Christopher Douangnaly
SID: 998541781
ECS 152A / EEC 173A

Project -- Phase II
Simulation Analysis of Token Passing Protocol based Local-Area Network (LAN)

Contents

Brief Implementation Logic 2
Analysis: Throughput 6
Analysis: Average Packet Delay 8

Brief Implementation Logic

High-level overview:

- User inputs N number of hosts and max_packets
    - max_packets represents the number of packets each host will simulate
    - Simulation will terminate once a total of (N hosts * max_packets) packets have been sent
- During initialization
    - Create a ring of N servers
    - Each server is initialized with exactly one packet
        - These packets will arrive at each server's queue at a random arrival_time, so the packets might not be in the queue right when simulation begins.
    - Randomly select a host to hold the token
- Simulation loop begins at sim_time = 0 and will run for each value of lambda
    - If token holder has packets to transmit
        - For each packet in the queue
            - Accumulate: bytes sent, delays, sim_time, packets sent
            - Transmit the packet
                - (We can simulate the data doing a round trip by calculating the delays associated with transmitting the packet circulating N hosts in the ring).
        - Pass token to next server.
    - If token holder has no packets to transmit
        - Pass token to next host
    - If token has been passed for N times and zero packets have been sent
        - This means that the arrival time of the next packet is probably far away from the current simulation time
        - Calculate the number of rounds the token will need circulate the ring before a token_owner has actual data to send
        - Jump to new simulation time
        - If it weren't for this, the simulation would take a really long time to simulate
- Simulation terminates once (N hosts * max_packets) packets have been sent
- Accumulate *throughput* and *average packet delay* using the variables mentioned above
- Repeat & run the simulation for all values in the lambda array.

Sample Output (Without Comments, Full Output, All Lambda Values)

Set number of servers in the ring: 25
Set packet count per server to simulate. Terminates when (packetCount * N hosts) are sent. 30

[Initialization]

[initialization complete. 1 packet has been initialized at each of the 25 servers]
[simulation terminates when a total of (max_packets * N hosts) = 750 packets have been sent]

*** End of simulation for lambda = 0.01 -> max # packets sent of 750 reached.

** RESULTS **
lambda = 0.01 :
Throughput = 183.241
Average packet delay = 0.00102406

[Initialization]

[initialization complete. 1 packet has been initialized at each of the 25 servers]
[simulation terminates when a total of (max_packets * N hosts) = 750 packets have been sent]

*** End of simulation for lambda = 0.05 -> max # packets sent of 750 reached.

** RESULTS **
lambda = 0.05 :
Throughput = 957.776
Average packet delay = 0.001084

[Initialization]

[initialization complete. 1 packet has been initialized at each of the 25 servers]
[simulation terminates when a total of (max_packets * N hosts) = 750 packets have been sent]

*** End of simulation for lambda = 0.1 -> max # packets sent of 750 reached.

** RESULTS **
lambda = 0.1 :
Throughput = 2047.42
Average packet delay = 0.00108305

[Initialization]

[initialization complete. 1 packet has been initialized at each of the 25 servers]
[simulation terminates when a total of (max_packets * N hosts) = 750 packets have been sent]

*** End of simulation for lambda = 0.2 -> max # packets sent of 750 reached.

** RESULTS **
lambda = 0.2 :

Throughput = 3891.9
Average packet delay = 0.00108475

[Initialization]

[initialization complete. 1 packet has been initialized at each of the 25 servers]
[simulation terminates when a total of (max_packets * N hosts) = 750 packets have been sent]

*** End of simulation for lambda = 0.3 -> max # packets sent of 750 reached.

** RESULTS **
lambda = 0.3 :
Throughput = 5894.64
Average packet delay = 0.00106715

[Initialization]

[initialization complete. 1 packet has been initialized at each of the 25 servers]
[simulation terminates when a total of (max_packets * N hosts) = 750 packets have been sent]

*** End of simulation for lambda = 0.5 -> max # packets sent of 750 reached.

** RESULTS **
lambda = 0.5 :
Throughput = 10272.6
Average packet delay = 0.00104995

[Initialization]

[initialization complete. 1 packet has been initialized at each of the 25 servers]
[simulation terminates when a total of (max_packets * N hosts) = 750 packets have been sent]

*** End of simulation for lambda = 0.6 -> max # packets sent of 750 reached.

** RESULTS **
lambda = 0.6 :
Throughput = 12428.1
Average packet delay = 0.00111426

[Initialization]

[initialization complete. 1 packet has been initialized at each of the 25 servers]
[simulation terminates when a total of (max_packets * N hosts) = 750 packets have been sent]

*** End of simulation for lambda = 0.7 -> max # packets sent of 750 reached.

** RESULTS **
lambda = 0.7 :
Throughput = 13148.6
Average packet delay = 0.00106022

[Initialization]

[initialization complete. 1 packet has been initialized at each of the 25 servers]
[simulation terminates when a total of (max_packets * N hosts) = 750 packets have been sent]

*** End of simulation for lambda = 0.8 -> max # packets sent of 750 reached.

** RESULTS **
lambda = 0.8 :
Throughput = 16518.2
Average packet delay = 0.00111119

[Initialization]

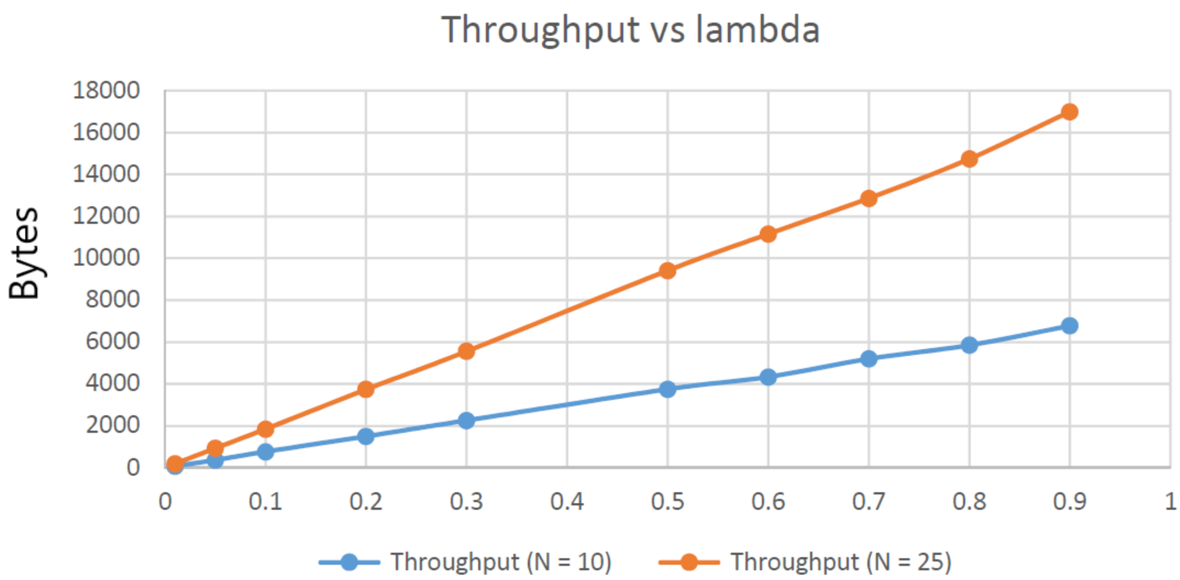
[initialization complete. 1 packet has been initialized at each of the 25 servers]
[simulation terminates when a total of (max_packets * N hosts) = 750 packets have been sent]

*** End of simulation for lambda = 0.9 -> max # packets sent of 750 reached.

** RESULTS **
lambda = 0.9 :
Throughput = 18953.9
Average packet delay = 0.00114385

Type anything and press ENTER to exit simulation ...

Analysis: Throughput



- `double throughput = ((double)total_bytes_count) / sim_time;`
- Throughput is how much data is sent successfully up until termination time. We will not consider the bits to pass the token since this is not considered goodput.
- Throughput is affected by many variables, such as:
    - Arrival_time (see example in next page):
        - Is the random time in the future a specific packet will arrive at a host before it can be forwarded to next host.
        - Since packets choose to arrive to a queue “independently on their own”, a token_owner at the current sim_time can only send if it has packets in its queue; thus, the random arrival_time (which depends on lambda) limits the throughput the ring protocol can process within any period of sim_time.
    - `double average_delay = total_delay / total_packets;`
        - These are the delays associated with sending the packet to its destination, which is dependent on the number of hops it takes to get there.
        - For example, there are more delays to send a packet from Host 1 to 6 than from Host 1 to 2.
            - Also, since destinations are randomly generated, this, too, will affect overall throughput, since the number of hops will vary.
    - As we will see, the first has a more direct impact than the second.
- We can apply the above points to the graph.
    - The takeaway is that with increasing lambda, the greater the probability that a host, when it becomes the token_owner at the current sim_time, will have packets to transmit. Thus, throughput increases with increasing lambda, at a linear growth rate. These conditions apply equally to both scenarios of either 10 hosts or 25 hosts, except that at 25 hosts, more data is being transferred successfully within the same time frame.
    - Let’s take an example of the arrival times <u>during initiation</u>, and then look at the throughput <u>at end of simulation</u>, for two simulations total at lambda = 0.05 and at lambda = 0.09.

[Initialization]
arrival_time: 21.4672
arrival_time: 8.47642
arrival_time: 26.2146
arrival_time: 18.1121
arrival_time: 18.8048
arrival_time: 6.0653
arrival_time: 58.631
arrival_time: 10.6077
arrival_time: 5.07911
arrival_time: 34.7252
[initialization complete. 1 packet has been initialized at each of the 10 servers]
*** End of simulation for lambda = 0.05 -> max # packets sent of 10 reached.

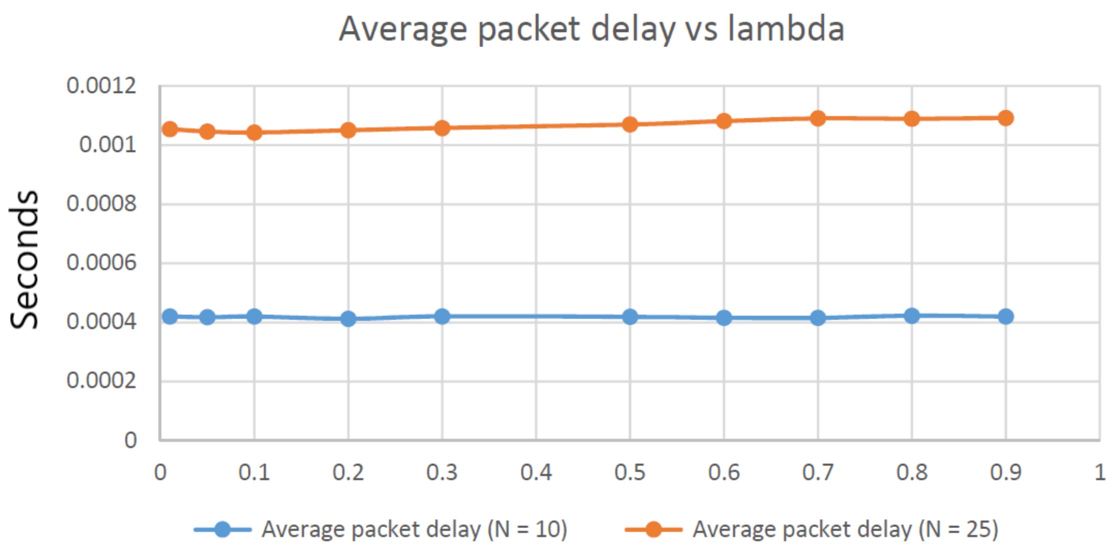
** RESULTS **
lambda = 0.05 :
Throughput = 351.145
Average packet delay = 0.000405535

[Initialization]
arrival_time: 2.85375
arrival_time: 0.0489763
arrival_time: 0.992724
arrival_time: 0.418647
arrival_time: 1.81479
arrival_time: 1.99519
arrival_time: 1.87051
arrival_time: 1.51578
arrival_time: 1.93555
arrival_time: 1.38962
[initialization complete. 1 packet has been initialized at each of the 10 servers]
*** End of simulation for lambda = 0.05 -> max # packets sent of 10 reached.

** RESULTS **
lambda = 0.9 :
Throughput = 3249.65
Average packet delay = 0.000288365

- At lambda = 0.05, the arrival_time of the packets at each host is much **further** away from the current sim_time = 0. As a result, the overall throughput is much less than at lambda = 0.9.
- At lambda = 0.9, with an increased packets/second threshold, the arrival_time of the packets at each host is much **closer** to the current sim_time = 0. As a result, the overall throughput is greater, because there is a greater chance of a token_owner having a non-empty queue at current simulation time.

Analysis: Average Packet Delay



For each packet sent, the average packet delay is accumulated as follows:

```
Total delay += (sim_time – packet.arrival_time) + hop_count * (d_prop + d_trans)
```

Where (sim_time – packet.arrival_time) represents the queueing delay, and where the other two delays are dependent on the number of hops between token_owner and destination host. Note that total_delay accumulates delays between sender and receiver, not round-trip delays. A different variable, sim_time, will accumulate such delays for the *round-trip* so that the token_owner can release the token after sending all packets in its queue. Sim_time also represents the current cumulative simulation time with respect to delays.

So, the total average delay is calculated after all simulations (lambdas) are completed as:

```
double average_delay = total_delay / total_packets;
```

Relationship between N = 10 and N = 25:

- Increasing the ring size potentially increases hop_count, the number of links between token_owner and receiver. This increases the overall average delay as the packet might need to traverse a greater amount of links to reach the destination.
    - Thus, the plot of N = 25 has a greater average packet delay over all values of lambda versus the plot of N = 10.

Relationship between increasing lambda values and number of hosts:

- Here, the differences are more difficult to spot. It seems that whether the ring has 10 hosts or 25 hosts, as lambda values increase, the average packet delay stays relatively the same.
- Since this simulation in this graph sends the same amount of packets for both N = 10 and N = 25, let's see what happens when we increase the number of packets **within** the same system, and compare their average delays.