1. **TCP Simultaneous Open.**

   The sim_open.pcapng file is attached.

2. **TCP Reno vs Cubic.**

   a) Description of the setup used:
   - For the TCP competition, the file competition2.py is used with different values for RTT.
   - Two of the nodes, h1 and h3 are used from it.
   - h3 running the dualreceive2.py file, with block size of 20,000.
   - h1 will send blocks, by initiating two different connections with TCP Reno and TCP Cubic, and sending it to h3.
   - Any of the connection completing its 20,000 blocks first, will close the connection, and display the time taken by it.

   b) Output/Calculations:

RTT=50ms

TCP Reno	TCP Cubic	Time Finished	Reno Throughput	Cubic Throughput	Total Throughput
20,000	19,043	41.175 sec	485.73	462.49	948.22
20,000	18,955	41.228 sec	485.107	459.76	944.867
20,000	15,113	37.133 sec	538.6	406.99	945.59
RTT=100ms					
14,430	20,000	37.34 sec	386.45	535.618	922.07
15,151	20,000	37.81 sec	400.71	528.96	929.67
17,507	20,000	40.742 sec	429.7	490.894	920.6
RTT=200ms					
13,668	20,000	40.1 sec	340.84	498.75	839.6
16,263	20,000	43.23 sec	376.2	462.64	838.84
15,147	20,000	41.69 sec	363.3	479.73	843.03

   c) Description:

   When any of the connection completes its 20,000 blocks first, dualreceive2.py closes the connection. So, total blocks transferred per unit time will be the total throughput. Relative throughput of both the connections can also be calculated, which is shown in the above table. The above table also shows that, by increasing the RTT value, the total throughput decreases. And TCP Cubic performs well with large RTT values as compared to TCP Reno.