Networks Lab 5

Brian Mackwan (22B0413) and Arohan Hazarika (22B3948)

October 2024

1 Q1

To prove that $\frac{P}{t_2 - t_1} = C$, where C is the bottleneck link speed, we will use induction on the number of links in the path.

a. Assumptions and Definitions:

- P is the size of each packet in bits.
- t_1 is the time of arrival of the last bit of the first packet at destination D.
- t_2 is the time of arrival of the last bit of the second packet at destination D.
- C is the bottleneck link speed (the slowest link speed in the entire end-to-end path).
- Let the path from the sender S to the destination D consist of k links, each with speed $C_1, C_2, \ldots, C_k$. The bottleneck speed C is the minimum of $C_1, C_2, \ldots, C_k$, i.e., $C = \min(C_1, C_2, \ldots, C_k)$.

b. Step 1: Base Case ($k = 1$)

For $k = 1$, there is only one link between S and D, and the speed of the link is C_1.

- Time to transmit the first packet from S to D:

  $$t_{\text{transmit}} = \frac{P}{C_1}$$

  where P is the size of the packet in bits and C_1 is the link speed in bits per second.

- Since there is only one link and no other packets, after the first packet is transmitted, the second packet starts transmitting immediately. The time of arrival of the last bit of the second packet at D is simply:

  $$t_2 = t_1 + \frac{P}{C_1}$$

  Thus, the difference between the times of arrival of the last bits of the two packets is:

  $$t_2 - t_1 = \frac{P}{C_1}$$

  This gives:

  $$\frac{P}{t_2 - t_1} = C_1$$

For $k = 1$, the link speed is C_1, which is the bottleneck speed (since there is only one link), so the statement is true for $k = 1$.

c. Step 2: Induction Hypothesis

Assume that the statement $\frac{P}{t_2 - t_1} = C$ holds for a path consisting of k links, where C is the bottleneck speed for the first k links.

d. Step 3: Induction Step ($k \to k+1$)

Now, we consider a path with $k+1$ links, with speeds $C_1, C_2, \ldots, C_k, C_{k+1}$. The bottleneck speed of the path till the router just before D is $C = \min(C_1, C_2, \ldots, C_k)$.

Let t_1 be the time of arrival of the last bit of the first packet at the router just before D, and let t_2 be the time of arrival of the last bit of the second packet at the router just before D.

- By the induction hypothesis, for the first k links, the difference between the times of arrival of the last bits of the two packets at the end of the k-th link is $\frac{P}{C}$, where C is the bottleneck speed of the first k links.
- The time taken for the packets to traverse the $(k+1)$-th link depends on the speed C_{k+1}.
- Let T_1 and T_2 be the times of arrival of the last bit of the first and second packets respectively at D.
- We can write $T_1 = t_1 + P/C_{k+1}$ and $T_2 = P/C_{k+1} + max(t_1 + P/C, t_1 + P/C_{k+1})$, the first term in the max expression means we are starting to send the second packet to its final destination D as soon as we receive the last bit of the second packet and the second term implies we will start sending the second packet as soon as the first packet is completely sent, therefore we take max of these two times
- Now, if $t1 + P/C \geq t_1 + P/C_{k+1}$ implies $C_{k+1} \geq C$ which means C remains the bottleneck speed and $C = P/(T_2 - T_1)$
- Else, $t1 + P/C < t_1 + P/C_{k+1}$ implies $C_{k+1} < C$ which means C_{k+1} becomes the bottleneck speed and $C_{k+1} = P/(T_2 - T_1)$

Thus, in both cases, $\frac{P}{T_2 - T_1} = C$, where C is the bottleneck speed of the entire path.

e. Conclusion

By induction, we have shown that $\frac{P}{T_2 - T_1} = C$, where C is the bottleneck link speed, for any number of links in the path. The base case for $k = 1$ holds, and the inductive step proves that the statement holds for $k+1$ links if it holds for k links. Therefore, the result is true for any path consisting of any number of links.

2 Implementation

a.

The following code is common to both sender and receiver which ensures creation of Datagram Sockets.

```
if ((sockfd = socket(AF_INET, SOCK_DGRAM, 0)) < 0) {
    perror("Socket creation failed");
    exit(EXIT_FAILURE);
}
```

The socket() function creates a new socket using the specified address family, socket type, and protocol. SOCK_DGRAM is used to establish a UDP connection. Since Unix sockets are simply handles using files as well hence its stored as a file descriptor in sockfd. We get -1 as output if it fails and hence, we use that as the condition for the error code.

b.

sender.c

```
if (sendto(sockfd, packet, packet_size, 0,
           (const struct sockaddr *)&dest_addr, sizeof(dest_addr)) == -1) {
    perror("sendto failed");
    exit(EXIT_FAILURE);
}
```

The above lines are used to send/write data to the socket.
The function sends data on the socket by taking the following as parameters - socket descriptor (sockfd), buffer containing data (packet), size of the data to be sent (packet_size), flags, destination address where the packet should be sent as well as Size of the destination address structure. It returns -1 if it fails and hence, we use that as the condition for the error code.

c.

receiver.c

```
double P = recvfrom(sockfd, buffer, sizeof(buffer), 0,
                    (struct sockaddr *)&client_addr, &addr_len);
if (P == -1) {
    perror("recvfrom failed");
    continue;
}
```

The above lines are used to read data from the socket.
The function receives data from the socket by taking the following as parameters - socket descriptor (sockfd), buffer to store the received data (buffer), size of the buffer used to store the data, flags, a pointer to the client address structure to store the sender's address as well as size of the client address structure.

The function returns the number of bytes of the data we receive from the socket. This number will be less than or equal to the size of the buffer passed as an argument to the function. It returns -1 if it fails and hence, we use that as the condition for the error code.

Hence, we store the value returned by this function to find the size of the packet sent which is used to find $C = \frac{P}{t_2 - t_1}$

d.

receiver.c

```
gettimeofday(&t2, NULL);
```

The gettimeofday() function gets the current time in seconds and microseconds since the Epoch. The function takes a pointer to a struct timeval to store the current time as well as a pointer to a struct timezone which in this is not needed since we want time difference anyway. The struct timeval stores the time in seconds and microseconds and we use the difference with the previous time to get the time delay $t_2 - t_1$ between the two packets in a pair.

e.

receiver.c

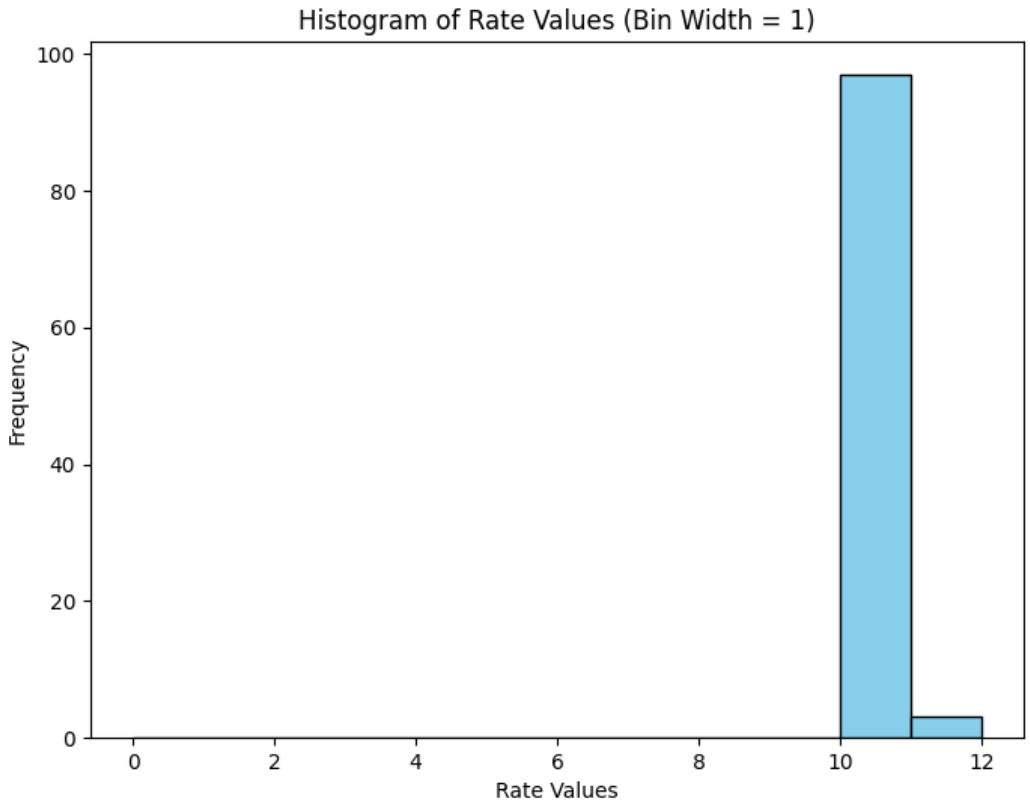
```
\begin{lstlisting}
sprintf(packet, "%d", 2 * i + 1);
sendto(sockfd, packet, packet_size, 0,
      (const struct sockaddr *)&dest_addr, sizeof(dest_addr));

sprintf(packet, "%d", 2 * i + 2);
sendto(sockfd, packet, packet_size, 0,
      (const struct sockaddr *)&dest_addr, sizeof(dest_addr));
```

The consecutive sendto() calls, with no delay between them, ensure that the first and second packets in a pair are sent back-to-back with no gap. The only gap between successive packet pairs is introduced by the usleep() function, which comes after both packets in a pair are sent. This guarantees that packets in a pair are transmitted immediately one after the other.

3 Experimentation

a. Experiment 1

In this part, we used the tc command to limit maximum throughput to 10Mbps and burst size to 9 kbits as we send 8kbits of data with 1000 ms spacing between packet pairs. 100 such pairs are sent which results in the histogram formed as seen ahead. As we see that the mode of the histogram i.e the most probable value of C is in the 10-11 bin. Hence, we would estimate the bottleneck speed to be the mean of the bin which is 10.5 Mbps which is very close to the limit of 10 Mbps that we had set.



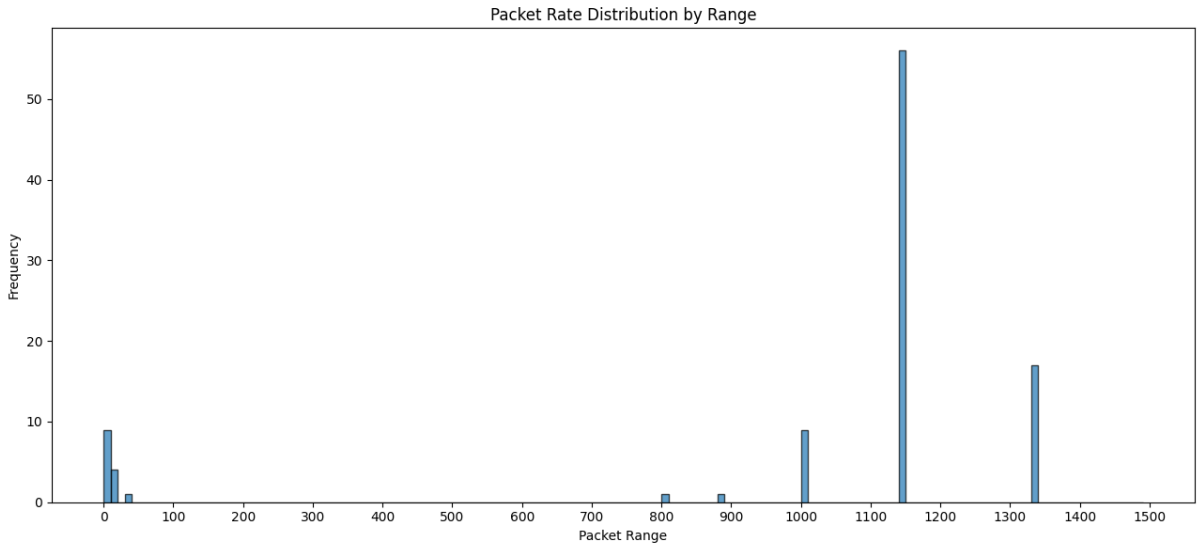
b. Experiment 2

i. Path 1

The first path we chose was between our personal laptop at H9 and the SL lab machine at CC. The IP of the lab machine was found using ifconfig and hence, we can see the number of hops between the two machines on the same network as seen ahead.

```
12  * *^C
brian@MACK1:/mnt/c/Users/brian/Downloads/Networks Lab 5$ traceroute 10.130.154.8
traceroute to 10.130.154.8 (10.130.154.8), 30 hops max, 60 byte packets
 1  MACK1.mshome.net (172.31.192.1)  0.901 ms  0.649 ms  0.550 ms
 2  10.64.0.1 (10.64.0.1)  3.812 ms  3.780 ms  4.147 ms
 3  172.16.12.2 (172.16.12.2)  12.387 ms  17.642 ms  12.359 ms
 4  10.250.130.2 (10.250.130.2)  8.418 ms  8.406 ms  8.392 ms
 5  10.130.154.8 (10.130.154.8)  8.396 ms  8.378 ms  7.405 ms
```

We also plot the histogram with estimates for C using a bin size of 10 Mbps. We observe that the mode of the histogram i.e the most probable value of C is in the 1140-1150 bin. Hence, we would estimate the bottleneck speed to be the mean of the bin which is 1145 Mbps.
(The actual mode of the collected singular data values was 1142.857 Mbps).

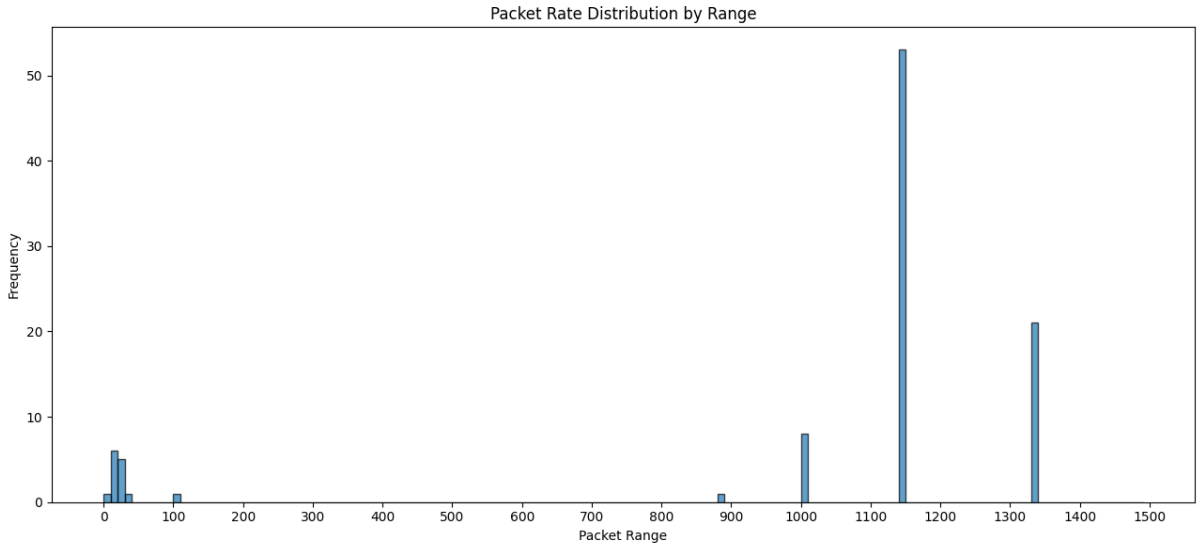


ii. Path 2

The second path we chose was between our personal laptop at H2 and the SL lab machine at CC. The IP of the lab machine was found using ifconfig and hence, we can see the number of hops between the two machines on the same network as seen ahead.

```
arohanh@LAPTOP-9PGAITT6:/mnt/c/users/admin/Downloads$ traceroute 10.130.154.8
traceroute to 10.130.154.8 (10.130.154.8), 30 hops max, 60 byte packets
 1  LAPTOP-9PGAITT6.mshome.net (172.26.192.1)  0.584 ms  0.471 ms  0.479 ms
 2  10.64.0.1 (10.64.0.1)  9.304 ms  9.293 ms  9.313 ms
 3  172.16.12.2 (172.16.12.2)  9.200 ms  9.187 ms  9.178 ms
 4  10.250.130.2 (10.250.130.2)  9.010 ms  8.999 ms  9.550 ms
 5  10.130.154.8 (10.130.154.8)  8.887 ms  8.827 ms  8.821 ms
arohanh@LAPTOP-9PGAITT6:/mnt/c/users/admin/Downloads$
```

We also plot the histogram with estimates for C using a bin size of 10 Mbps. We observe that the mode of the histogram i.e the most probable value of C is in the 1140-1150 bin. Hence, we would estimate the bottleneck speed to be the mean of the bin which is 1145 Mbps.
(The actual mode of the collected singular data values was 1142.857 Mbps).

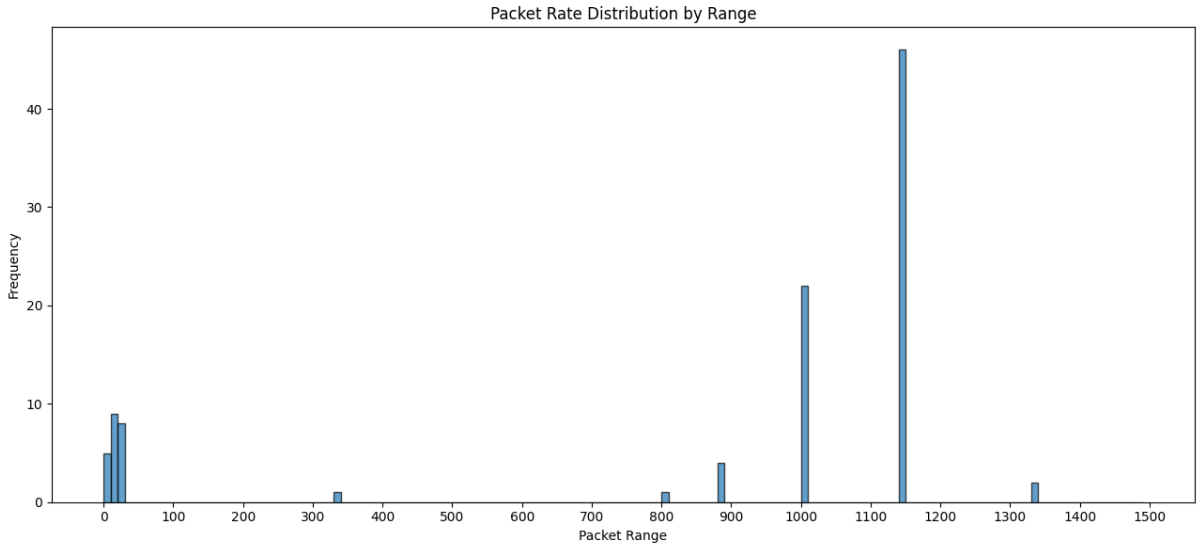


iii. Path 3

The third path we chose was between our personal laptop at H3 and the SL lab machine at CC. The IP of the lab machine was found using ifconfig and hence, we can see the number of hops between the two machines on the same network as seen ahead.

```
arohanh@LAPTOP-9PGAITT6:/mnt/c/users/admin/downloads$ traceroute 10.130.154.8
traceroute to 10.130.154.8 (10.130.154.8), 30 hops max, 60 byte packets
 1  LAPTOP-9PGAITT6.mshome.net (172.26.192.1)  2.031 ms  3.284 ms  1.081 ms
 2  10.64.0.1 (10.64.0.1)  4.222 ms  4.121 ms  4.342 ms
 3  172.16.12.2 (172.16.12.2)  4.464 ms  3.288 ms  2.995 ms
 4  10.250.130.2 (10.250.130.2)  3.473 ms  4.346 ms  3.388 ms
 5  10.130.154.8 (10.130.154.8)  3.753 ms  2.657 ms  3.035 ms
arohanh@LAPTOP-9PGAITT6:/mnt/c/users/admin/downloads$
```

We also plot the histogram with estimates for C using a bin size of 10 Mbps. We observe that the mode of the histogram i.e the most probable value of C is in the 1140-1150 bin. Hence, we would estimate the bottleneck speed to be the mean of the bin which is 1145 Mbps.



iv. Conclusion

Based on the observations, we can see that although the histograms do have variance as expected, they do have the same mode which is the most probable bottleneck link speed. Thus, if someone were to see the histograms they would estimate the bottleneck link speed to be 1145 Mbps since this estimation is consistent for all 3 paths.