University of Pisa
Department of Computer Science

MASTER'S DEGREE IN COMPUTER SCIENCE AND NETWORKING

TELETRAFFIC MODELS AND MEASUREMENTS

Academic Year: 2023/2024

Date: 18/09/2024

NETWORK TRAFFIC SIMULATION AND QUEUE LENGTH ANALYSIS USING MININET AND SCAPY

Author

Hassan Shabir

h.shabir@studenti.unipi.it

Abstract

This report details the implementation and analysis of network traffic generation and queue length measurement in a simulated Mininet environment. Using Python's Scapy module, traffic was generated with exponential inter-arrival times and fixed packet lengths, while TCLink was used to limit link bandwidth. Queue length measurements were compared with theoretical M/G/1 queueing model predictions under various bandwidth constraints. Latency and throughput were measured using ping and iperf. The study highlights the effects of bandwidth limitations and traffic intensity on network performance, particularly under high traffic loads.

Contents

List of Figures

Introduction

This project aims to simulate network traffic between hosts to study how different network conditions affect queue lengths and performance. Using Mininet, an open-source network emulator, we set up a virtual network with hosts and switches connected by links with limited bandwidth. Traffic is generated between the hosts using Python's Scapy module, with packets arriving at exponential intervals and having a fixed size. TCLink is used to limit the bandwidth of network links, creating a more realistic network environment.

The main goal of this project is to measure the queue lengths in the network and compare them with theoretical values predicted by the M/G/1 queueing model, which helps us understand how well the network handles traffic under different loads. By comparing the measured queue lengths with the theoretical values, we can see how changes in traffic intensity and bandwidth affect network performance.

This report outlines the setup of the network, the methods used to generate traffic and limit bandwidth, and the comparison between the experimental results and theoretical predictions. The findings will provide insight into how network limitations impact traffic flow and queuing.

Network Topology

This diagram provides a simpler overview of the network components, highlighting the switch, router, and bandwidth limitations between the hosts

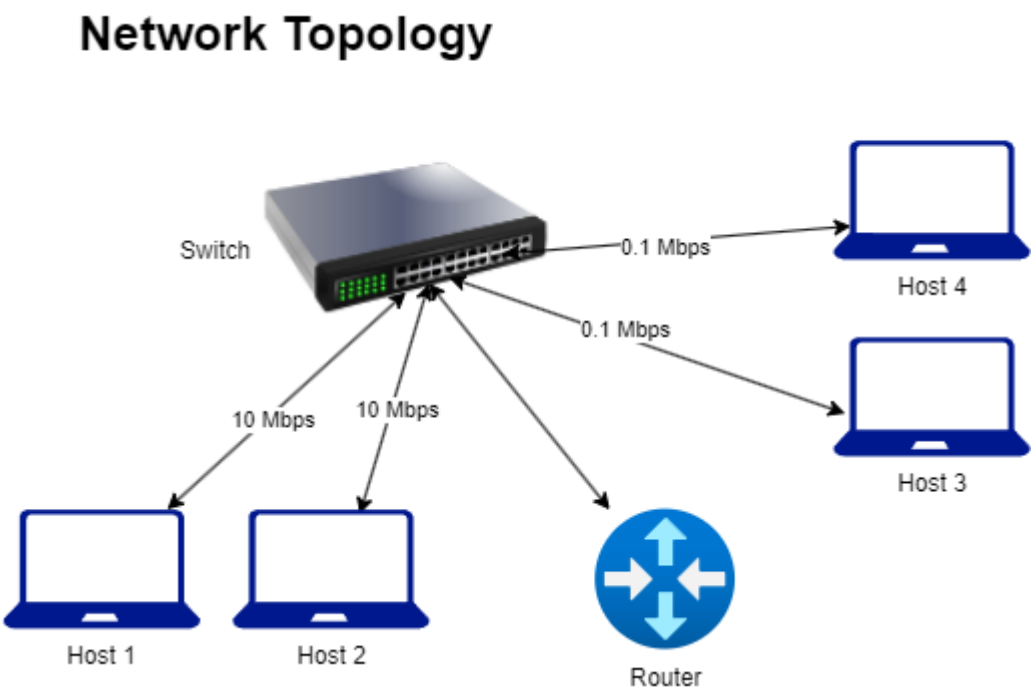

Figure 1: overview of the network components

Detailed Network Topology

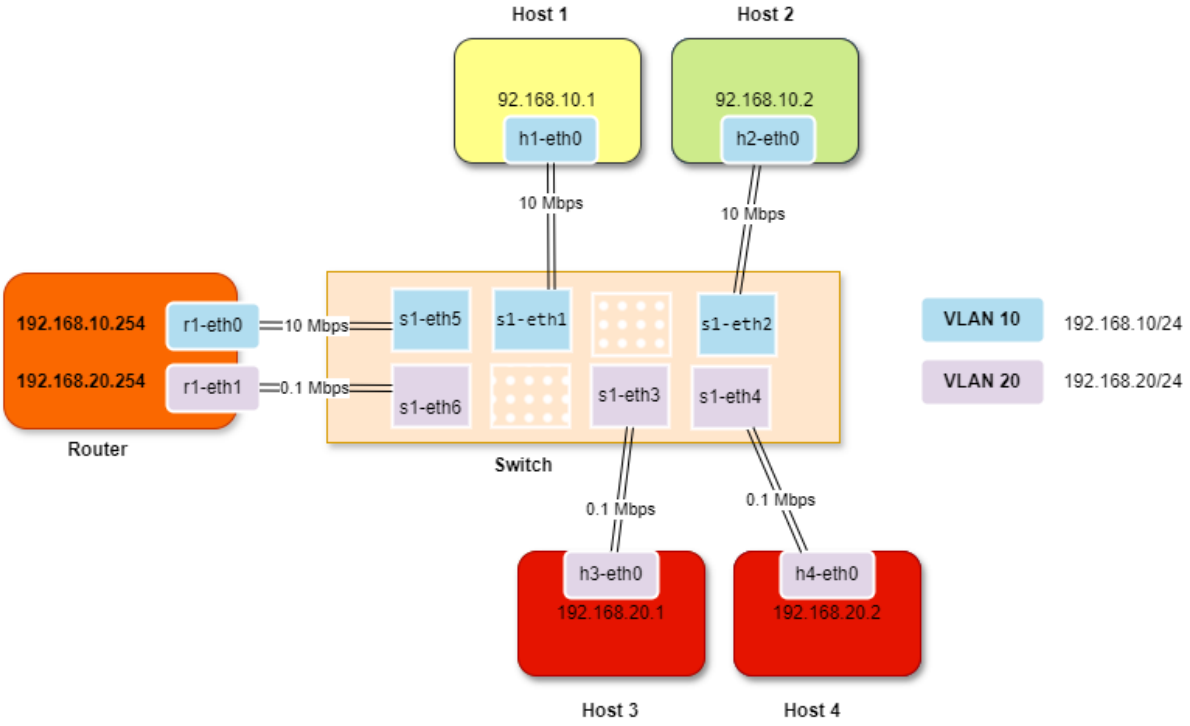

Figure 2: Network Structre Diagram

The network topology for this project simulates a multi-VLAN setup with bandwidth-limited links using Mininet and TCLink. The topology consists of the following components:

1. **Hosts:**

   Four hosts are created, divided into two VLANs:
   - **Host 1 (h1):** IP 192.168.10.1, connected to VLAN 10.
   - **Host 2 (h2):** IP 192.168.10.2, also part of VLAN 10.
   - **Host 3 (h3):** IP 192.168.20.1, connected to VLAN 20.
   - **Host 4 (h4):** IP 192.168.20.2, part of VLAN 20.

2. **Router (r1):**

   The router serves as the gateway between VLAN 10 and VLAN 20. It has two interfaces:
   - **r1-eth0**: Connects to VLAN 10 with IP 192.168.10.254.
   - **r1-eth1:** Connects to VLAN 20 with IP 192.168.20.254.

   The router enables inter-VLAN communication by routing packets between the two subnets.

3. **Switch (s1):**

   All hosts and the router are connected to a single switch (s1). The switch is configured with VLAN tags to ensure that hosts h1 and h2 are part of VLAN 10, while hosts h3 and h4 are part of VLAN 20.
   Each host and the router are connected via different ports on the switch with bandwidth limitations applied using **TCLink**:

   - **Host 1 and Host 2**: Both connected with a bandwidth of 10 Mbps (Ports s1-eth1 and s1-eth2).
   - **Host 3 and Host 4:** Both connected with a bandwidth of 0.1 Mbps (Ports s1-eth3 and s1-eth4).

4. **VLAN Configuration:**

   VLAN tagging is applied on the switch ports to segregate traffic between VLAN 10 (hosts h1, h2, and router interface r1-eth0) and VLAN 20 (hosts h3, h4, and router interface r1-eth1).
   The switch uses Open vSwitch (OVS) to implement VLAN tagging, ensuring that traffic between different VLANs must pass through the router.

```
alessio@labns:~/Downloads/project/Project Hassan$ sudo python3 vlans.py
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2 h3 h4 r1
*** Adding switches:
s1
*** Adding links:
(10.00Mbit) (10.00Mbit) (h1, s1) (10.00Mbit) (10.00Mbit) (h2, s1) (0.10Mbit) (0.10Mbit) (h3, s1) (0.10Mbit) (0.10Mbit) (h4, s1) (r1, s1) (r1, s1)
*** Configuring hosts
h1 h2 h3 h4 r1
*** Starting controller
c0
*** Starting 1 switches
s1 ...(10.00Mbit) (10.00Mbit) (0.10Mbit) (0.10Mbit)
Enabling IP forwarding on the router (r1)...
Configuring routes on hosts...
Ping Test: h1 to h3 (VLAN 10 -> VLAN 20)
h1 -> h3
h3 -> h1
*** Results: 0% dropped (2/2 received)
Ping Test: h2 to h4 (VLAN 10 -> VLAN 20)
h2 -> h4
h4 -> h2
*** Results: 0% dropped (2/2 received)
Ping Test: h1 to h2 (same VLAN 10)
h1 -> h2
h2 -> h1
*** Results: 0% dropped (2/2 received)
Ping Test: h3 to h4 (same VLAN 20)
h3 -> h4
h4 -> h3
*** Results: 0% dropped (2/2 received)
*** Starting CLI:
mininet>
```

Figure 3: Miniet Netwrok

Traffic Generation Using Scapy

The goal of this section is to generate network traffic between two hosts in the simulated environment. To achieve this, we used Python's Scapy library, which allows for low-level network traffic generation and manipulation. In this setup, the traffic was configured to simulate real-world conditions with fixed packet lengths and exponential inter-arrival times. These parameters model typical Poisson traffic, which is often used to represent network packet arrival patterns in real networks.

Packet Specifications:

Packet size: 1460 bytes

Arrival rate: Variable

Traffic pattern: Exponential inter-arrival times, which follow a Poisson distribution, are widely used to simulate random packet arrivals typical in networking scenarios.

Traffic Generation Code (Scapy):

The following code snippet from poisson_scapy_fixed.py demonstrates how we generated packets with fixed sizes and exponential inter-arrival times between two hosts:

```
def generate_packets_fixed_length(src_ip, dst_ip, iface, link_bw, lambda_arrival, packet_length, max_packets=50, backlog_threshold=50000):
    service_rate = (link_bw * 10**6) / (packet_length * 8)  # Service rate in packets/sec
    theoretical_queue_length_initial = calculate_theoretical_queue_length(lambda_arrival, service_rate)

    # Debugging output
    print(f"Service Rate: {service_rate:.2f} packets/sec")
    print(f"Theoretical Queue Length (Initial): {theoretical_queue_length_initial:.4f} packets")

    try:
        while generated_packets < max_packets:
            inter_arrival_time = random.expovariate(lambda_arrival)
            time.sleep(inter_arrival_time)  # Simulating packet arrival

            # Create a fixed-size payload
            packet_data = bytearray(random.getrandbits(8) for _ in range(packet_length))

            # Create the packet (Ethernet + IP + Raw payload)
            packet = Ether() / IP(src=src_ip, dst=dst_ip) / Raw(load=packet_data)

            # Track when the packet is sent (service starts)
            send_start_time = time.time()

            # Calculate waiting time
            waiting_time = max(0, send_start_time - previous_packet_end_time)
            total_waiting_time += waiting_time  # Add waiting time for this packet

            # Send the packet
            send(packet, iface=iface, verbose=False)
```

Figure 4: generated packets with fixed sizes and exponential inter-arrival

Significance of Exponential Inter-Arrival Times:

Using exponential inter-arrival times to generate packets ensures that the traffic mimics real-world networking conditions, where packet arrivals are random but follow a Poisson process. This kind of traffic generation is widely used in networking research because it provides a more accurate representation of unpredictable network load than fixed interval packet generation

Limiting Link Bandwidth with TCLink

To simulate real-world network scenarios where bandwidth is often limited due to various constraints (such as hardware or network policies), TCLink was used to apply bandwidth limitations between the hosts in the Mininet topology. TCLink is a custom Mininet link type that allows fine-grained control over the link's bandwidth, latency, and packet loss rate, making it ideal for this kind of network performance testing.

Bandwidth Limitation Configuration:

In this project, different bandwidth limits were applied to specific links between the hosts and the switch:

- Host 1 (h1) and Host 2 (h2): 10 Mbps bandwidth.
- Host 3 (h3) and Host 4 (h4): 0.1 Mbps bandwidth.

These settings ensure that Hosts 1 and 2, which are part of VLAN 10, operate on a relatively higher bandwidth compared to Hosts 3 and 4 in VLAN 20, which are deliberately bandwidth-constrained to simulate network congestion.

```
# Add links between hosts and switch with bandwidth limits
self.addLink(host1, switch, cls=TCLink, bw=10)  # 10 Mbps for h1
self.addLink(host2, switch, cls=TCLink, bw=10)  # 10 Mbps for h2
self.addLink(host3, switch, cls=TCLink, bw=0.1)  # 0.1 Mbps for h3
self.addLink(host4, switch, cls=TCLink, bw=0.1)  # 0.1 Mbps for h4
```

Figure 5:Bandwidth Limitation Configuration

Iperf Stats for Host1 and Host2

```
root@labns:~/PycharmProjects/HassanProject/project/lab-04# iperf -s -p 5002
------------------------------------------------------------
Server listening on TCP port 5002
TCP window size: 85.3 KByte (default)
------------------------------------------------------------
[  6] local 192.168.10.1 port 5002 connected with 192.168.10.2 port 54540
[ ID] Interval       Transfer     Bandwidth
[  6]  0.0- 5.1 sec  5.38 MBytes  8.87 Mbits/sec
[  6] local 192.168.10.1 port 5002 connected with 192.168.10.2 port 37176
[  6]  0.0-10.2 sec  10.8 MBytes  8.81 Mbits/sec
[  6] local 192.168.10.1 port 5002 connected with 192.168.10.2 port 52468
[  6]  0.0-10.2 sec  11.0 MBytes  9.07 Mbits/sec
[  6] local 192.168.10.1 port 5002 connected with 192.168.10.2 port 53954
[  6]  0.0-10.1 sec  10.6 MBytes  8.80 Mbits/sec
[  6] local 192.168.10.1 port 5002 connected with 192.168.10.2 port 38110
[  6]  0.0- 1.1 sec  1.25 MBytes  9.18 Mbits/sec
[  6] local 192.168.10.1 port 5002 connected with 192.168.10.2 port 49624
[  6]  0.0- 2.2 sec  2.25 MBytes  8.76 Mbits/sec
[  6] local 192.168.10.1 port 5002 connected with 192.168.10.2 port 41774
[  6]  0.0-10.2 sec  10.8 MBytes  8.87 Mbits/sec
[  6] local 192.168.10.1 port 5002 connected with 192.168.10.2 port 47488
[  6]  0.0-10.2 sec  10.8 MBytes  8.81 Mbits/sec
[  6] local 192.168.10.1 port 5002 connected with 192.168.10.2 port 34084
[  6]  0.0-10.2 sec  10.9 MBytes  8.95 Mbits/sec
[  6] local 192.168.10.1 port 5002 connected with 192.168.10.2 port 46498
[  7] local 192.168.10.1 port 5002 connected with 192.168.10.2 port 46502
[  8] local 192.168.10.1 port 5002 connected with 192.168.10.2 port 46514
[  9] local 192.168.10.1 port 5002 connected with 192.168.10.2 port 46520
[  9]  0.0-10.8 sec  2.88 MBytes  2.24 Mbits/sec
[  6]  0.0-10.9 sec  3.00 MBytes  2.31 Mbits/sec
[  7]  0.0-10.9 sec  3.00 MBytes  2.31 Mbits/sec
[  8]  0.0-10.9 sec  3.00 MBytes  2.31 Mbits/sec
[SUM]  0.0-10.9 sec  11.9 MBytes  9.15 Mbits/sec
[  6] local 192.168.10.1 port 5002 connected with 192.168.10.2 port 48344
[  7] local 192.168.10.1 port 5002 connected with 192.168.10.2 port 48358
[  8] local 192.168.10.1 port 5002 connected with 192.168.10.2 port 48368
[  9] local 192.168.10.1 port 5002 connected with 192.168.10.2 port 48378
[ 10] local 192.168.10.1 port 5002 connected with 192.168.10.2 port 48390
[  6]  0.0-10.5 sec  2.25 MBytes  1.79 Mbits/sec
[  7]  0.0-10.5 sec  2.25 MBytes  1.79 Mbits/sec
[  8]  0.0-10.5 sec  2.25 MBytes  1.79 Mbits/sec
[ 10]  0.0-10.5 sec  2.25 MBytes  1.79 Mbits/sec
[  9]  0.0-10.7 sec  2.38 MBytes  1.87 Mbits/sec
[SUM]  0.0-10.7 sec  11.4 MBytes  8.95 Mbits/sec
```

Figure 6: Host1 as Server

```
mininet> h2 iperf -c 192.168.10.1 -p5002
------------------------------------------------------------
Client connecting to 192.168.10.1, TCP port 5002
TCP window size: 85.3 KByte (default)
------------------------------------------------------------
[  3] local 192.168.10.2 port 34084 connected with 192.168.10.1 port 5002
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0-10.1 sec  10.9 MBytes  9.07 Mbits/sec
mininet> h2 iperf -c 192.168.10.1 -p5002 -P 4
------------------------------------------------------------
Client connecting to 192.168.10.1, TCP port 5002
TCP window size: 85.3 KByte (default)
------------------------------------------------------------
[  6] local 192.168.10.2 port 46520 connected with 192.168.10.1 port 5002
[  5] local 192.168.10.2 port 46514 connected with 192.168.10.1 port 5002
[  3] local 192.168.10.2 port 46498 connected with 192.168.10.1 port 5002
[  4] local 192.168.10.2 port 46502 connected with 192.168.10.1 port 5002
[ ID] Interval       Transfer     Bandwidth
[  6]  0.0-10.2 sec  2.88 MBytes  2.36 Mbits/sec
[  5]  0.0-10.4 sec  3.00 MBytes  2.42 Mbits/sec
[  3]  0.0-10.4 sec  3.00 MBytes  2.42 Mbits/sec
[  4]  0.0-10.4 sec  3.00 MBytes  2.42 Mbits/sec
[SUM]  0.0-10.4 sec  11.9 MBytes  9.56 Mbits/sec
mininet> h2 iperf -c 192.168.10.1 -p5002 -P 5
------------------------------------------------------------
Client connecting to 192.168.10.1, TCP port 5002
TCP window size: 85.3 KByte (default)
------------------------------------------------------------
[  3] local 192.168.10.2 port 48344 connected with 192.168.10.1 port 5002
[  5] local 192.168.10.2 port 48368 connected with 192.168.10.1 port 5002
[  4] local 192.168.10.2 port 48358 connected with 192.168.10.1 port 5002
[  6] local 192.168.10.2 port 48378 connected with 192.168.10.1 port 5002
[  7] local 192.168.10.2 port 48390 connected with 192.168.10.1 port 5002
[ ID] Interval       Transfer     Bandwidth
[  4]  0.0-10.1 sec  2.25 MBytes  1.87 Mbits/sec
[  7]  0.0-10.1 sec  2.25 MBytes  1.87 Mbits/sec
[  3]  0.0-10.1 sec  2.25 MBytes  1.86 Mbits/sec
[  5]  0.0-10.2 sec  2.25 MBytes  1.85 Mbits/sec
[  6]  0.0-10.5 sec  2.38 MBytes  1.90 Mbits/sec
[SUM]  0.0-10.5 sec  11.4 MBytes  9.09 Mbits/sec
mininet>
```

Figure 7: Host2 as Client

Iperf Stats for Host3 and Host4

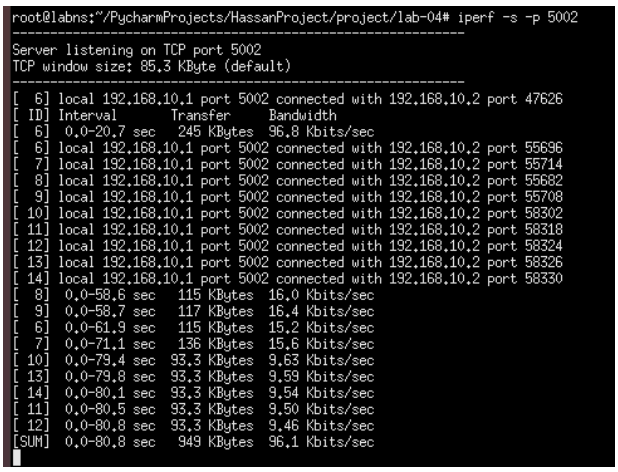

Figure 8: Host3 as Server

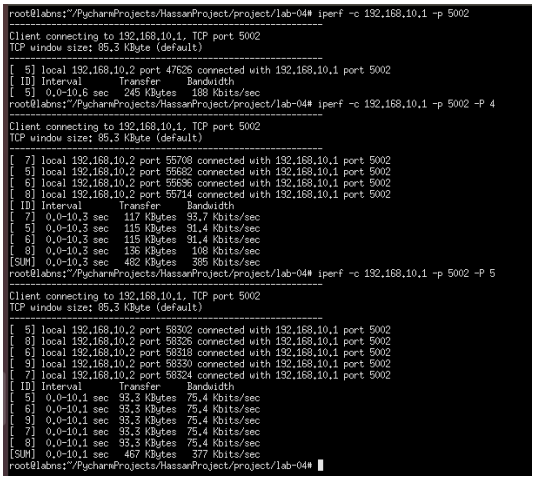

Figure 9: Host 4 as Client

Queue Length Measurements

Theoretical Queue Length Calculation (M/G/1 System with 10 Mbps Bandwidth)

In this project, we calculate the theoretical queue length based on the M/G/1 queueing model, which is used to model queues with a single server and general service times. The calculation is performed using the following inputs for a link with 10 Mbps bandwidth:

- **Link Bandwidth:** 10 Mbps
- **Packet Arrival Rate:** 100 packets per second
- **Packet Size:** 1460 bytes

Explanation of the Calculation:

Service Rate Calculation:

The service rate (μ) is the number of packets that the link can handle per second. To calculate this, we first convert the packet size into bits and then divide the total link bandwidth by the packet size.

- **Link bandwidth:** 10 Mbps = 10,000,000 bits per second
- **Packet size:** 1460 bytes = 1460 × 8 = 11,680 bits per packet

$$Service\ Rate(\mu) = \frac{Bandwidth\ (bps)}{Packet\ Size\ (bits)} = \frac{10{,}000{,}000}{11{,}680} = 856.16\ packets/sec$$

Service Time (T_s)

The service time is the time it takes to transmit one packet over the link, and it is the inverse of the service rate:

$$\text{Service } Time = \frac{1}{\text{Service Rate}} = \frac{1}{856.16} \approx 0.001168 \text{seconds}$$

Traffic Intensity (ρ):
Traffic intensity (ρ) represents the proportion of the system's capacity that is being utilized. It is calculated as the ratio of the packet arrival rate to the service rate:

- **Arrival Rate:** 100 packets/sec
- **Service Rate:** 856.16 packets/sec

$$\rho = \frac{Arrival\ Rate}{Service\ Rate} = \frac{100}{856.16} = 0.1168$$

Queue Length Calculation:
The **queue length** in an M/G/1 system is calculated using the formula:

$$L_q = \frac{\rho^2}{2 \cdot (1 - \rho)}$$

$$\rho = 0.1168$$

$$L_q = \frac{0.1168^2}{2 \cdot (1 - 0.1168)} = \frac{0.0136}{1.7664} = 0.0077 \text{packets}$$

Interpretation:
The calculated queue length of 0.0077 packets indicates that, on average, fewer than one packet is expected to be waiting in the queue. This low value reflects that the traffic intensity is relatively low (approximately 11.68%), meaning the system has sufficient capacity to handle the incoming traffic with very little queue buildup.

Simulation Results and Analysis for 10 Mbps Bandwidth

The following results were obtained from the simulation, which was run with a packet arrival rate of 100 packets per second and a link bandwidth of 10 Mbps. The traffic generation used fixed packet sizes of 1460 bytes, and the queue length was monitored throughout the simulation. Below is a detailed breakdown of the results for the 50th packet, followed by a summary of the overall performance.

```
### Packet 50 ###
Packet size: 1460 bytes
Service time: 0.001168 seconds
Waiting time before transmission (in system): 0.082601 seconds
Total time spent in the system: 0.083769 seconds
Elapsed time since start: 3.77 seconds
Current packet arrival rate: 13.26 packets per second
Average waiting time (across all packets): 0.073122 seconds
Current backlog size: 0 bytes
Theoretical queue length (initial parameters): 0.0077 packets
Theoretical queue length (current observed rate): 0.0001 packets
Actual queue length: 0.0000 packets

### Final Summary ###
Total packets sent: 50
Average packet length: 1460.00 bytes
Final packet arrival rate: 13.24 packets per second
Average waiting time (across all packets): 0.073122 seconds

Final Transmission Queue Stats:
qdisc htb 5: root refcnt 2 r2q 10 default 1 direct_packets_stat 0 direct_qlen 1000
 Sent 442440 bytes 5608 pkt (dropped 0, overlimits 0 requeues 0)
 backlog 0b 0p requeues 0

Exiting gracefully.
```

Figure 10: Simulation Results 10 Mbps

the system operated efficiently with no dropped packets or backlog buildup. The use of TCLink to limit the bandwidth effectively simulated real-world conditions, while the low traffic intensity (ρ ≈ 0.1168) ensured that the system remained stable and there was almost no queuing.

Theoretical Queue Length Calculation (M/G/1 System with 0.1 Mbps Bandwidth)

This section calculates the theoretical queue length for an M/G/1 system, based on a link bandwidth of **0.1 Mbps**. The input parameters are as follows:

- **Link Bandwidth:** 0.1 Mbps
- **Packet Arrival Rate:** 100 packets per second
- **Packet Size:** 1460 bytes

Service Rate Calculation:

The service rate (μ) is the number of packets that the link can serve per second. To calculate this, we convert the packet size to bits and then divide the link bandwidth by the packet size:

- **Link bandwidth:** 0.1 Mbps = 100,000 bits per second
- **Packet size:** 1460 bytes = 1460 × 8 = 11,680 bits per packet

$$Service\ Rate(\mu) = \frac{Packet\ Size\ (bits)}{Bandwidth\ (bps)} = \frac{100{,}000}{11{,}680} = 8.57\ packets/sec$$

Service Time (Ts):

The service time is the time it takes to transmit one packet over the link, which is the inverse of the service rate:

$$\text{Service } Time = \frac{1}{\text{Service Rate}} = \frac{1}{8.57} = 0.1168 \text{ seconds}$$

Traffic Intensity (ρ):
Traffic intensity (ρ) is a measure of how much of the system's capacity is being utilized. It is calculated as the ratio of the arrival rate to the service rate:

- **Arrival Rate:** 100 packets/sec
- **Service Rate:** 8.57 packets/sec

$$\rho = \frac{Arrival\ Rate}{Service\ Rate} = \frac{100}{8.57} = 11.68$$

Queue Length Calculation:
The queue length in an M/G/1 system is calculated using the following formula:

$$L_q = \frac{\rho^2}{2 \cdot (1 - \rho)}$$

However, in this case, the traffic intensity ρ=11.68 is greater than 1, meaning the system is overloaded. When ρ >1, the system cannot process packets as quickly as they arrive, leading to an **infinitely growing queue**.

$$Lq\ becomes\ infinite\ because\ \rho > 1$$

Interpretation:
Since the traffic intensity exceeds 1, the system is in an **overloaded state**. This means that the packet arrival rate (100 packets/sec) is significantly higher than the service rate (8.57 packets/sec), causing the queue to grow without bound. In practical terms, the system will not be able to process packets quickly enough, resulting in continuous queuing and significant delays.

This scenario demonstrates the importance of ensuring that the arrival rate remains below the service rate (ρ < 1) to maintain a stable system. When ρ>1, as seen in this case, the queue becomes infinite, and the network experiences severe congestion.

Simulation Results and Analysis for 0.1 Mbps Bandwidth
In this scenario, the system was simulated with a link bandwidth of **0.1 Mbps** to observe the behaviour of the network under low-bandwidth conditions. Below are the key results for the 50th packet and a summary of the overall simulation. The simulation was designed with a packet arrival rate of 100 packets per second, while the observed rate was lower due to bandwidth limitations.

```
### Packet 50 ###
Packet size: 1460 bytes
Service time: 0.116800 seconds
Waiting time before transmission (in system): 0.000000 seconds
Total time spent in the system: 0.116800 seconds
Elapsed time since start: 3.61 seconds
Current packet arrival rate: 13.87 packets per second
Average waiting time (across all packets): 0.001115 seconds
Current backlog size: 28652 bytes
Theoretical queue length (initial parameters): inf packets
Theoretical queue length (current observed rate): inf packets
Actual queue length: 19.6247 packets

### Final Summary ###
Total packets sent: 50
Average packet length: 1460.00 bytes
Final packet arrival rate: 13.85 packets per second
Average waiting time (across all packets): 0.001115 seconds

Final Transmission Queue Stats:
qdisc htb 5: root refcnt 2 r2q 10 default 1 direct_packets_stat 0 direct_qlen 1000
 Sent 48496 bytes 55 pkt (dropped 0, overlimits 30 requeues 0)
 backlog 28652b 19p requeues 0

Exiting gracefully.
mininet>
Interrupt
mininet>
```

Figure 11:Simulation Results 0.1 Mbps

this simulation highlights the significant impact of low bandwidth on network performance. While the system's actual queue length and backlog did not grow infinitely due to practical simulation limits, the theoretical predictions highlight the risk of congestion when the traffic intensity exceeds the system's capacity. In real-world scenarios, such overload conditions could lead to severe congestion, packet loss, and delayed transmissions.

Conclusion

In this project, we explored the simulation of network traffic and queue length measurement using Mininet, Python's Scapy, and TCLink. The objective was to understand how different traffic intensities and bandwidth limitations affect network performance, particularly focusing on queue buildup and system behaviour.

The results showed that when bandwidth was limited, such as in the **0.1 Mbps** scenario, the network quickly became overloaded. The traffic intensity was much higher than the system could handle, which caused the queue length to grow significantly. Although the theoretical queue length was calculated to be infinite in such cases, the actual queue in the simulation was kept under control, showing the effect of real-world constraints like buffering and system limitations.

In contrast, when higher bandwidth (e.g., **10 Mbps**) was available, the system operated smoothly with minimal queuing, as expected. The theoretical and observed results closely matched, confirming that when traffic intensity remains below the system's capacity, queuing is negligible.

This project gave me practical experience with network emulation tools and a better understanding of how bandwidth and traffic loads impact network performance. Simulating real-world scenarios like these helped us see the importance of traffic management and how network configurations can affect packet delays, throughput, and overall efficiency.