**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 |

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

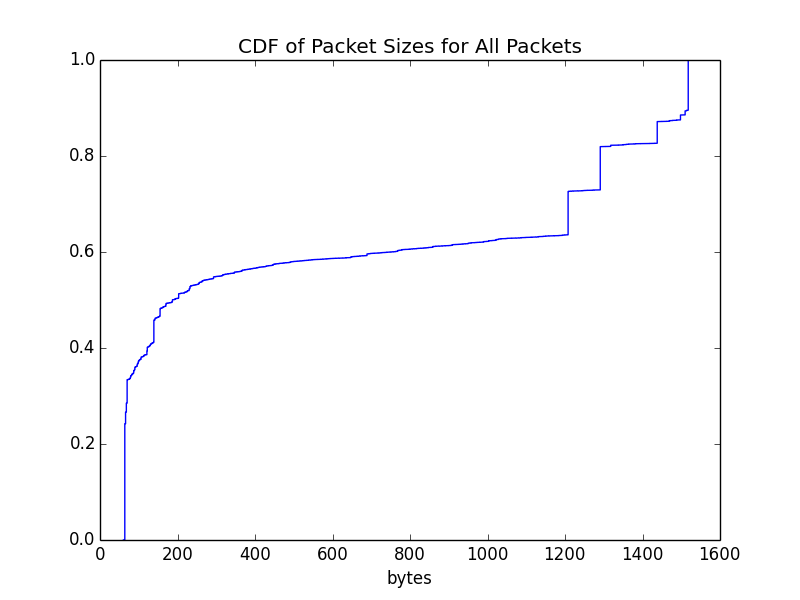
Data collection strategy:

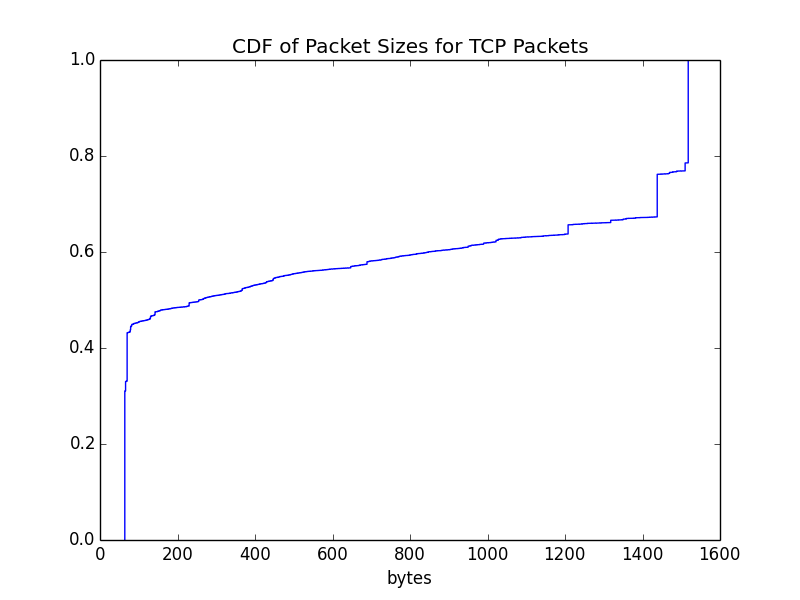
* Using Wireshark, we add the field “tcp.len” to the CSV
  + This is the payload length noted in the TCP header
    - Adding this column to the CSV is described in the (**Overview**) section of this report
* We separate the four groups as follows:
  + TCP packets – packets using the TCP protocol
  + 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

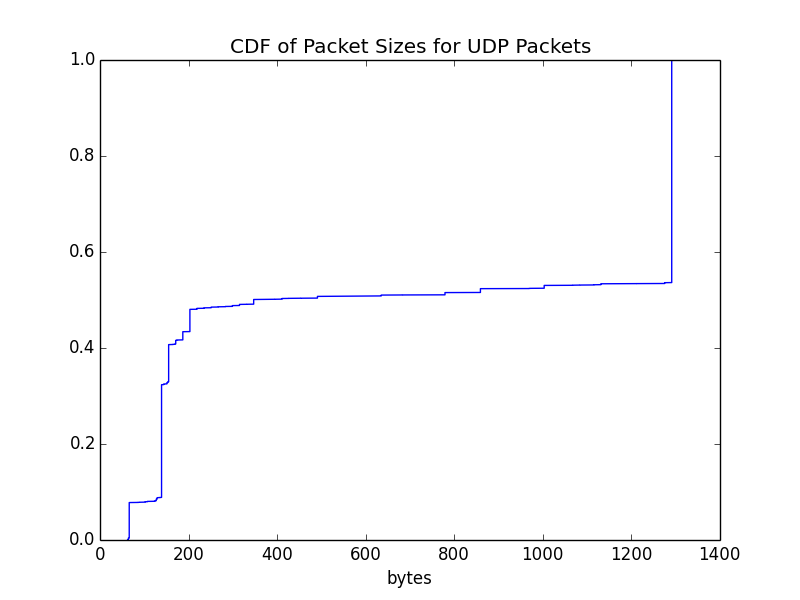
Script overview (**packet\_size.py**):

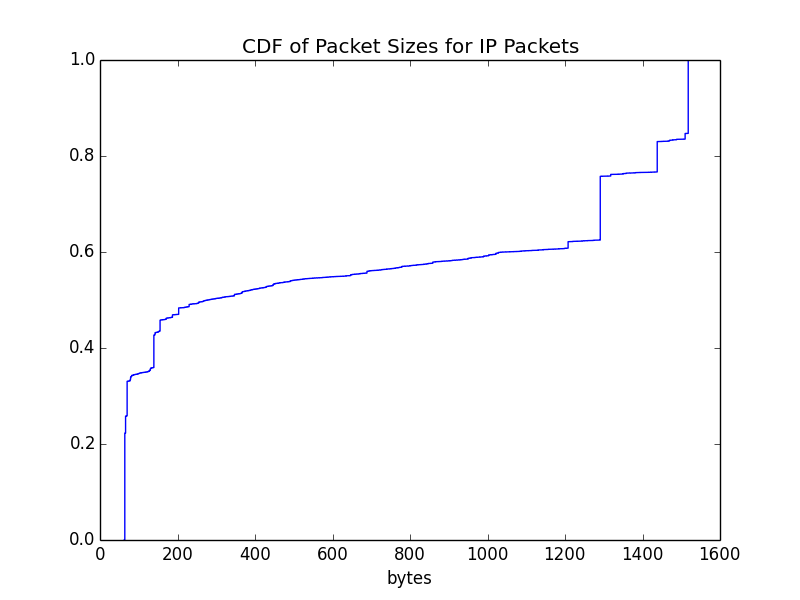
* Packet dissections extracted from Wireshark in CSV format
* Script collects packet lengths and header lengths (of appropriate packets) according to criteria defined above
* Plots CDF graphs

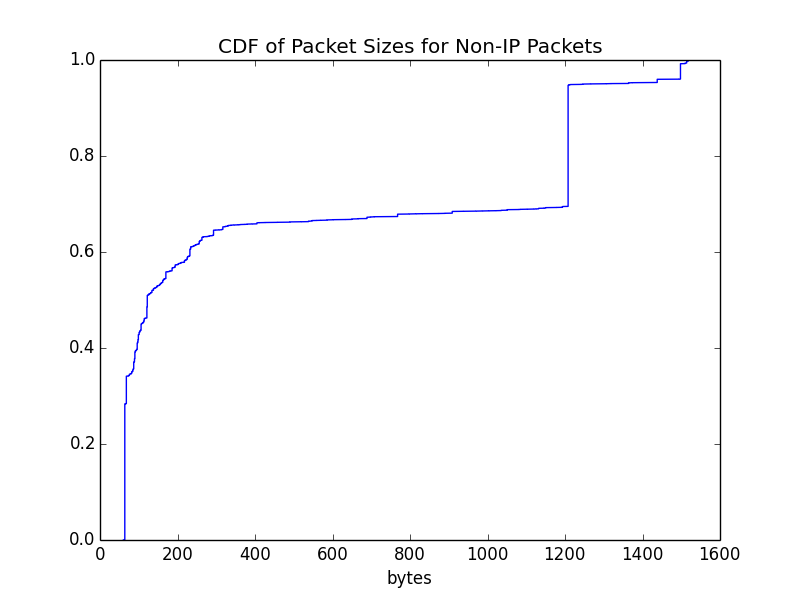
Packet size results:



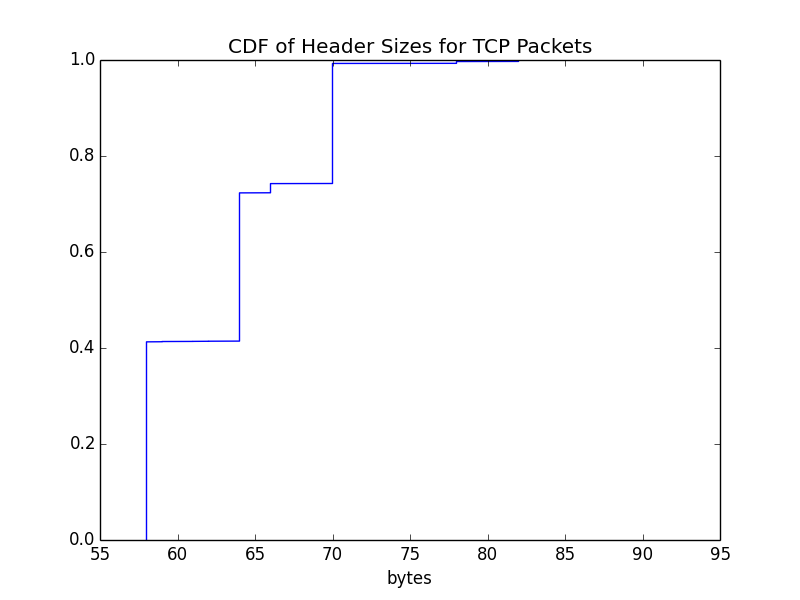


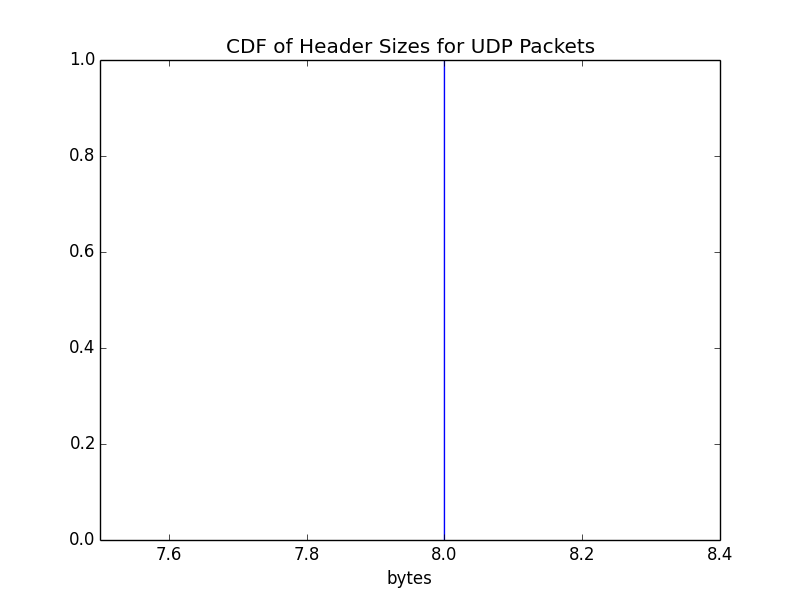


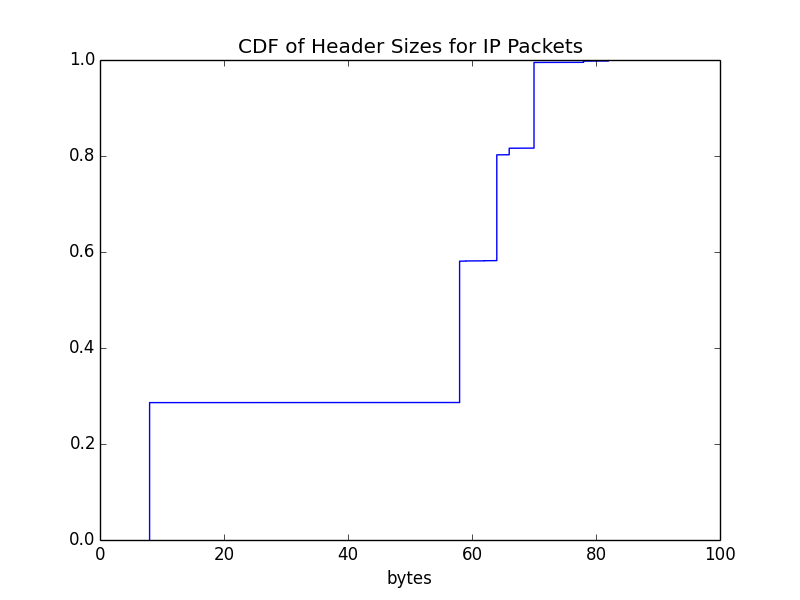




Header size results:







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:

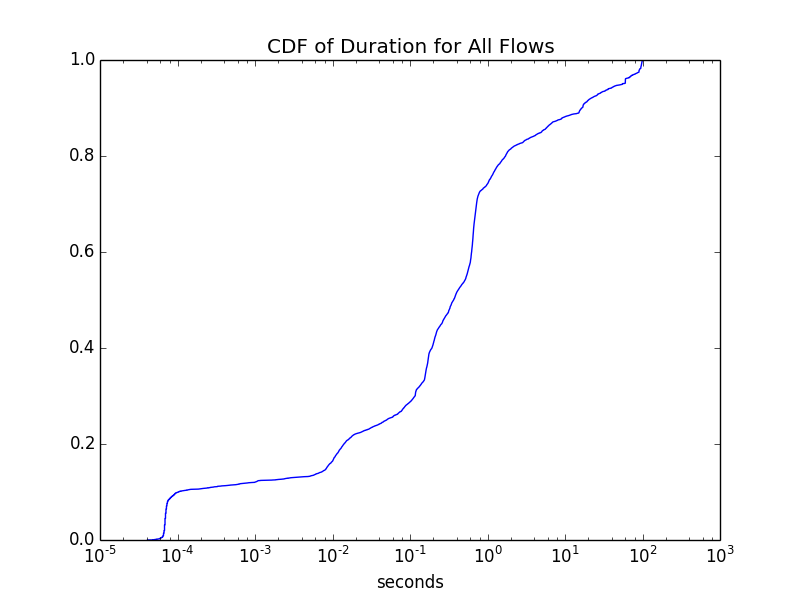
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **FLOW COUNT** | | | | |
| **PROTOCOL** | **# OF FLOWS** | **% OF TOTAL** | **PACKETS IN FLOW** | **BYTES IN FLOW** |
| TCP | 8825 | 96.51% | 477958 | 316114599 |
| UDP | 319 | 3.49% | 191650 | 134281370 |

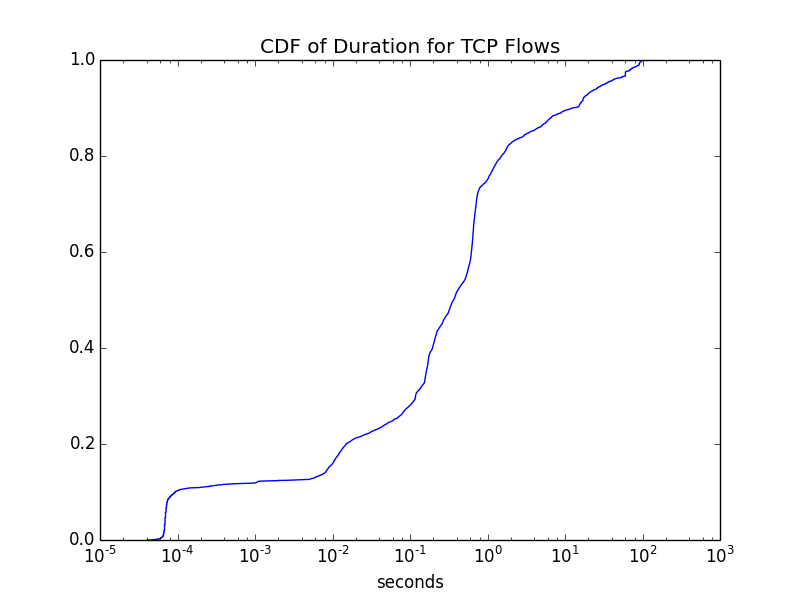
**FLOW DURATION:**

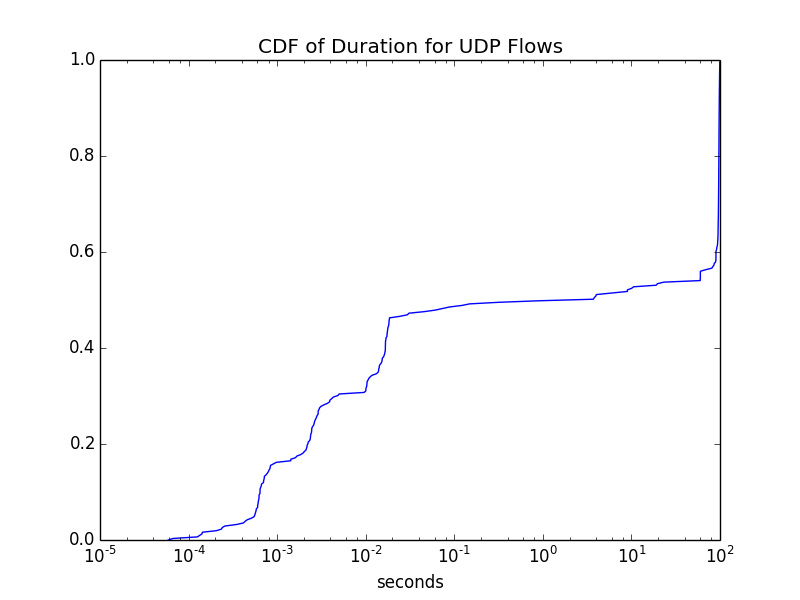
Script overview (**flow\_duration.py**):

* 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.

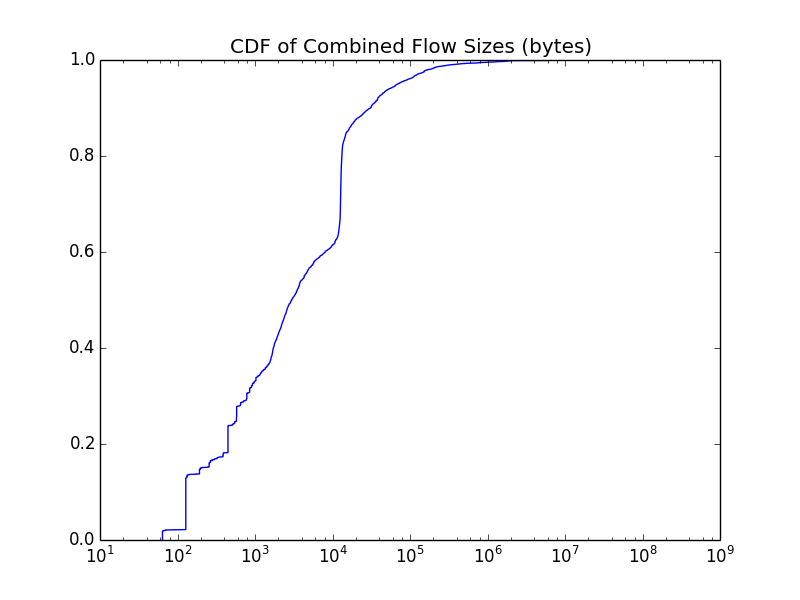
In general, UDP flows tend to have shorter durations in comparison to TCP flows. This can be evidently seen as the probability of a UDP flow to have a duration of around seconds is much higher than a TCP flow being of the same duration. The reasoning behind this might be because a TCP flow requires acknowledgement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

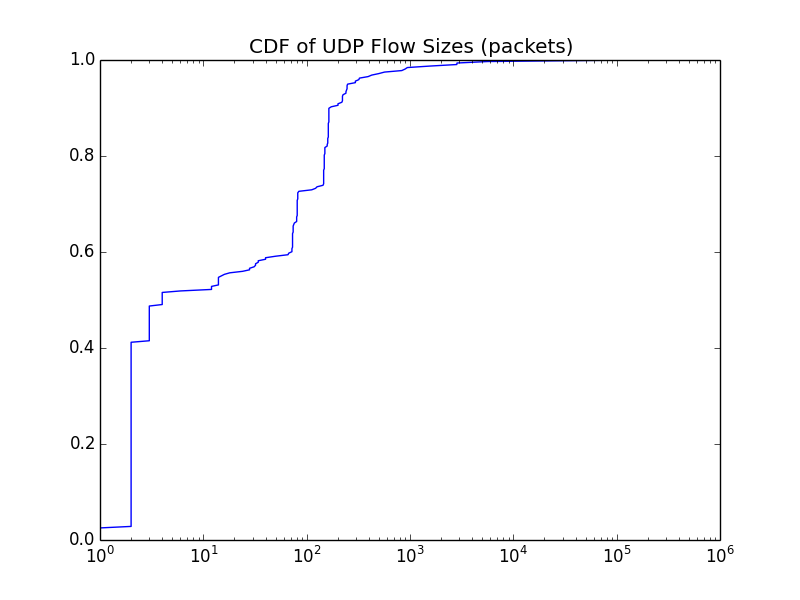
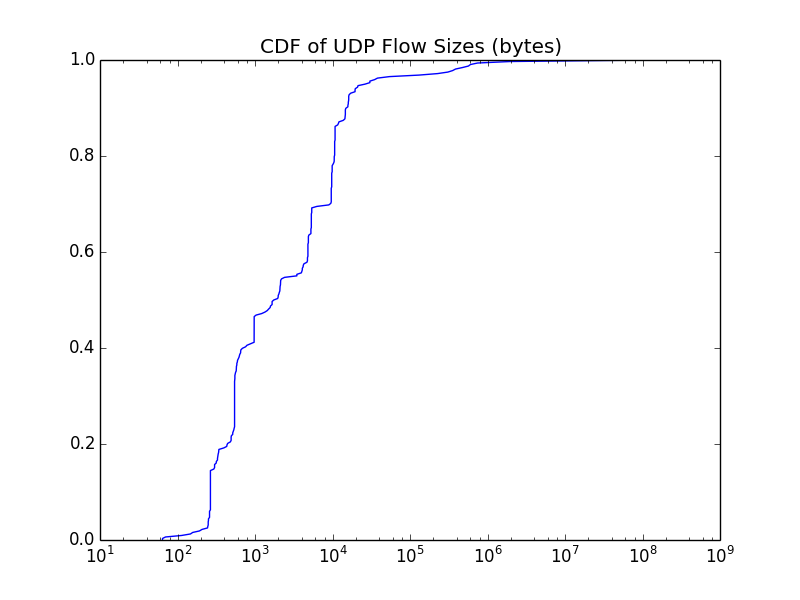
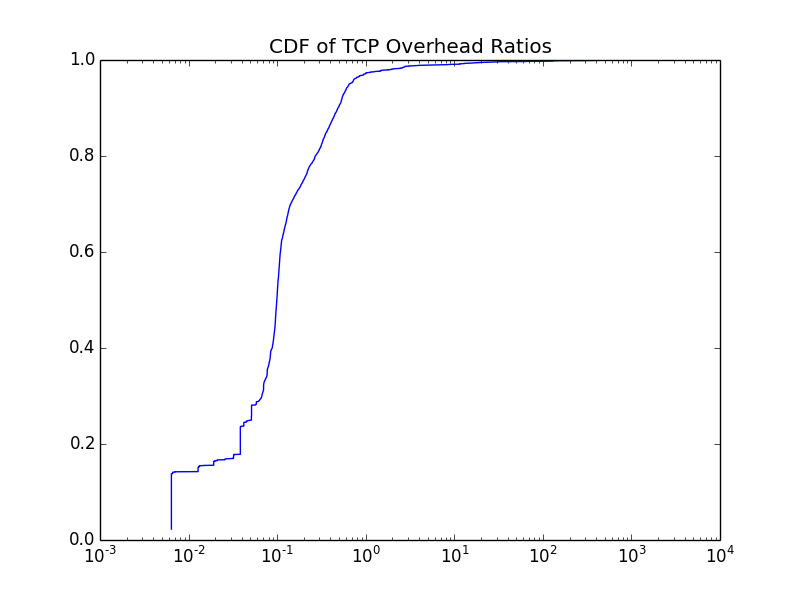
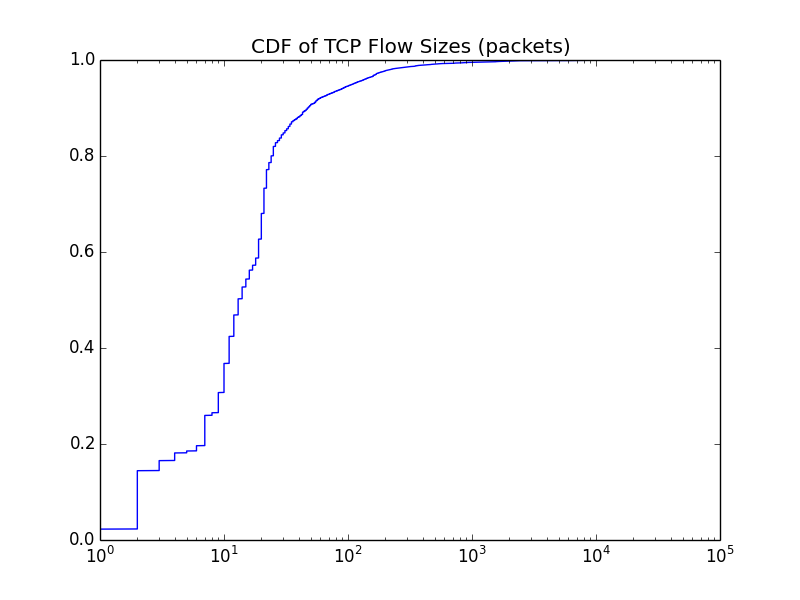
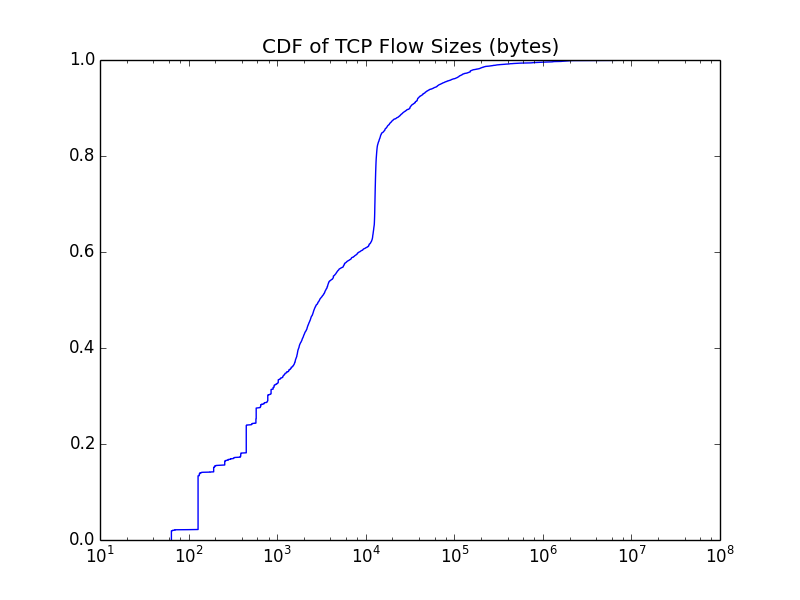
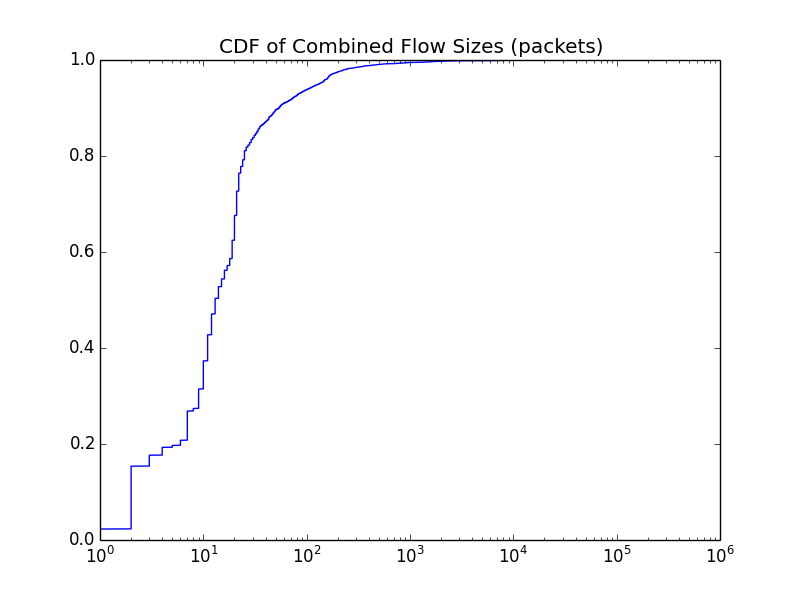
**FLOW SIZE:**

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?

What is the difference between packet count vs byte sum?

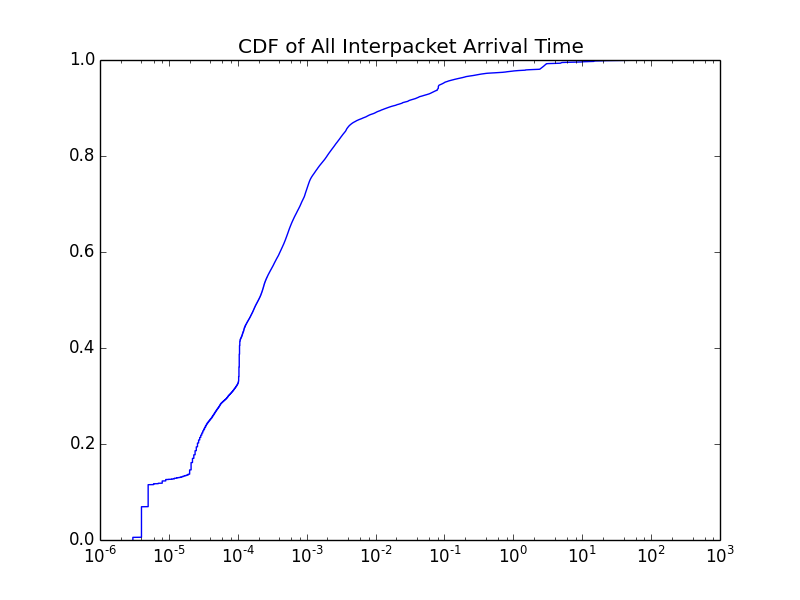
What can you say about the TCP overhead base on the chart?

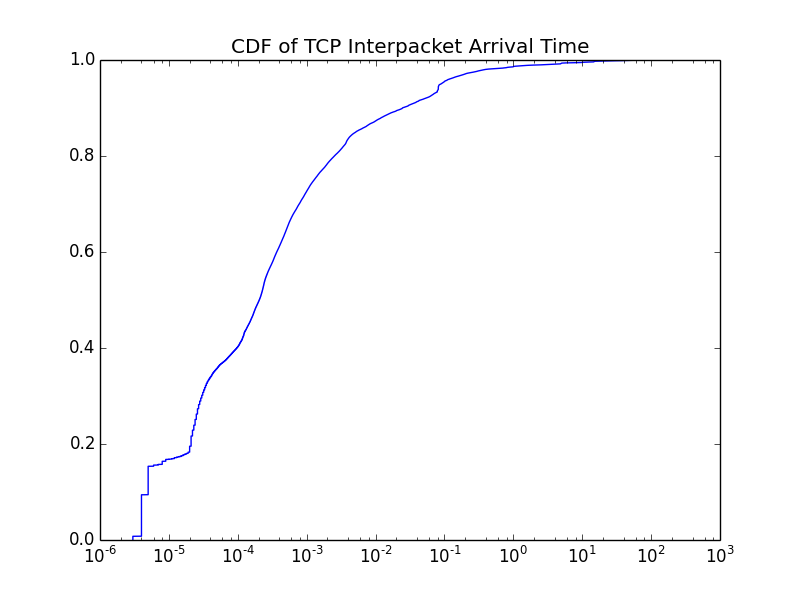
**INTER-PACKET ARRIVAL TIME:**

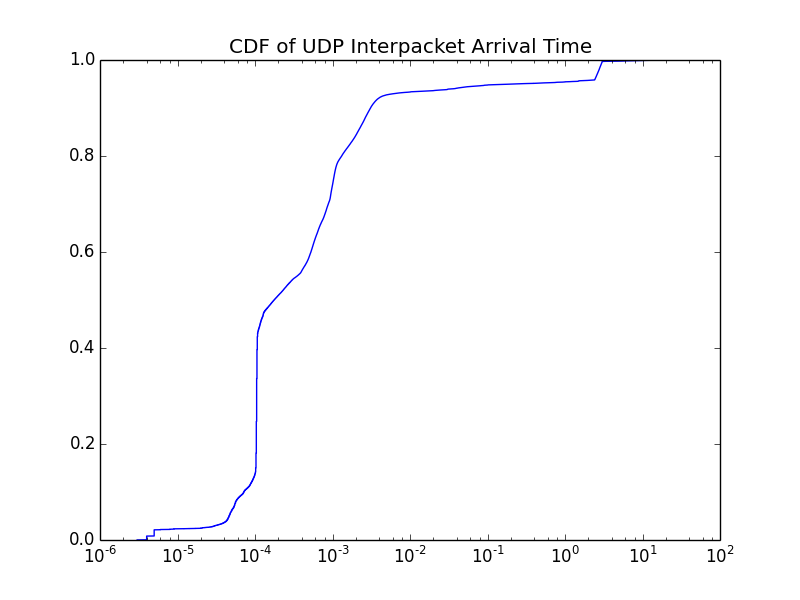
Script overview (**ipat.py**):

* 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?

* If yes, is it present in:
  + All flows?
  + TCP flows?
  + UDP flows

Do you see any difference between TCP and UDP flows?

**TCP STATE:**

reset count: 3644

fin count: 4246

request count: 103

other count: 7927