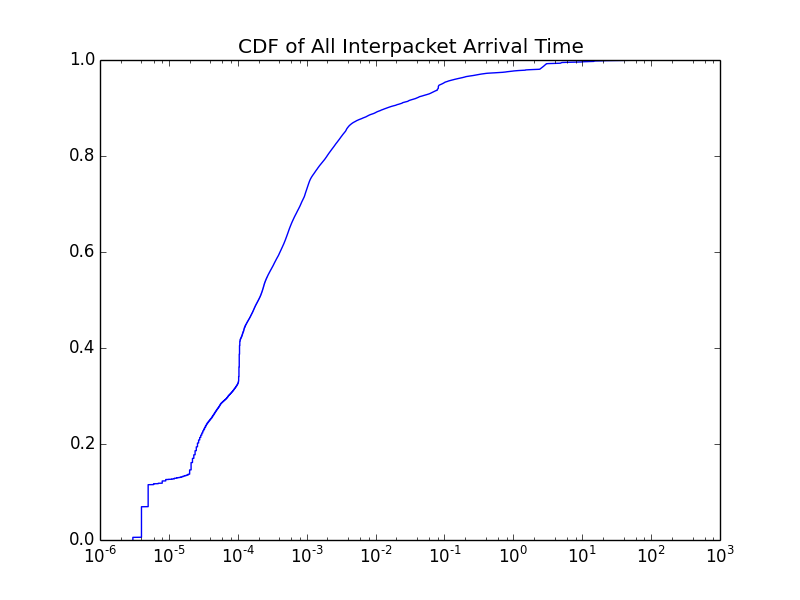
Most of the overhead ratios fall close to or under 1. However, there is a small percentage of packets (approximately 5%, give or take) that have overheads significant enough to create a ratio greater than 1. These two statistics mean that most of the packets seem to have an equal ratio of unused to used space whereas a small group of packets aren’t using their capacities wisely. Fortunately, most of the TCP packets have a small overhead ratio of 0.1 or less, meaning optimality isn’t completely lost just yet.

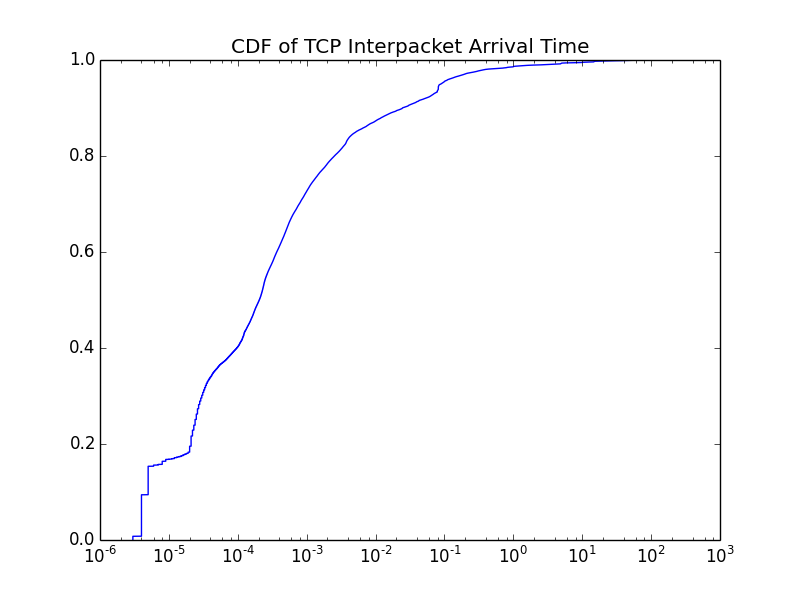
**INTER-PACKET ARRIVAL TIME:**

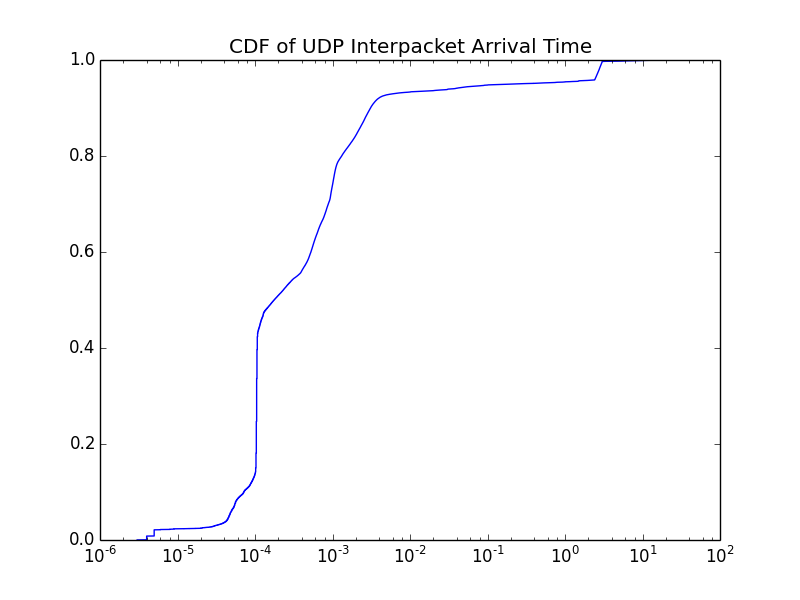
Script overview (**ipat.py**):

* Using the same flow finding algorithm as seen in (**flow\_count.py**), compute all of the TCP/UDP flows
* Extract the necessary information and graph the CDF

Results:







Analysis:

**Is there any specific inter-arrival time that appears more commonly?**

The most commonly appearing specific inter-arrival times are:

* In all flows – 0.001681 seconds
* In TCP flows – 0.001681 seconds
* In UDP flows – 0.015625 seconds

**Do you see any difference between TCP and UDP flows?**

For both UDP and TCP, all of the inter-packet arrival times are very likely to take 1 second or less. However, for TCP flows, it is more likely that two packets will have a significantly lower inter-packet arrival time in comparison to a UDP flow albeit not by a significant margin.

**TCP STATE:**

Script overview (**tcp\_state.py**):

* Using the same flow finding algorithm as seen in (**flow\_count.py**), compute all of the TCP flows
* Extract the necessary information

Results

|  |  |
| --- | --- |
| **TCP STATE** | |
| **COUNT TYPE** | **COUNT** |
| Reset | 3644 |
| Finished | 4246 |
| Request | 103 |
| Other | 7927 |

Analysis:

Given the times found in the PCAP, there was no flow time that was greater than 5 minutes. Thus, we could not properly define an “Ongoing” or even a “Failed” TCP state. These results were stored under “Other” and make up majority of final TCP states

It is interesting to see the amount of “Reset” TCP states in this data set as well as the amount of TCP flows that managed to finish.

**RTT ESTIMATION:**

Largest flows (packet):

* 244.3.176.224:445 -> 244.3.153.247:4174
* 68.157.168.194:51986 -> 41.177.26.91:80
* 68.157.168.194:35468 -> 41.177.26.91:80

Largest flows (bytes):

* 68.157.168.194:51986 -> 41.177.26.91:80
* 68.157.168.194:35468 -> 41.177.26.91:80
* 210.197.167.105:3237 -> 41.177.26.15:80

Largest flows (duration):

* 244.3.176.224:445 -> 244.3.153.247:4174
* 210.197.167.105:3237 -> 41.177.26.15:80
* 244.3.47.39:3260 -> 41.177.31.94:49240

NOTE – there is a considerable overlap in the largest flow types

Regarding the sample RTTs:

* The sample RTTs are not stable and fluctuate a lot
  + There is a lot of increase and decrease constantly of RTT over time
* The fluctuation in RTT is seemingly random
  + No apparent pattern

Regarding the estimated RTTs:

* The estimated RTTs are very stable and do not fluctuate a lot
* Reason for difference in result:
  + Error margin in calculation

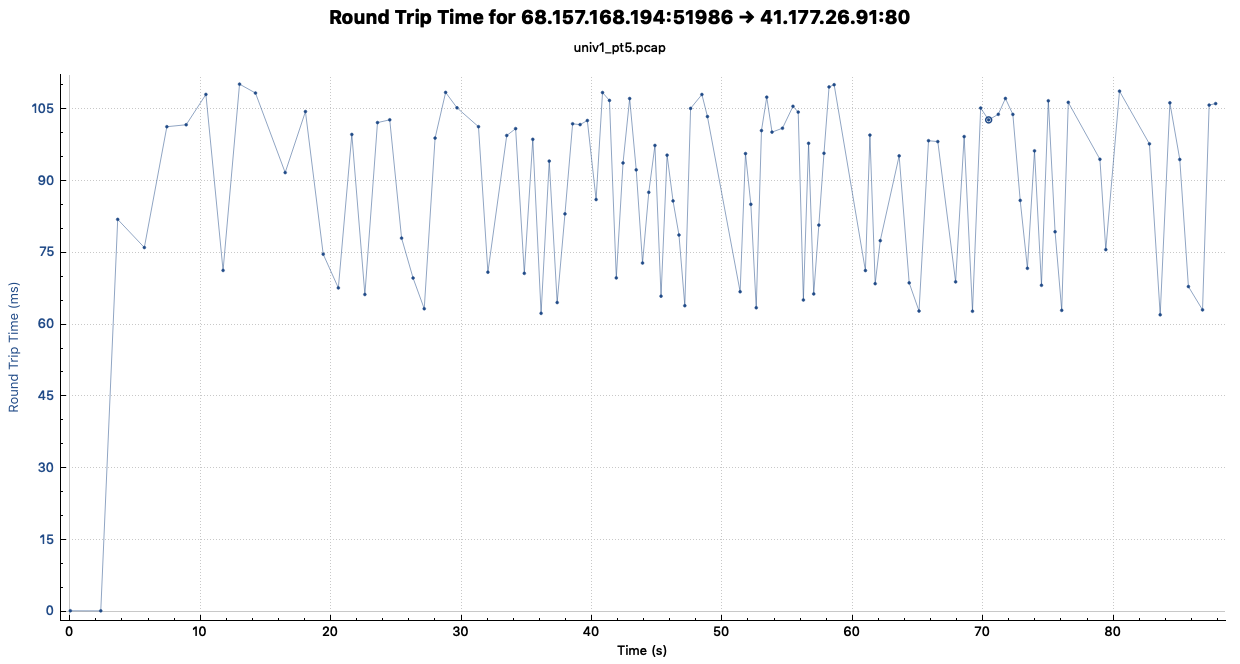
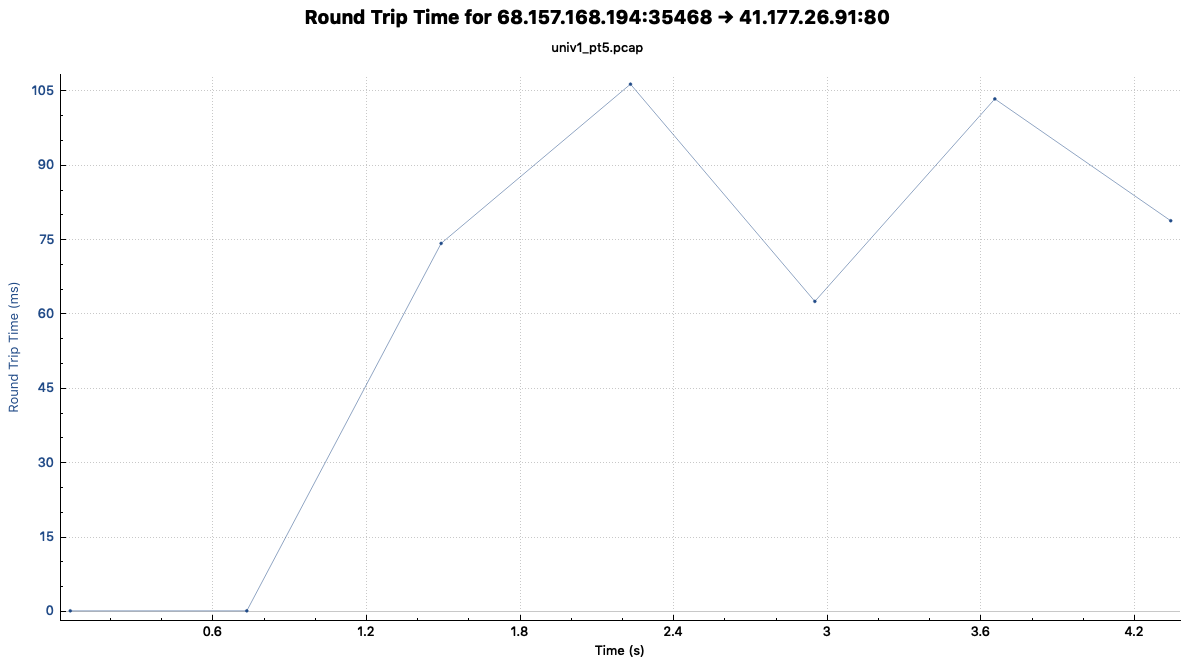
Factors that can affect the RTT:

* Change in best pathway between the two hosts
* Change in TCP window size

The top three host pairs are:

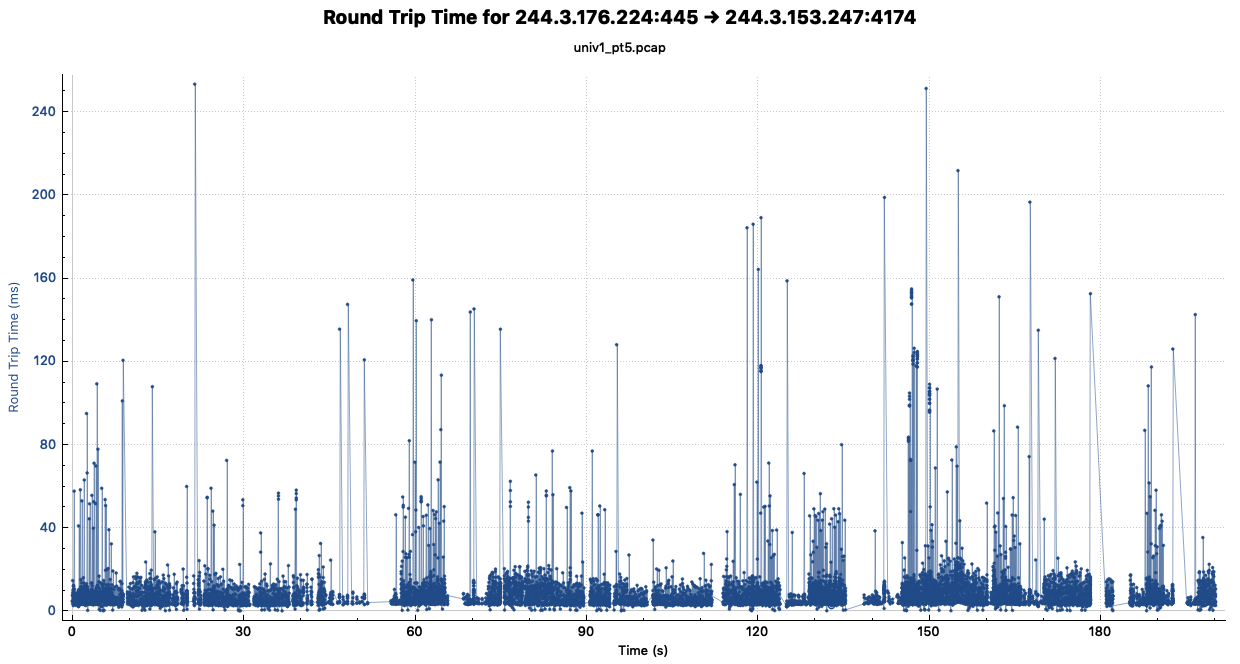
* 244.3.176.224:445 -> 244.3.153.247:4174
* 68.157.168.194:51986 -> 41.177.26.91:80
* 210.197.167.105:3237 -> 41.177.26.15:80

Largest flows (packet):



Largest flows (bytes):





Largest flows (duration):

