**1) Describe the parameters you use, e.g., length of time to transmit (with -t), number of parallel connections (with -P), and other parameters if you specify any. Note, only the number of parallel connections should vary, and all other parameters should be fixed.**

Parameters:

* Port (-p) = 5201 Reason for choice: arbitrary. Any port that would connect to the iperf server was sufficient.
* Length of time to transmit (-t) = 5. Reason for choice: Anything less than 4 seemed like an insufficient amount of time, but anything over 7 dragged out testing time. Thus, -t 5 was deemed a fair compromise.
* Parallel connections (-P) = [1, 20]. Reason for choice: chose a large enough window to showcase throughput v. parallel connections trend.
* Trial runs = 3. Reason for choice: the iperf command for –P [1, 20] were run three times to ensure that the observed trends would corroborate each other.

The iperf command was run with the above parameters, and three repetitive trials were done with the above parameters (Figure 2), and their results were averaged in Figure 1 (for a given P, calculate mean throughput across the the three trials) to diminish the effect of one-time, unrelated outliers (Figure 1).

**2) What's the number of parallel connections that give the highest aggregate bandwidth? What's the trend of aggregate bandwidth as the number of parallel connections increases?**

Based on Figure 1, the number of parallel connections P that give the highest bandwidth seems to level off at P=[3, 6].

The trend of aggregate bandwidth resembles an asymptotic relationship: initially as P increases, aggregate bandwidth quickly increases at P = [1, 3]. As P continues to grow, aggregate throughput hovers at just below 200 Mbits / sec and rarely deviates.

**3) Briefly explain the possible reasons behind what you observe. For instance, if aggregate bandwidth increases with more parallel connections at the beginning, then what's the mathematical formula that may explain this? And if aggregate bandwidth stops increasing or even decreases a bit later on, what might be the possible causes?**

the initial increase in throughput from P=[1, 3] can be explained by the observation we made in class that clients sometimes establish multiple connections in an attempt to demand more than their fair share of bandwidth from the network. They attempts to exploit TCP fairness algorithms (we learned proportional fairness and max-min fairness in class) allocate resources, knowing that TCP awards resources by attempting to maximize utility for all users, subject to the constraints of the network’s capacity. Therefore, clients set up multiple “users” (connections) to increase their chance of being awarded larger shares of bandwidth. Based on Figure 1, the client is rewarded for this strategy from P=[1, 3]; the number of connections that my client establishes correlates positively with the aggregate bandwidth.

However, the “levelling-off” that we observe from P >= 3 can be explained by the effects congestion offsetting the the gains from increased bandwidth allocation. We learned in class that the load that TCP allows a sender to transmit (aka throughput) depends the sender’s congestion window size. That window sized is halved (according to the AIMD algorithm) when the client detects congestion in the form of packet drops (likely due to the receiver’s packet buffer overflowing)—an attempt to lighten load on the network. However, the multiplicative decrease in window size also decreases the client’s throughput. Thus, opening too many parallel connections can result in congestion, causing the client’s throughput to decrease. At some point, the decrease in bandwidth from congestion offsets—and occasionally overpowers—the gains from multiple connections. This phenomenon manifests itself as a “levelling-off” after P=6 (see Figure 1, and Figure 2 trial 10) or even a decrease (see Figure 2, trial 8, P=[11, 15]) in bandwidth.

**Figure 1. Aggregate Bandwidth vs. Number of Parallel Connections, average of three repetitive trials.**



Figure 2. Aggregate Bandwidth vs. Number of Parallel Connections, data from three separate trials.

**1) Describe the parameters you use, e.g., length of time to transmit (with -t), which servers you use (and where they are located), and other parameters if you specify any. Note, only the server names should vary, and all other parameters should be fixed.**

Parameters:

* Port (-p) = 5201. Reason for choice: arbitrary. Any port that would connect to the iperf server was sufficient.
* Length of time to transmit (-t) = 5. Reason for choice: Anything less than 4 was deemed an insufficient amount of time, but anything over 7 dragged out testing time. Thus, -t 5 was deemed a fair compromise.
* Parallel connections (-P) = 1. Reason for choice: no need for parallelism, one connection is sufficient.
* Servers chosen: see Figure 3. Reason for choice: every single server on the iperf servers page that would accept pings were used. I wanted as many data points as possible for accuracy.
* Ping count = 5. Reason for choice: Anything less than 4 was deemed an insufficient amount of time, but anything over 7 dragged out testing time. Thus, 5 was deemed a fair compromise.
* RTT: average round trip time used from ping.

|  |  |  |  |
| --- | --- | --- | --- |
| Bandwidth (Mbits / sec) | Avg RTT (ms) | Server | Country |
| 101 | 68.708 | iperf.scottlinux.com | USA |
| 90.3 | 78.876 | bouygues.iperf.fr | France |
| 40.1 | 121.214 | iperf.volia.net | Ukraine |
| 35.7 | 165.251 | iperf.it-north.net | Kazakhstan |
| 8.01 | 298.303 | iperf.biznetnetworks.com | Indonesia |

**Figure 3. Chosen servers and the corresponding countries, noted with their bandwidth and average RTT.**

**2) What mathematical relationship should you expect to see between throughput and RTT? Why?**

Bandwidth is inversely proportional to RTT (see Figure 3). BW(rtt) = 1 / rtt.

The reason for this is that TCP interprets delay and packet drops as symptoms of network congestion, and the protocol reacts by halving its congestion window (AIMD algorithm). TCP might be interpreting a large RTT as a delay (the chances of a timeout occurring are much higher). In addition, servers located in geographically distant locations (Indonesia, Kazakhstan) may also require that transmitted packets endure more hops before arriving at their destinations. This increases the chance of a packets being dropped enroute for reasons unrelated to congestion. However, we discussed in class that TCP interprets packet drops as symptoms of congestion, and the protocol still reacts by halving its congestion window, effectively decreasing its bandwidth. Thus, the longer delay and increased packet dropping associated with larger RTT’s “look” very much like congestion to TCP, causing the protocol to restrict is output. This manifests itself as an inverse relationship between bandwidth and RTT.



**Figure 3. Bandwidth vs. Avg RTT. Bandwidth is inversely proportional to average RTT.**

**3) If what you observe from your plot is not strictly the same as what's shown in the math equation (which is usually the case), then briefly explain what are other possible factors which may affect throughput.**