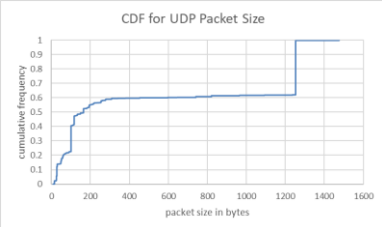
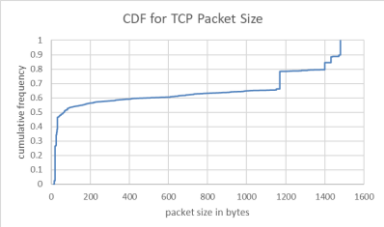
Project Report

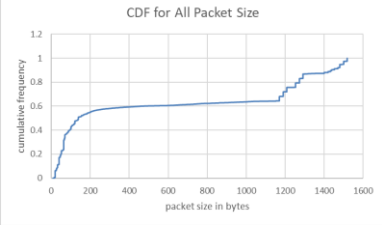
For the following report, all statistics about the packets, flows, and round-trip time estimations are extracted using the actual size of the packets (not capture length) in the trace file *univ1_pt9*, by using python and the libraries *dpkt* and *xlsxwriter*. Furthermore, all the appealing charts are produced by Excel to visualize the information and properties of the trace file. Note some of the CDFs are modify so that outliers are not seen.

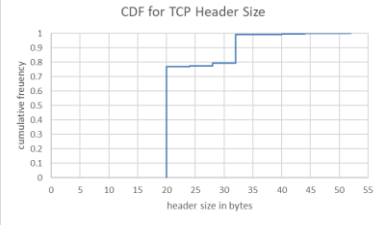
To analysis the general statistics about the traffic inside the trace file, the following information provides a good overview of that. From the table below, not surprisingly, the ethernet packets occupies 100% the link layer since it is mostly commonly used technology in local area network. Out of all ethernet packets, IPv4 datagram occupies 89.32% of it but the number of IPv6 datagrams in the trace file is so small that we can even neglect its count; ICMP and ARP packets are expected to occupies portion of the traffic since most likely there are errors sent by the routers and host may need to send ARP packets to get the mapping between IP address and MAC address. Lastly, out of all the network layer packets, most of encapsulated payloads are TCP segments (73%) and UDP segments occupies much less (23.45%), it implies that most of the host uses application that requires reliable services to transmit the data.

Type of Packet Statistics	Packet Counts	Percentage	Total bytes
	LINK LAYER		
Ethernet	998830	100%	580480828
	NETWORK LAYER		
IPv4	892187	89.32%	554555037
IPv6	15	0.000015%	1404
ICMP	24036	2.41%	4874069
ARP	61899	6.20%	2850344
Others	20693	2.07%	2895332
	TRANSPORT LAYER		
TCP	731620	73.24%	392888882
UDP	234257	23.45%	129803376
Others	32953	3.31%	18507813

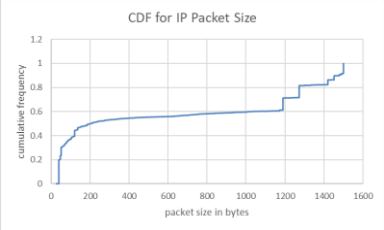


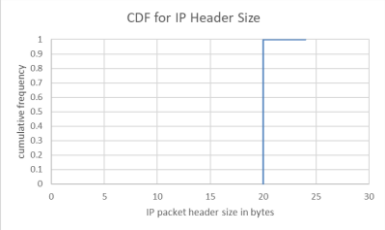


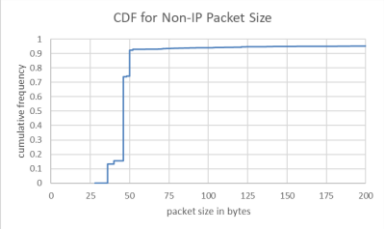


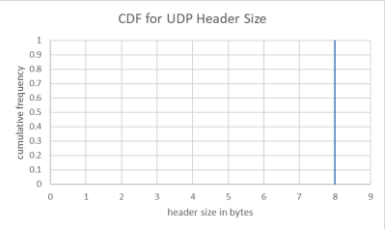


To analyze the traffic further, over all packet sizes in the trace file, around 60% of the packets with size 600 bytes but worth to notice that 40% of packets have size more than 1KB. Specifically, both UDP and TCP have common packet size of 600 bytes, but they have different common payload size; the overall payload size of UDP packets is much larger that of TCP packets since the header of UDP packet is fxied to 8 bytes and TCP



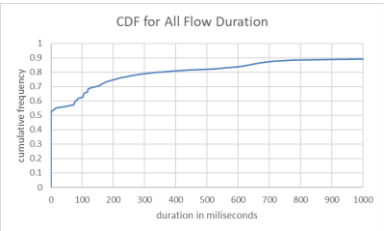


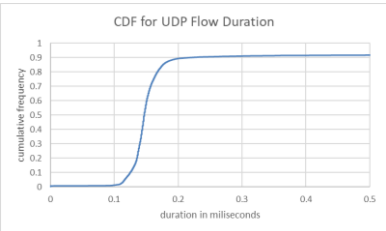


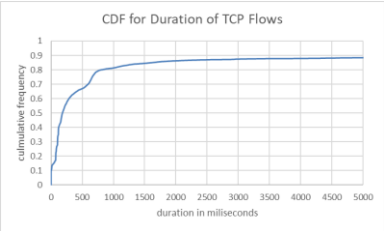


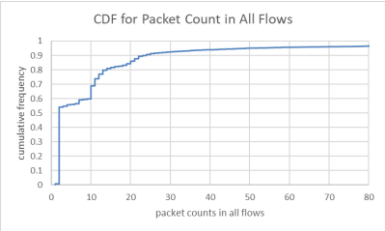
headers has size at least twenty bytes. This may attributes to the application that uses UDP, such as media streaming might takes up more payload space for each packet whereas common application uses TCP are web browser or mailing service. Furthermore, notice that around 40% of UDP packets have size of 1220 bytes and a few outlier packet with size of more than 1.4KB. Thus, since UDP and TCP segments are encapsulated inside IP datagram, the CDF shape of IP packet size and all packet size are determined by the size of UDP and TCP packets, therefore, packet size charts have similar shape. Futhermore, the header size of both IP and TCP packets have varaible sizes due to to the existence of option field in the header. Since the option field in IP header are mostly not used, we see that the percentage of IP packets that use the option fields are negligible. However we see that more than 20% of TCP packets has header size of more than twenty bytes and that is logical since TCP may include the maximum segment size inside the option fields. Lastly, we see that almost 92% of non-IP packets have size less than 50 bytes.

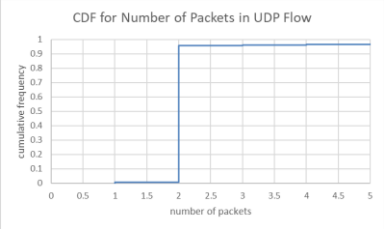
Number of TCP Flows in the trace file	Number of UDP Flows in the trace file
11355	11933

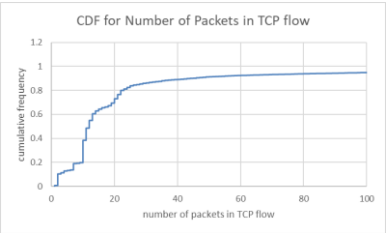




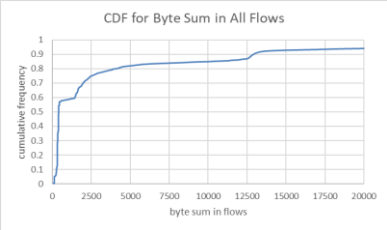


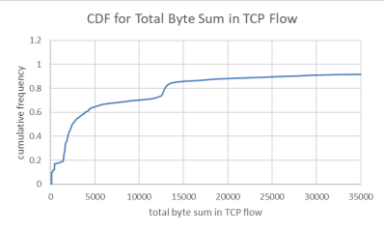


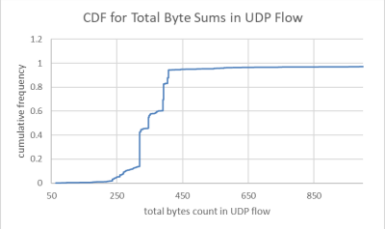


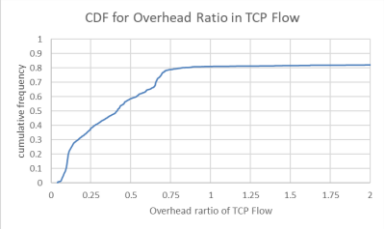


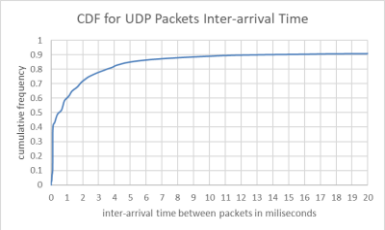
After reconstruct the TCP and UDP flows from the trace file, we get a more in depth understanding of the captured network traffic. In total, there are 11355 TCP flow connections and 11933 UDP flow connections. In comparison, there is huge difference in flow duration between TCP and UDP flows, around 90% of TCP flows has duration of 5000 miliseconds (the rest are considered as outliers) whereas 90% of UDP flows only takes 0.2 miliseconds (the rest are considered as outliers). The time difference between the two type of flows can be explained by the type of service that each provides to applications and what kind of applications use it. For UDP, it is very light weight, it only provides de/multiplexing service to applications. For TCP, in additional to the de/multiplexing service, it is connection oriented in order to connect any two hosts in the network, it provides reliable, in-order delivery

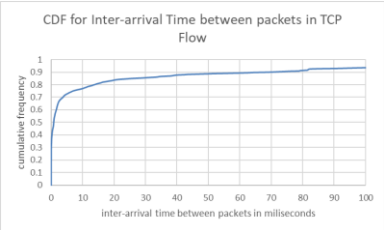


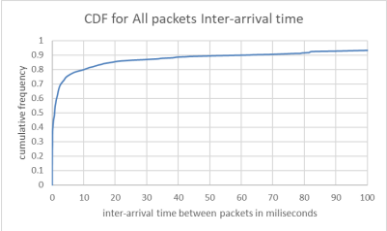










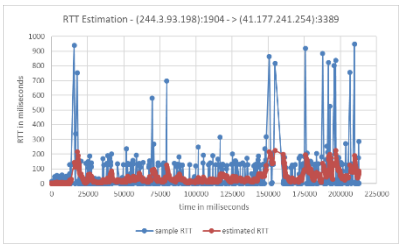


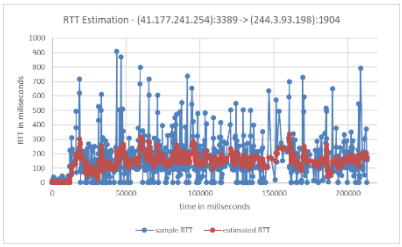
of data between two hosts, it has flow control to prevent overwhelm the receiver, and it has congestion control to prevent overload the network. The longer duration of TCP flows could blame to the additional features that it proivdes, such as time for the connection establishing phrase and tear down phrase, retransimission of packets due to timeout or losses, and slow down the transmission of data due to congestion of the network or the limited buffer size echoed from the receiver. However, the CDF of flow size in terms of byte sums provides a different view of the duration. We see that almost 90% of the flow size are less than 450 bytes in UDP flows whereas approximately 80% of TCP flow size are 12500 bytes or less. Those CDF results also suggest that the captured traffic have much larger TCP flows in terms of total bytes count. Futhermore, excluding outliers, almost all of the UDP flows have only two packets and in comparison to the CDF of UDP flow byte sum, it suggests that most of the two-packets UDP flow has size in between 250 and 450 bytes. Whereas TCP flow has significantly larger flow size both in terms of byte sums and packets count. In terms of network resources utilization, the CDF for TCP overhead ratio shows that almost 80% of the TCP flows have overhead ratio of 0.75, which implies that for every 100 bytes sent, we have up to 75 bytes of wasted bandwidths. The rest of 20% attributes to failed TCP connections (not transfer any data), which are not shown. In terms of inter packet arrival time statistics, 20 ms is the common elapsed time between packet arrivals both in UDP flow (90%) and TCP flow (> 80%) but TCP flows seem to have a higher variance of inter-packet arrival time since the 90th percentile is 100 ms. Overall, these statistics suggest the tradeoff between using TCP and UDP; TCP provides more reliable service but much slower and more complex whereas UDP provides faster process-to-process communication but limited services.

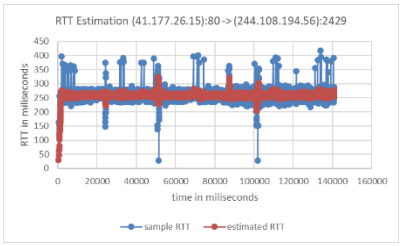
The final piece of flow statistics are the TCP state. Since pcap file captures less than three minutes of network traffics, no failed connections occurs. However, it is worth to note that over the three minutes period, most of the TCP connections are finished, this depends on the application that uses TCP, an example that has such short-lived connection can be google some information. Furthermore, since trace file only record less than five minutes of network traffics, all the requested TCP connections are belong to ongoing state.

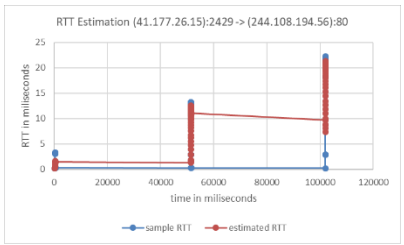
TCP state	Number of connections
Ongoing	1858
Reset	1821

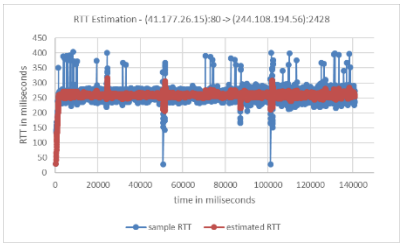
Finished	7675
Requested	0
Failed	0

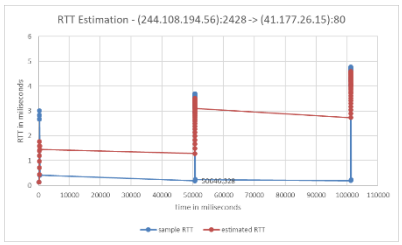


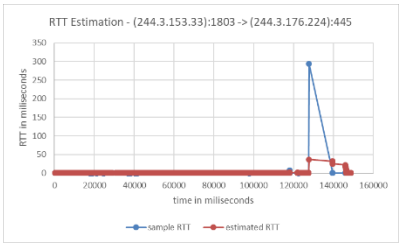


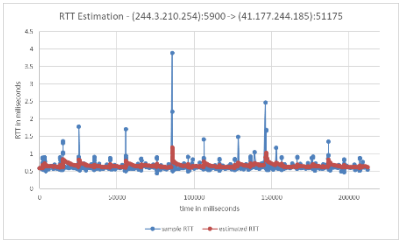


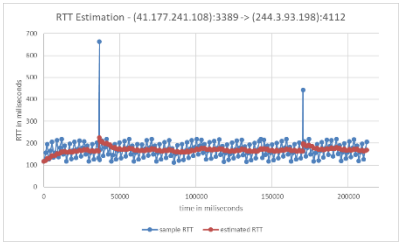


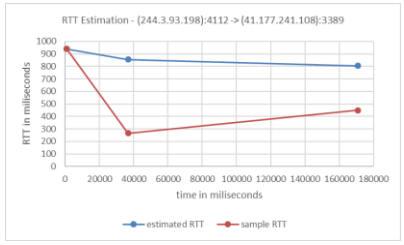


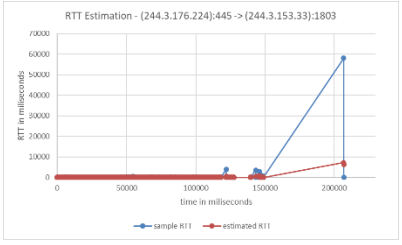


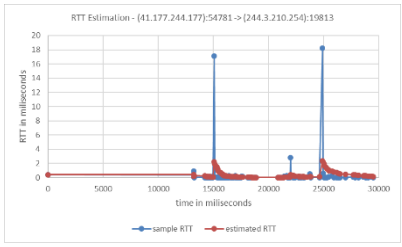




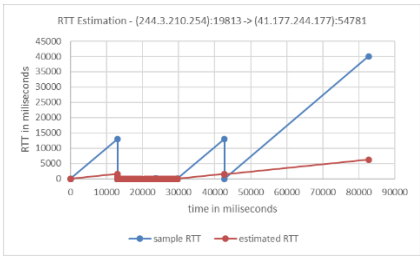




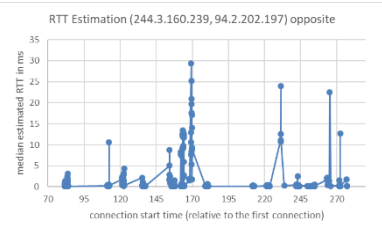


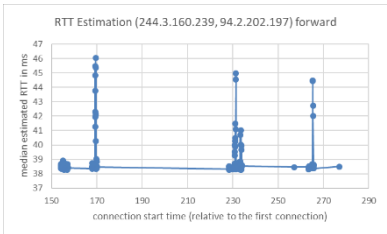


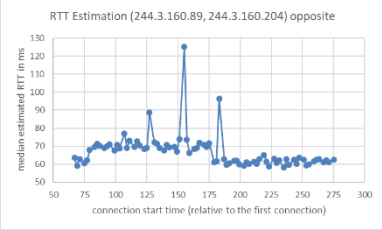
The last part the analysis is about round-trip time estimation. In my trace file, two of the largest TCP flows in terms of packet number are the same as two of the three in largest TCP flows in terms of total byte size so we have total of 13 charts; each side by side chart on the left represents each direction of a same TCP flow except the last one (it only has one sample, the SYN packet). Let's consider the RTT estimation for each direction of the flow and we see some interesting results out of these charts. First of all, we see the some of the TCP flows have sampled RTTs fluctuate very rapidly while the corresponding estimated RTT are relatively smooth. The smooth trend of theses estimated RTT can be attributed to two properties of the estimation procedure, smooth running average and the *alpha* constant (1/8) that we chose to find the estimated RTT. Reference to RFC 6298 provided in the handout, the estimated RTT is (1 - *alpha*) * *running_avg* + *alpha* * *new_sampled_rtt*. According to the chosen *alpha* value and the equation, we see that it gives more credits to the *running_avg*, which means that the trend of the estimated RTT reacts very slowly to temporary fluctuations of sampled RTT, which has a trade off that it does not adapt the real change of network traffic quickly. Furthermore, rapid changes of the sampled RTT during the life of a connection might be caused by a few reasons. Firstly, it can be caused by the delayed acknowledgement packet from receiver, this does not caused by congestion of the

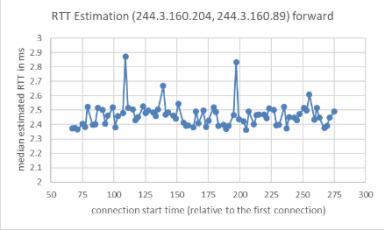


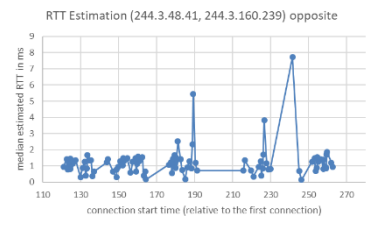
network, instead, the receiver tries to reduce the overhead and so waits for the application sends more data and piggyback the acknowledgemnt inside the same data packet, thus the time of waiting for new data from the receiver side can be the cause. Secondly, the changes of sampled RTT can be caused by the queueing delay in routers' buffer along the end-to-end path; as a example, it can be a situation where multiple TCP flows arrive to the same routers in a burstly way so that the packets are delayed by queueing in the buffers. Last but certainly not least, the sudden increase in sampled RTT could be caused by receiving culmulative acknowledgement of a sample packet, that is, the current packet arrives to the receiver while the previous packets may be delayed or lost due to congestion of the network and require retransimssion from sender, therefore, the acknowledgement of the current packet is not sent by receiver until the all the preivously lost packets are received. It is worth to note that the round-trip times are sampled only for packets that sent exactly once based on the Karn's algorithm. Subsequently, let's consider the other direction of TCP flows, notice that one of the flow has only one or a few sampled RTTs. The situation could be caused by the type of application used on each end, probably one sides kept sending data while the other side only accept data packet and only sents acknowledgement packets. Also, the general pattern of the RTT are very different in both direction of the same flow.

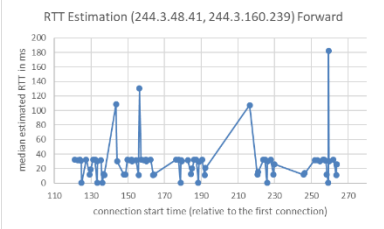












Lastly, let's analyze the the RTT estimation of the top three host pairs that have the highest number of TCP connections in-between (each side by side chart represents a host pair, one for each direction); each point of these chart is the median estimated RTT of a particular connection of a host pair. For each of the host pair RTT estimation charts, some of the host pairs have similar median estimated RTTs over time despite a few outliers, such as host pair (244.3.160.89, 244.3.160.204) and (244.3.160.204, 244.3.160.89). The change of median estimated RTT seems mostly follow the pattern of up and down. Since the measurement are taken from packet that only transmitted once, the cause of changes could be made by the queueing delay along the end to end path; congestion occurs so that each packet receives the culmulative aknowledgement from receiver; or delayed acknowledgement from receivers. Compare all three pairs, (244.3.160.89, 244.3.160.204) and (244.3.160.204, 244.3.160.89) both have similar trends of estimated RTT, but the pair (244.3.160.89, 244.3.160.204) has much higher round trip time for each connection, the longer estimated RTT could be caused by the physical distance between the two ends or each packet in each connection chose a path that has longer queueing delay or congested. The trend of the third pair (244.3.16.239, 94.2.202.197) depicts a different story, host on each end seems

to have a cluster of connections within microscopic time interval and each cluster of connections are spaced apart. Furthermore, each median estimated RTT kept increasing for each cluster of connection, this might caused by the queueing delay in routers' buffer along the path between the two hosts and since the inter-connection time between the two hosts are so closed together, it made the already congested network even more congested, which explains the increasing estimated RTT for each cluster of connections.