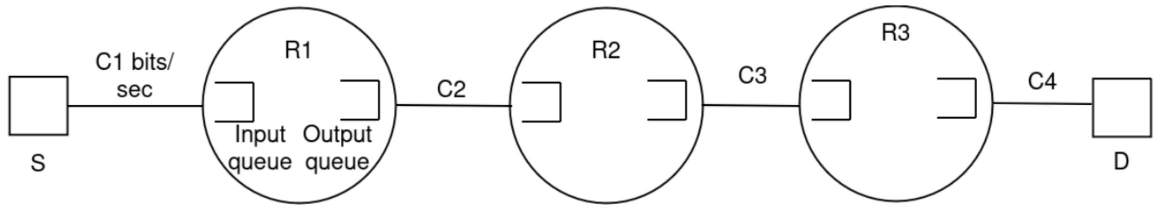
Report | Lab 5

CS378 - Computer Networks Lab

Omkar Shirpure (22B0910)
Suryansh Patidar (22B1036)

Contents

1 Part 01: Theory



Goal

We want to prove that

$$\frac{P}{t_2 - t_1} = C,$$

where P is the size of the packet, t_1 is the time of arrival of the last bit of the first packet at destination D, and t_2 is the time of arrival of the last bit of the second packet at D. The bottleneck link speed C is the smallest link speed along the end-to-end path.

Base Case: $k = 1$ (One Link)

Consider the first link between source S and the first router R_1, with link speed C_1. Since there are no other packets in the system, we can analyze the time it takes for both packets to traverse this link.

- The first packet of size P bits takes time $\frac{P}{C_1}$ to traverse the first link.
- Similarly, the second packet, which starts transmission immediately after the first packet, also takes $\frac{P}{C_1}$.

Thus, the time difference between the arrival of the last bits of the two packets at R_1 is:

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

Therefore, for the first link:

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

This holds true for $k = 1$.

Inductive Step

For the Induction step, assume that for a sub-path consisting of the first k links (i.e., from S to R_k), the statement holds. That is, the time difference between the two packets arriving at router R_k is given by:

$$t_2 - t_1 = \frac{P}{C^{(k)}_{\text{bottleneck}}},$$

where $C^{(k)}_{\text{bottleneck}}$ is the bottleneck link speed for the first k links.

Now consider the path consisting of $k + 1$ links. The time taken for the packets to traverse link $k + 1$ (from R_k to R_{k+1}) depends on the link speed C_{k+1}. The time difference between the last bit of the two packets arriving at R_{k+1} is determined by the slowest (bottleneck) link on the entire path from S to R_{k+1}.

Thus, the bottleneck speed for the first $k + 1$ links is:

$$C_{\text{bottleneck}}^{(k+1)} = \min(C_{\text{bottleneck}}^{(k)}, C_{k+1}),$$

and the time difference between the arrival of the last bits of the two packets at R_{k+1} is:

$$t_2 - t_1 = \frac{P}{C_{\text{bottleneck}}^{(k+1)}}.$$

Conclusion

By induction, the result holds for any $k + 1$ links. Therefore, the bottleneck speed C for the entire path is the smallest link speed among all links:

$$C = \min(C_1, C_2, \ldots, C_{n+1}).$$

And hence, we have proved that:

$$\frac{P}{t_2 - t_1} = C.$$

2 Part 02: Implementation

Introduction

Below explains how the sender and receiver programs work for sending and receiving packets using datagram sockets in C. The key aspects of the implementation are detailed, including socket creation, data transmission, packet reception, time measurement, and verification of packet pairs sent with minimal spacing.

(a) Datagram Socket Creation

In both programs, the sockets are created using the `socket` function:

```
int sock = socket(AF_INET, SOCK_DGRAM, 0);
```

The `socket` function creates a new socket and takes three arguments:

- `AF_INET`: Specifies the address family (IPv4).
- `SOCK_DGRAM`: Specifies that the socket is a datagram (UDP) socket.
- `0`: Specifies the protocol (default for UDP).

The function returns a socket descriptor, which is used in subsequent socket operations.

(b) Sending/Writing Data to the Socket

In the `sender.c` program, data is sent using the `sendto` function:

```
sendto(sock, message, packet_size, 0, (struct sockaddr *)&server_addr,
    sizeof(server_addr));
```

The `sendto` function sends a message to a specific destination. It takes the following arguments:

- `sock`: The socket descriptor.
- `message`: The message to be sent.
- `packet_size`: The size of the message in bytes.
- `0`: Flags (usually 0 for default behavior).
- `(struct sockaddr *)&server_addr`: Pointer to the destination address.
- `sizeof(server_addr)`: The size of the destination address structure.

(c) Reading Data from the Socket

In the `receiver.c` program, data is read using the `recvfrom` function:

```
int recv_len = recvfrom(sock, buffer, MAX_PACKET_SIZE, 0, (struct
    sockaddr *)&client_addr, &addr_len);
```

The `recvfrom` function receives data from the socket and takes the following arguments:

- `sock`: The socket descriptor.
- `buffer`: The buffer to store the received data.
- `MAX_PACKET_SIZE`: The maximum size of the buffer.

- 0: Flags (usually 0 for default behavior).
- (struct sockaddr *)&client_addr: Pointer to the sender's address.
- &addr_len: The size of the sender's address structure.

(d) Measuring Time of Arrival of Packets

In the receiver.c program, the time of arrival is measured using the gettimeofday function:

```
gettimeofday(&t1, NULL);  // For first packet
gettimeofday(&t2, NULL);  // For second packet
```

The gettimeofday function gets the current time and stores it in the structure t1 and t2. It takes two arguments:

- &t1: Pointer to a struct timeval where the time will be stored.
- NULL: Time zone information (not used in this case).

The difference between the two times is calculated to determine the time difference between the packet pair.

(e) Verifying Consecutive Packet Sending Without Gap

In the sender.c program, two packets are sent consecutively within each iteration of the loop. Since there is no sleep between the two sendto calls, the two packets are sent one after the other, with no delay. The relevant lines are:

```
snprintf(message, sizeof(message), "Packet %d", 2 * i + 1);
sendto(sock, message, packet_size, 0, (struct sockaddr *)&server_addr,
    sizeof(server_addr));

snprintf(message, sizeof(message), "Packet %d", 2 * i + 2);
sendto(sock, message, packet_size, 0, (struct sockaddr *)&server_addr,
    sizeof(server_addr));
```

This ensures that two packets are sent out in quick succession without any gap between them.

3 Part 03: Experimentation

3.1 Experiment - 1

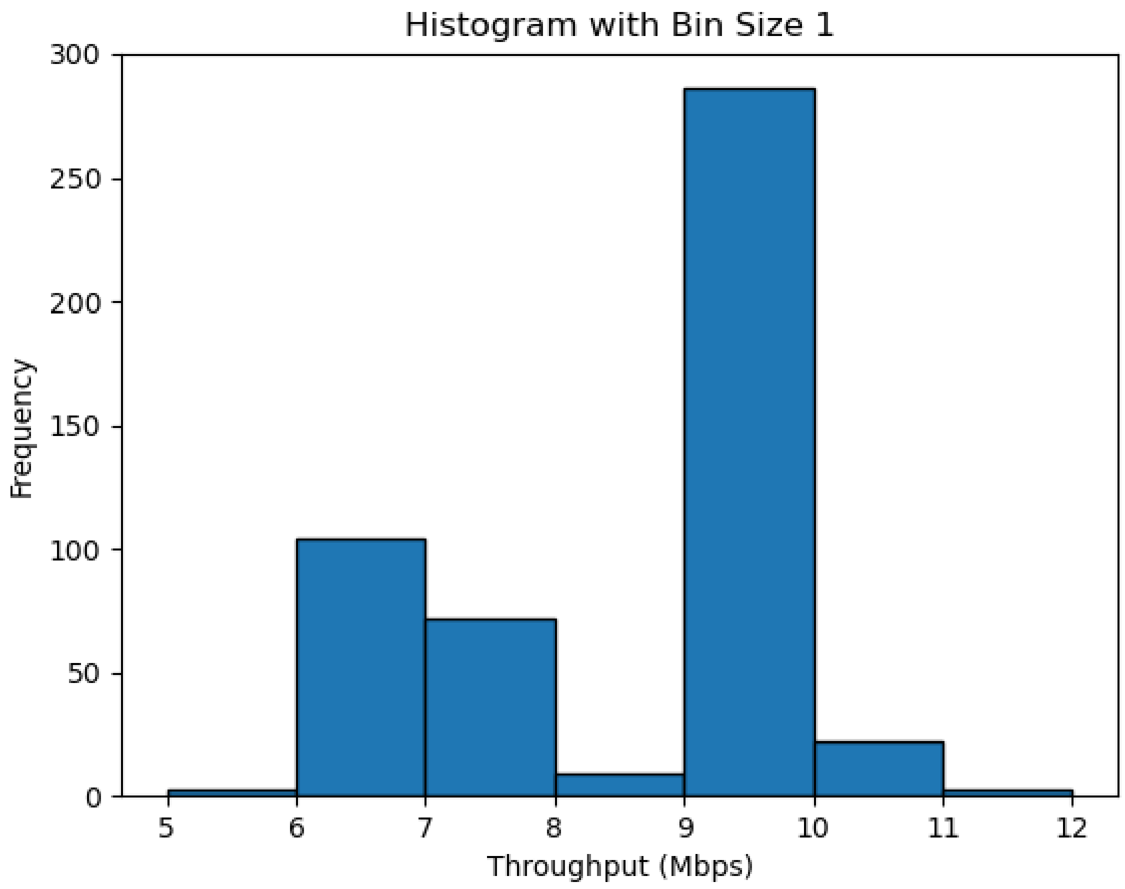


Figure 1: Experiment-1

Based on the observed histogram, the bottleneck value falls between 9 Mbps and 10 Mbps. This is identified as the mode of the histogram, representing the most frequent throughput observed in our experiment, and thus indicating the bottleneck throughput of the system.

3.2 Experiment - 2

3.2.1

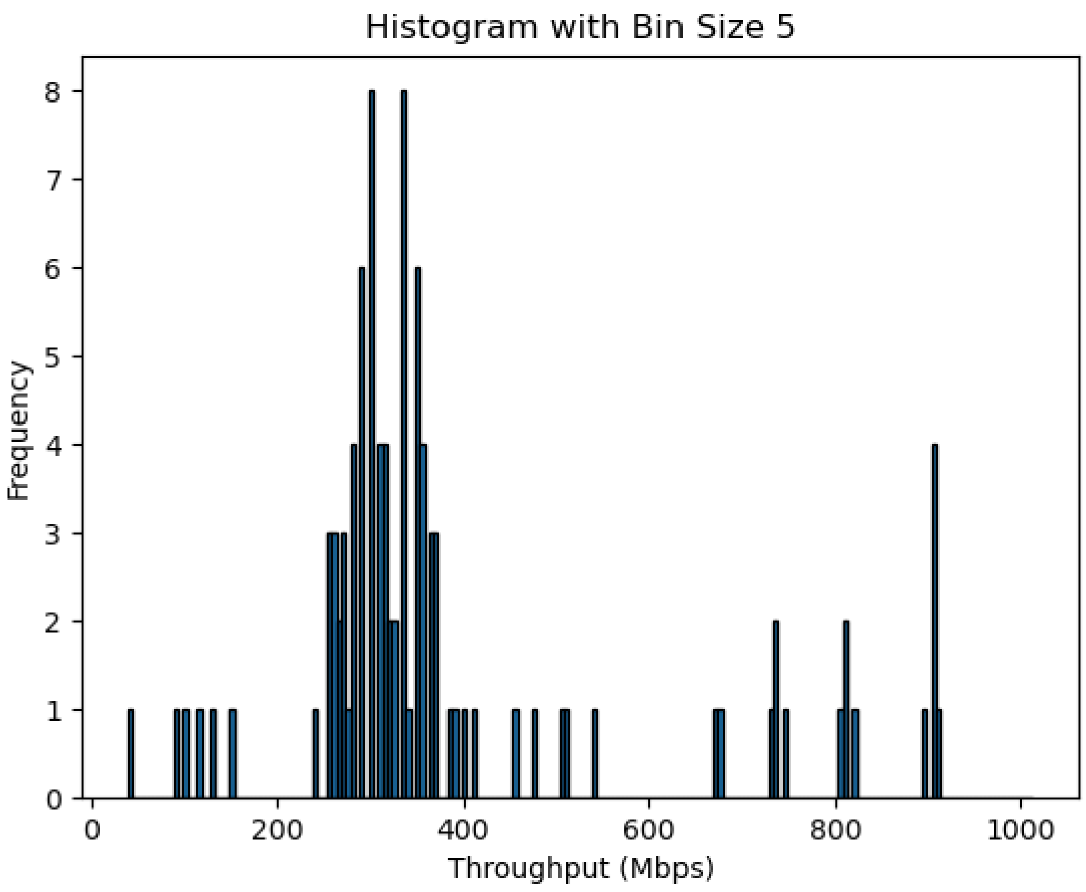
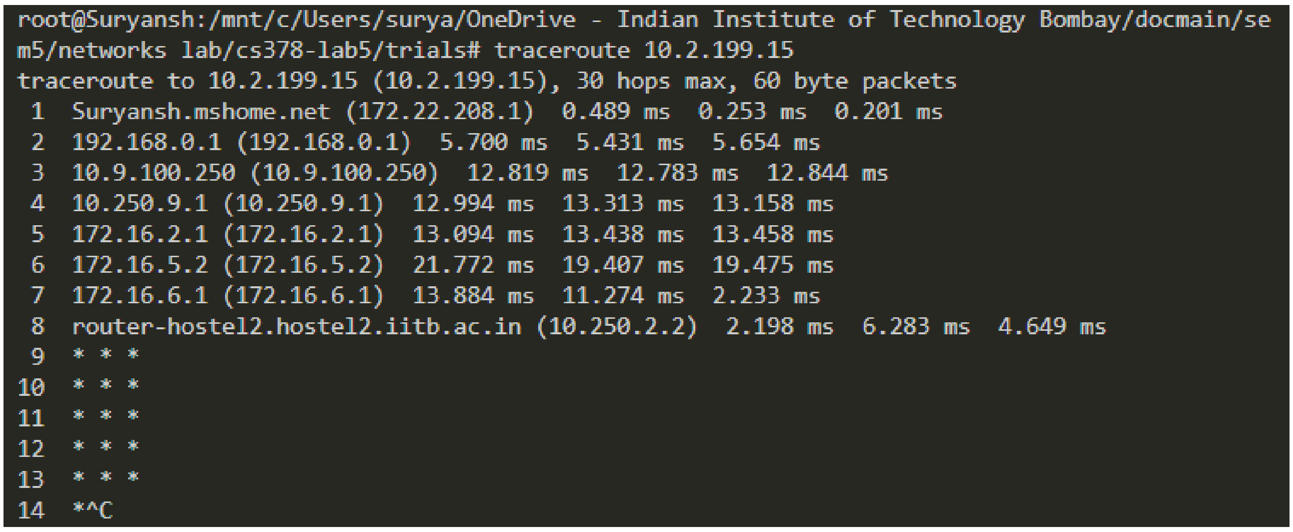


Figure 2: Experiment-2 (1) (From H2)



```
root@Suryansh:/mnt/c/Users/surya/OneDrive - Indian Institute of Technology Bombay/docmain/se
m5/networks lab/cs378-lab5/trials# traceroute 10.2.199.15
traceroute to 10.2.199.15 (10.2.199.15), 30 hops max, 60 byte packets
 1  Suryansh.mshome.net (172.22.208.1)  0.489 ms  0.253 ms  0.201 ms
 2  192.168.0.1 (192.168.0.1)  5.700 ms  5.431 ms  5.654 ms
 3  10.9.100.250 (10.9.100.250)  12.819 ms  12.783 ms  12.844 ms
 4  10.250.9.1 (10.250.9.1)  12.994 ms  13.313 ms  13.158 ms
 5  172.16.2.1 (172.16.2.1)  13.094 ms  13.438 ms  13.458 ms
 6  172.16.5.2 (172.16.5.2)  21.772 ms  19.407 ms  19.475 ms
 7  172.16.6.1 (172.16.6.1)  13.884 ms  11.274 ms  2.233 ms
 8  router-hostel2.hostel2.iitb.ac.in (10.250.2.2)  2.198 ms  6.283 ms  4.649 ms
 9  * * *
10  * * *
11  * * *
12  * * *
13  * * *
14  *^C
```

Figure 3: Experiment-2 (1) (Traceroute to H2)

Based on the observed histogram, the bottleneck value falls around 300 Mbps. This is identified as the mode of the histogram, representing the most frequent throughput observed in our experiment, and thus indicating the bottleneck throughput of the system

3.2.2

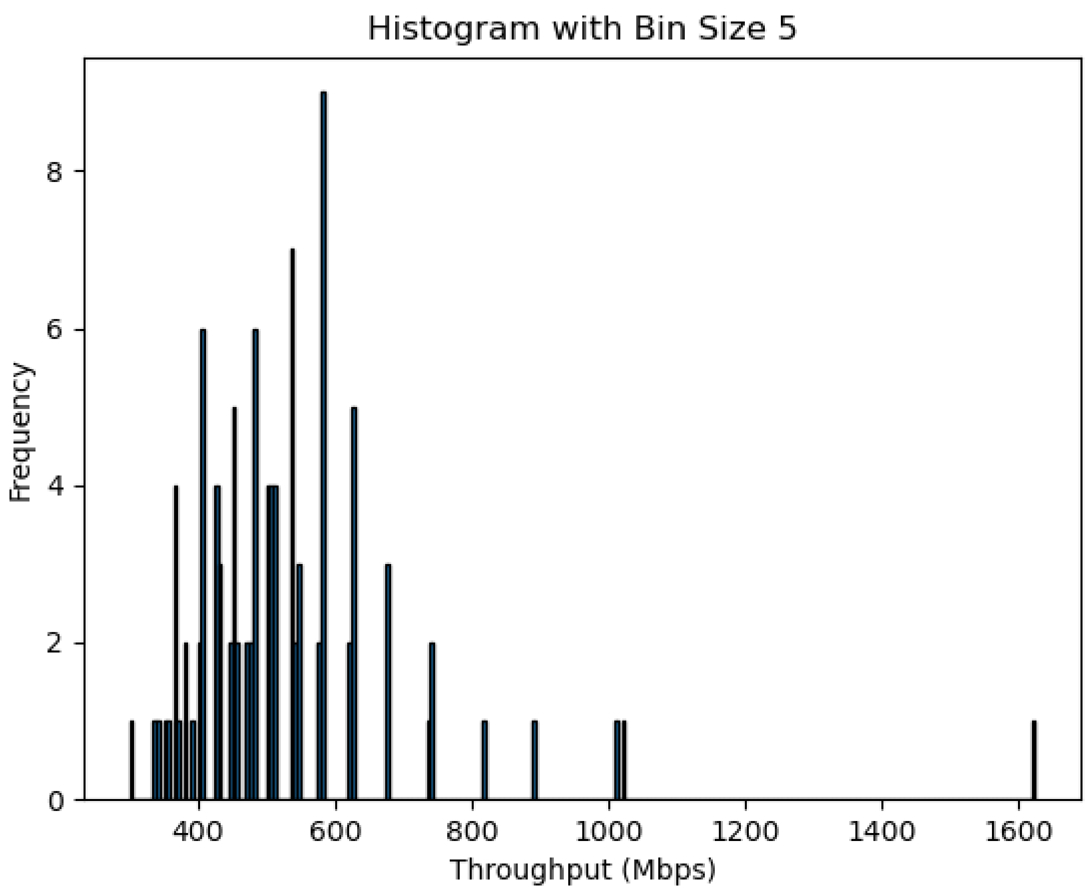
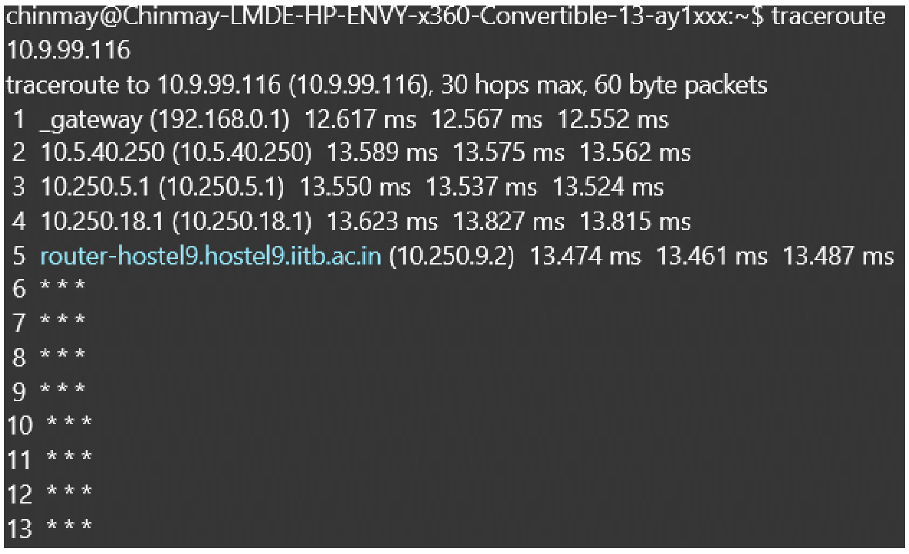


Figure 4: Experiment-2 (2) (H5)



```
chinmay@Chinmay-LMDE-HP-ENVY-x360-Convertible-13-ay1xxx:~$ traceroute
10.9.99.116
traceroute to 10.9.99.116 (10.9.99.116), 30 hops max, 60 byte packets
 1  _gateway (192.168.0.1)  12.617 ms  12.567 ms  12.552 ms
 2  10.5.40.250 (10.5.40.250)  13.589 ms  13.575 ms  13.562 ms
 3  10.250.5.1 (10.250.5.1)  13.550 ms  13.537 ms  13.524 ms
 4  10.250.18.1 (10.250.18.1)  13.623 ms  13.827 ms  13.815 ms
 5  router-hostel9.hostel9.iitb.ac.in (10.250.9.2)  13.474 ms  13.461 ms  13.487 ms
 6  * * *
 7  * * *
 8  * * *
 9  * * *
10  * * *
11  * * *
12  * * *
13  * * *
```

Figure 5: Experiment-2 (2) (Traceroute from H5)

Based on the observed histogram, the bottleneck value falls around 500-600 Mbps. This is identified as the mode of the histogram, representing the most frequent throughput observed in our experiment, and thus indicating the bottleneck throughput of the system

3.2.3

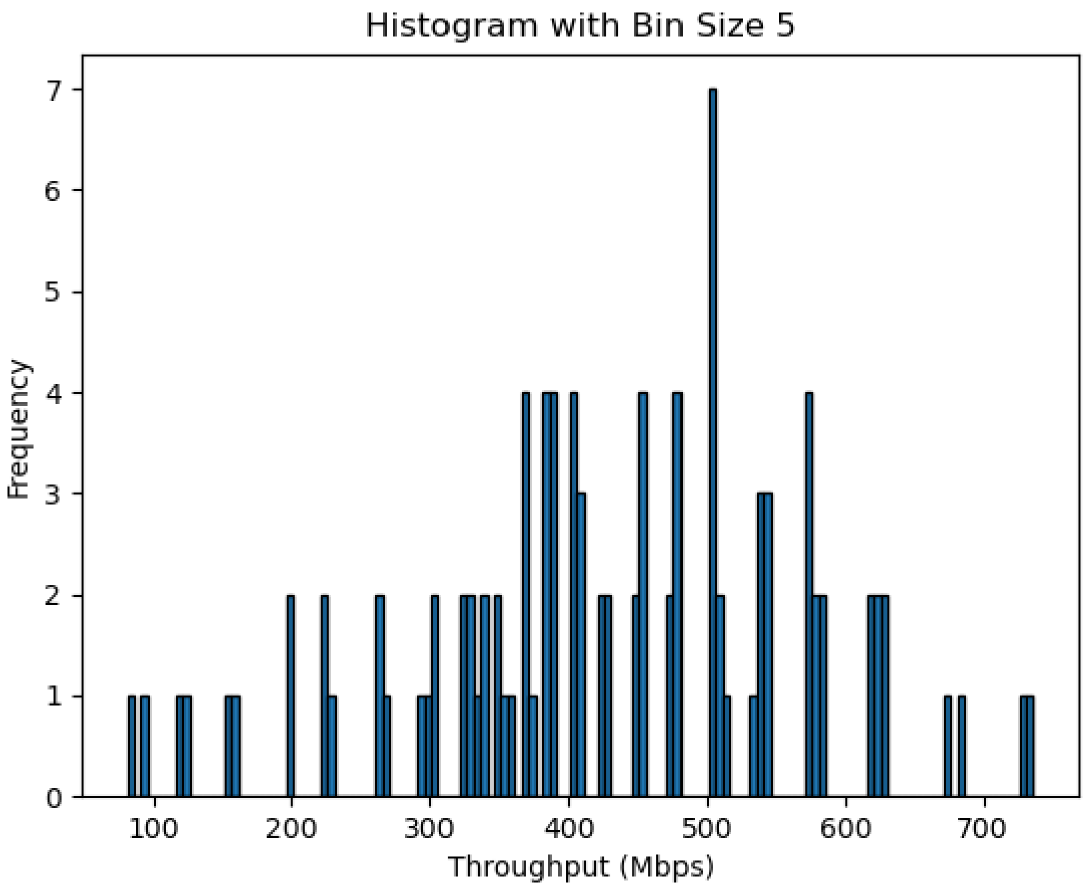
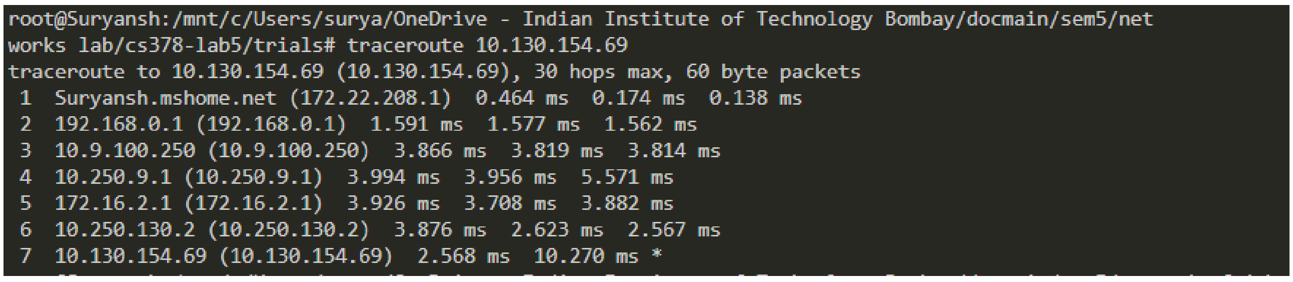


Figure 6: Experiment-2 (3) (SL PC)



```
root@Suryansh:/mnt/c/Users/surya/OneDrive - Indian Institute of Technology Bombay/docmain/sem5/net
works lab/cs378-lab5/trials# traceroute 10.130.154.69
traceroute to 10.130.154.69 (10.130.154.69), 30 hops max, 60 byte packets
 1  Suryansh.mshome.net (172.22.208.1)  0.464 ms  0.174 ms  0.138 ms
 2  192.168.0.1 (192.168.0.1)  1.591 ms  1.577 ms  1.562 ms
 3  10.9.100.250 (10.9.100.250)  3.866 ms  3.819 ms  3.814 ms
 4  10.250.9.1 (10.250.9.1)  3.994 ms  3.956 ms  5.571 ms
 5  172.16.2.1 (172.16.2.1)  3.926 ms  3.708 ms  3.882 ms
 6  10.250.130.2 (10.250.130.2)  3.876 ms  2.623 ms  2.567 ms
 7  10.130.154.69 (10.130.154.69)  2.568 ms  10.270 ms *
```

Figure 7: Experiment-2 (3) (Traceroute to SL PC)

Based on the observed histogram, the bottleneck value falls around 500Mbps. This is identified as the mode of the histogram, representing the most frequent throughput observed in our experiment, and thus indicating the bottleneck throughput of the system

4 References

- Network Sockets:
  `https://beej.us/guide/bgnet/html/`