Advanced Computer Networks

Assignment 1: Network Principles

Submitted by Jinank Jain

Solution 1

Bandwidth-Delay Product:
Mathematically it is defined as the product of bandwidth and round-trip delay time. Intuitively bandwidth-delay product of a transmission path defines the amount of data TCP should have within the transmission path at any one time, in order to fully utilize the available channel capacity.

Bandwidth-Delay product is usually most essential component of reliable communication as it will help to indicate congestion level in the network, thus it would be an important part of only those protocols which care about the reliability of the communication such as like TCP which uses it for TCP tuning because the protocol can only achieve optimum throughput if a sender sends a sufficiently large quantity of data before being required to stop and wait until a confirming message is received from the receiver, acknowledging successful receipt of that data.

Example of the system that has a large bandwidth-delay product are High-speed terrestrial network: 1 Gbit/s, 1 ms RTT, Mobile broadband (HSDPA): 6 Mbit/s, 100 ms RTT etc.

Solution 2

Part a
In order to solve this part I make a simple assumption that there is no congestion and no packet loss. In the first RTT 1KB would be send, in the second 2KB would be sent for a total of 3KB. On round n, the amount of data sent during this round is given by the equation:

$$data_n = \begin{cases} 2^{n-1}KB, & \text{if } x \leq 10 \\ 1024KB, & \text{otherwise} \end{cases}$$

In total of n iterations we would get the following

$$total\_data_n = \begin{cases} \sum_{i=0}^{n} 2^{i-1}KB, & \text{if } x \leq 10 \\ 1024 * (n-10) + 1024KB, & \text{otherwise} \end{cases}$$

After analysis, we find that in 19 RTTs, a total of 10,239 KBytes (this is equal to 10 MBs - 1 KB, assuming that 1024 KB = 1 MB) have been sent and in 20 RTTs, a total of 11,263 KBytes have been sent. So after 20 RTTs the file would be sent completely.

Part b
Time to send the file would be:

$$\text{Time} = 20 \text{ RTTs} * 100\text{ms} = 2 \text{ secs}$$

And the effective throughput would be the following:

$$\text{Throughput} = \frac{10 * 8Mb}{2sec} \text{ Mbps}$$

$$\text{Throughput} = 40 \text{ Mbps}$$

Part c
If receive sizes are much larger then 1MB there will be no saturation and we can modify the above formula:

$$10*1024 \text{ KB} = \sum_{i=0}^{n} 2^{i-1} * 1\text{KB}$$

In this case the approximate value n would be 13.3 so after 14 RTT we could say that complete file would be transferred. So the new effective throughput would be:

$$\text{Throughput} = \frac{10 * 8Mb}{1.4sec} \text{ Mbps}$$

$$\text{Throughput} = 57.14 \text{ Mbps}$$

Solution 3

According to the formula given in the question we can get the following parameters:

$$\text{Bandwidth} = 1 \text{ Gbps}$$

$$\text{File Size} = 128 \text{ Mb}$$

$$\text{Throughput} = \frac{128}{RTT + 0.128} \text{ Mbps}$$

Effect of RTT on file transfer:
With increase in RTT throughput of the link would decrease and we would not be utilizing the link to the maximum extend on the other if RTT goes to zero we would reach near 1 Gbps which is close to maximum bandwidth that link could achieve.

Graph of Throughput V/S RTT

Solution 4

- In the first if we consider the case of without segmentation total time taken from Source to Destination would be three time of time taken from source to packet switch 1 as all the links have the same speed and thus

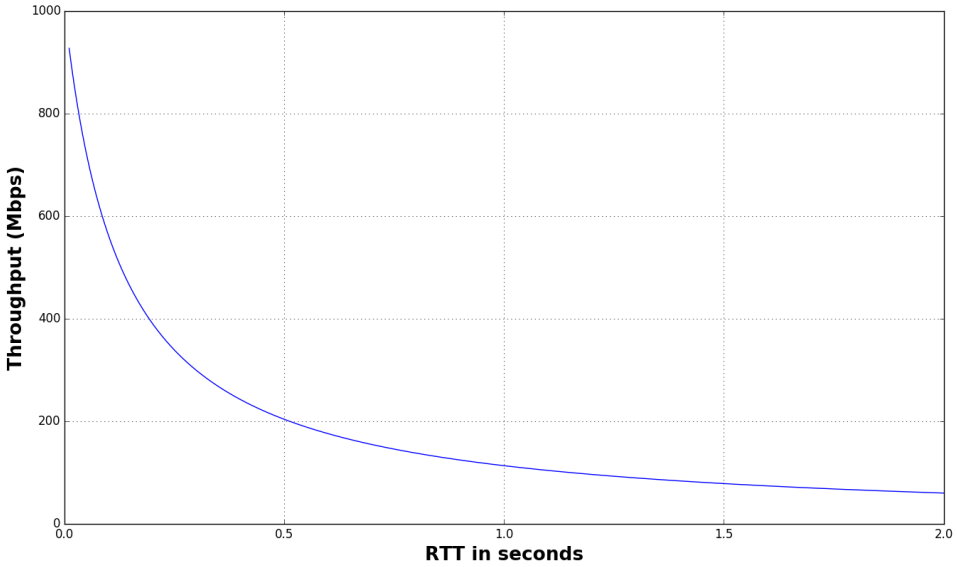

Figure 1: Throughput V/S RTT

$$\text{Time} = \frac{3 * 7.5 * 10^6}{1.5 * 10^6}$$

$$\text{Time} = 15\text{s}$$

- In the second case, time taken for a packet to reach from source to first packet switch would be following

$$\text{Time} = \frac{1500}{1.5 * 10^6}$$

$$\text{Time} = 1\text{ms}$$

- In the third case, time taken for a file to reach from source to destination considering fragmentation would be following

$$\text{Time} = \frac{3 * 1500}{1.5 * 10^6} + \frac{7.5 * 10^6 - 1500}{1.5 * 10^6}$$

$$\text{Time} = 5002 \text{ ms}$$

Solution 5

Part a

Some sender can create more than one flow as compared to some sender who is just creating one flow. In that case sender 1 has an advantage over sender 2 by creating more flows In the figure 2 sender(s1) creates two flows for the same request while sender(s2) creates only one flow and then according to flow fairness we violate per-sender fairness.

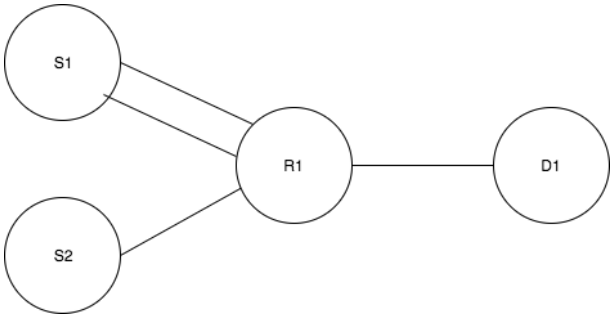

Figure 2: Flow Fairness v/s Per-user fairness

Part b

Network support seems like an obvious requirement because someone has to know which flow is coming from which sender and thus after that apply proper rules and regulations on it.