Final Report

ABSTRACT (CSMA/CD)

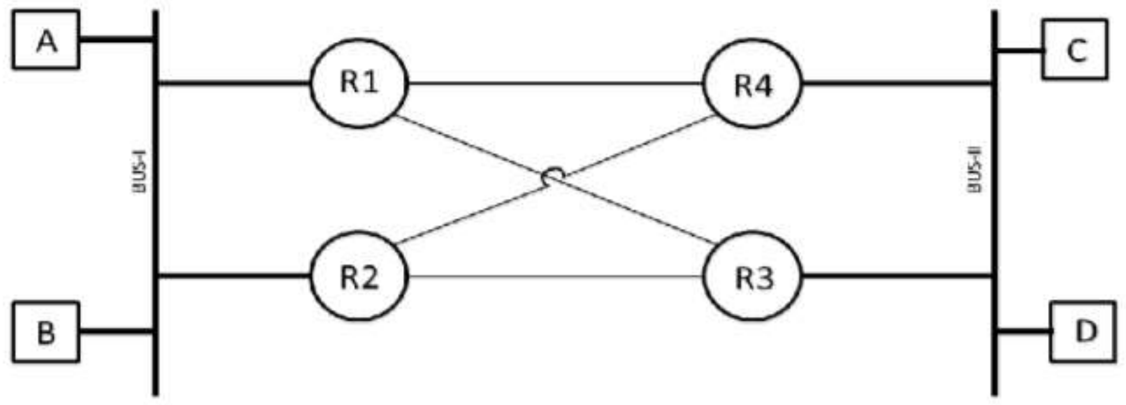
LAN can be defined as a network which spans over a small geographical area. There will be many number of independent devices, can be personal computers, laptops, printers etc. in the LAN network connected by a single bus (in most cases Ethernet) which want to communicate with each other. Since all the devices in the network share the same medium, only one device can use the bus at any given time. So we must have a protocol to determine who gets to transmit next. For this we use a MAC layer protocol which uses both Carrier Sensing and Multiple Access.

Ethernet uses CSMA/CD as its media access control protocol. It uses a carrier sensing scheme in which the transmitting station senses the medium and detects if any other transmission or collision happens. After a collision is detected, the transmission is stopped immediately and the station waits for a random amount of time before retransmitting, hence improving the overall performance.

OBJECTIVE

Implement a CSMA/CD based medium access control scheme and a routing protocol to deliver data between any pair of end nodes for a simulation time of 30 seconds, that is, to enable end to end data delivery in a virtual network whose architecture is as shown below:

NETWORK ARCHITECTURE



In the above network topology we have two LANs separated by a network of routers. Each LAN has two nodes which share a common Ethernet bus and are connected using coaxial cables. The distance between the nodes in a LAN is 2000 meters and speed of the bus is 100 Mbps. The distance between adjacent routers is also 2000 meters and they are connected with fiber optic cables which can transmit at a rate of 1Gbps.

From the above network it can be observed that nodes A,B are in LAN1 and sharing BUS 1 and nodes C,D are in LAN2 and share BUS 2. The propagation speed in Ethernet, coaxial cable and fiber optic cable is 2 x 10^8 m/s. So our objective is to establish connection between the nodes and start transmitting data using CSMA/CD protocol.

INTRODUCTION

Network Simulation

The protocol is implemented by using discrete event simulation which considers the whole system as a discrete sequence of events. It is assumed that no changes occur between two events and hence the clock jumps in time from one event to the next event.
The simulation clock keeps track of the current simulation time. The packet generation at the nodes, transmission of the packets and the received time of the packets are all tracked by the clock. At every event, the clock value is updated by adding the appropriate delay.
The simulation clock also helps in regulating the total simulation time of the algorithm.

Software Platform

The simulation has been implemented using MATLAB. It is an abbreviation for *Matrix Laboratory*. It is designed to operate primarily on matrices and arrays. Hence, our implementation of the simulation system (List of nodes, bus 1, bus 2, cost matrix and the simulation list) is based on Arrays and Matrices.
It has various in-built functions which aid in plotting the final results and maintaining a sorted list of all the events during simulation.

DESIGN AND IMPLEMENTATION

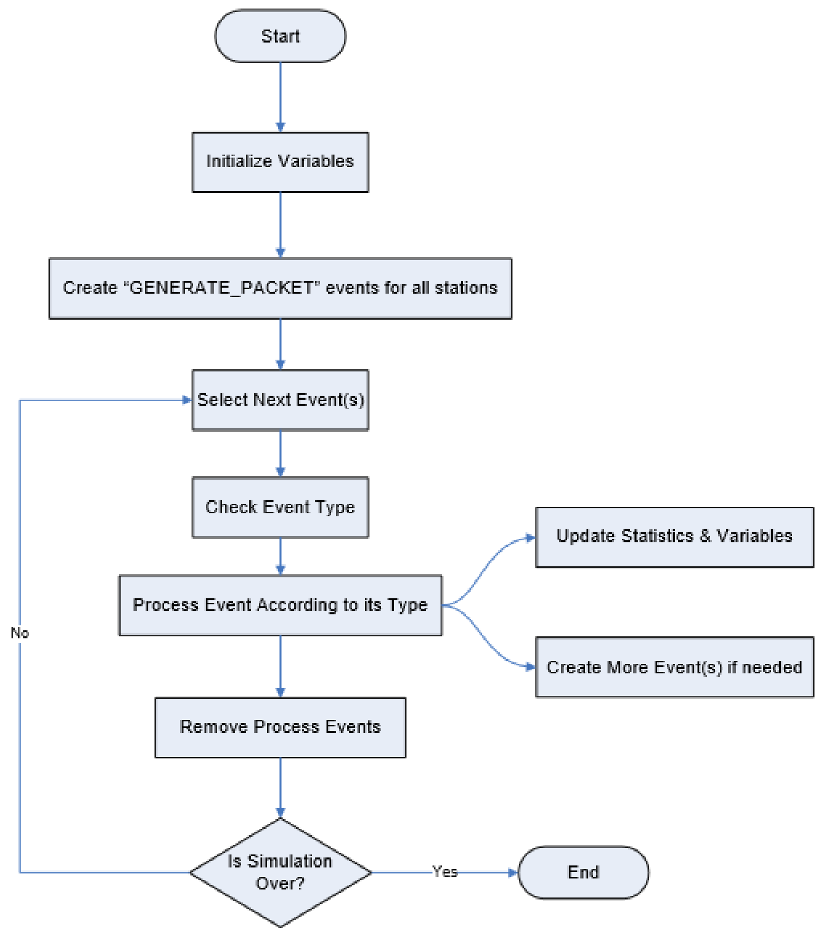
The implementation was carried on in three phases. The initial phases involved implementation of CSMA/CD for N nodes which share the same bus in the network. The nodes contend to access the channel, while the destination node was chosen randomly with equal probability.
The final implementation has a different network topology with two different buses which connect 2 nodes each to the network of routers.

Single-Threaded Event Driven Approach: Our simulation is programmed to process different events and generate reports from the updated network statistics using one thread. i.e. the program progresses in a sequential manner. Packet generation and packet transmission both are handled by the same thread, hence both the processes cannot take place at the same time. The program has a while loop which manages all the processes and also adds/removes the events from the list. The loop terminates when the simulation time is reached.

During our implementation, we have made the following ***assumptions***:

- No link or router failures occur during simulation.
- Every router has infinite buffer, hence no packets are lost during transmission.
- Cost of the link covers all possible factors such as distance, bandwidth and traffic load.
- Priority of the packets is not considered.
- The packets in the queue are transmitted on a FCFS basis.
- The same serves as a transmitter as well as a receiver for the packets.
- The packets are deleted from the queue immediately after it is transmitted successfully.

The basic design of our system is explained in the flowchart below:


Start
Initialize Variables
Create "GENERATE_PACKET" events for all stations
Select Next Event(s)
Check Event Type
Process Event According to its Type
Update Statistics & Variables
Create More Event(s) if needed
Remove Process Events
Is Simulation Over?
No
Yes
End

<ins>Part 1 and Part 2 of implementation –</ins>

1. **Initializing parameters:**
- An array which contains the node name and ID mapping
- An array to store the following details with regard to every packet(state) that is generated or is present in the queue: (Source ID, Destination ID, Generation time, Transmission time, Time at which packet is received at destination(Receive time, current time, number of collisions before successful transmission)
- A pointer for the generation time of most recent packets at all the N nodes.
- An array for storing the packets that are simulated
2. **Packet creation:** There are packets from every node i.e.; N nodes which will contend for the channel access at any time. Since all the nodes share the same bus, we have a single queue which maintains the packets to be transmitted from source node to destination node. In order to create a new packet, we calculate the inter-arrival time between the packets and update the generation time for respective packets.
3. To initiate the packet transmission, we consider the packet at the top of our queue to be the sent from the node at which it was created. The destination node is selected randomly from the list of nodes (except the source node).
4. After a packet is created, we delay the packets by the total delay time. We do so because it might be the case that when a packet is generated, another packet is already

being transmitted, so the new packet will have to wait until the previous one reaches the destination.

5. We update the transmission time of the packet with the current time and the received time by adding the propagation and transmission delay.
6. If the time difference of the current time of the two packets in our queue is greater than the propagation delay, collision will not occur. We then update the transmission time of the topmost packet in our queue with the current time and the received time of the packet by adding the total delay.
7. In case of collision, we update the transmission time of both the packets with current time and number of collisions by 1. A random number between 0 and 2^n - 1 (n - number of collisions) is chosen and multiplied by the backoff time. The packets transmission is then delayed by the calculated backoff time.
8. We check if there are any packets which were generated at a time greater than our simulation time, we stop if we encounter any such packets. Otherwise, we continue to generate packets repeat the steps.
9. The simulation time is tracked by keeping record of the current time. Clock value is updated at every packet transmission as the simulation proceeds.

Part 3 of Implementation –

1. Initializing Parameters:

- An array with the node Id and node name mapping.
- Two arrays/lists corresponding to the queues in bus 1 and bus 2 which stores the information regarding the state of the packets. (Source ID, Destination ID, Generation time, Transmission time, received time, Current time, Number of collisions before successful transmission)

2. Create packets for all the nodes. If the packet belongs to Node A or Node B, add it to the list 1 otherwise to list 2.
3. Randomly choose the destination node, with equal probability. If the source and destination nodes are not connected to the same bus, use routing protocol implementation to choose the path taken by the packet and delay calculation. If they are connected to the same bus, use the normal transmission
4. Check for collision scenarios, if there is no collision, update the transmission and received time for the successful transmission of the packet. In case of collision, we update the transmission time of both the packets with current time and number of collisions by 1. A random number between 0 and 2^n - 1 (n - number of collisions) is chosen and multiplied by the backoff time. The packets transmission is then delayed by the calculated backoff time.
5. Repeat the process of packet generation and transmission until the simulation time is reached. This time the simulator chooses the packet at the head of the sorted list first and adds the newly generated packet to the appropriate position.
6. Calculate statistics like number of packets sent, queueing delay, access delay and average throughput between all possible pair of nodes in the network and generate corresponding graphs.

Generation of random numbers and Poisson distribution:
In the simulation, the inter-arrival time of the packets follow Poisson distribution. All the nodes generates frames of 1000-bytes using Poisson distribution with a mean value of
λ = 0.5/ frame time and with a frame slot of 500 μs

Collisions and Exponential Back-off:
Collision occurs when two nodes try to transmit data at the same time. After collision occurs, all nodes using the same bus stops transmitting immediately, waits for a back-off time which is obtained by the back-off algorithm. The back-off algorithm uses uniform distribution to determine the backoff slots. As an example, consider there are N collisions, the nodes will have to choose the backoff slots from the range of [0, 2^n-1]. After selecting a value, the node has to wait for the time equal to the product of the slot value and the frame slot.
The random waiting time after consecutive collisions results in increase in average delay. After first collision, waits for 0 or 1 slot time. If second collision happens, wait time slot increases to 0, 1, 2, 3 and so on.

Uniform Distribution:
The selection of source and destination nodes is done randomly using uniform distribution. The costs of links between the routers varies in the range of [1, 10] for a time period using uniform distribution. The cost of a link covers all possible factors to be considered such as distance, bandwidth, traffic load, etc. The exponential back-off algorithm also uses uniform distribution.

Updated Channel Access:
The channel is accessed by any of the nodes which has a packet on top of the queue.

Choice of Routing Protocol:
We need the routing protocol to route the packets between two different LANs and we don't need the routing if the source and destination are in the same LAN. Since, the network is small and we have the knowledge of all the nodes, edges and costs in our topology, we have implemented a routing protocol that uses *Dijkstra's* algorithm. We have used the original variant of the algorithm which finds the shortest path from one source to one destination, and calculated the shortest cost path between any pair of nodes which are in different LANs.

Algorithm:
1. Create an adjacency matrix which indicates the connections between the routers.
2. Uniformly assign costs (in the range 1 to 10) to every link in the network.
3. Iterate over the list of nodes, considering every node as a source node.
4. For every source node, consider all of the neighbouring nodes and calculate their tentative costs. Compare the new cost values with the previous ones and store the minimum of the two.
5. Once, all the adjacent nodes have been covered, mark the source node as discovered and consider the next undiscovered node from the list as source. Repeat the same calculation.
6. Once all the nodes in the list are marked as visited, we have the least cost paths calculated for all possible pair of nodes. The algorithm will terminate.

Progress of Routing Table
The routing table is updated every 2 ms. During the simulation, we set the timer at the start of transmission. When the packet is to be routed over the network, we check the value of the timer and re-compute the shortest cost path, if more than 2 ms have elapsed since the last transmission. The timer is then reset for the next round. We stop the timer once the simulation is over.

Simulation Result

Part 1

We run the simulation for 30 seconds, which is the amount of time for which the bus is shared between the nodes and with a frame time slot of 500µs. We calculate statistics like no. of packets sent, average access delay, average queueing delay, average throughput from one source node to another destination node**.**

The average access delay is calculated as the delay from the beginning of transmission of a packet in its first attempt to the end of its successful transmission. The average queueing delay at a node is given as the time between the generation and its transmission onto the channel. Throughput is calculated as the number of successful transmitted packets over the total end to end delay.

	Node A	**Node B**
Total number of Packets sent	29876	30289
Average frame interval (µs)	1.003938e+03	9.904071e+02
Average access delay (µs)	1.242408e+02	1.242710e+02
Average queuing delay (µs)	9.181196e+01	9.701945e+01
Average end to end throughput (Mbps)	4.400151e+07	4.277630e+07

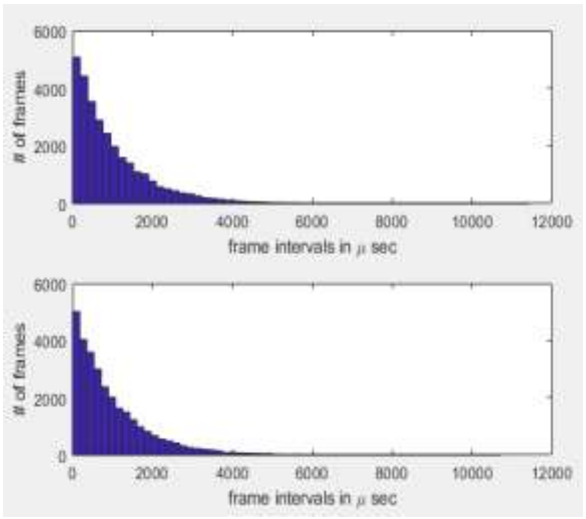


Fig. 1

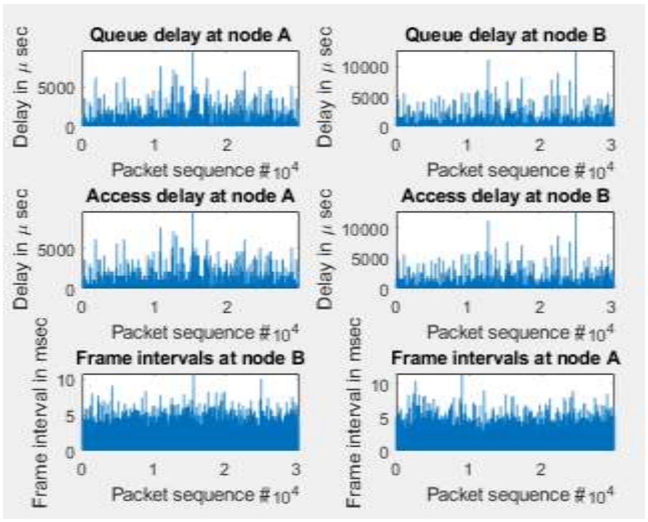


Fig. 2

Part 2

We ran the simulation for 30 sec and calculated the statistics for every pair of source and destination nodes. We expanded our implementation to 4 nodes, therefore we have 12 combinations of S and D which we have depicted in the form of a table.
As we can see the number of packets generated at node A are now distributed between three destinations, hence as compared to Part 1 (where we had 29876 packets for Node A), Part 2 appears to have equal distribution of packets (9952, 9983, 10029) for Node B, C and D respectively.

Number of Packets sent:

From/To	**Node A**	**Node B**	**Node C**	**Node D**
Node A	0	9952	9983	10029
Node B	10087	0	10103	10038
Node C	9997	10114	0	9954
Node D	9954	10207	9937	0

The queueing delay at every node has increased as compared to Part 1 because instead of two, now there are 4 nodes contending for the channel access.

Queueing delay: (in μs)

Node A	**Node B**	**Node C**	**Node D**
5.996314e+06	6.188456e+06	6.117557e+06	6.072341e+06

Average Access Delay: (in μs)

Node A	**Node B**	**Node C**	**Node D**
2.231953e+06	2.236604e+06	2.2250124e+06	2.166361e+06

Average Throughput:

From/To	**Node A**	**Node B**	**Node C**	**Node D**
Node A	0	1.346914e+07	1.321585e+07	1.337962e+07
Node B	1.288411e+07	0	1.292788e+07	1.297585e+07
Node C	1.291639e+07	1.320824e+07	0	1.301639e+07
Node D	1.272400e+07	1.291620e+07	1.309091e+07	0

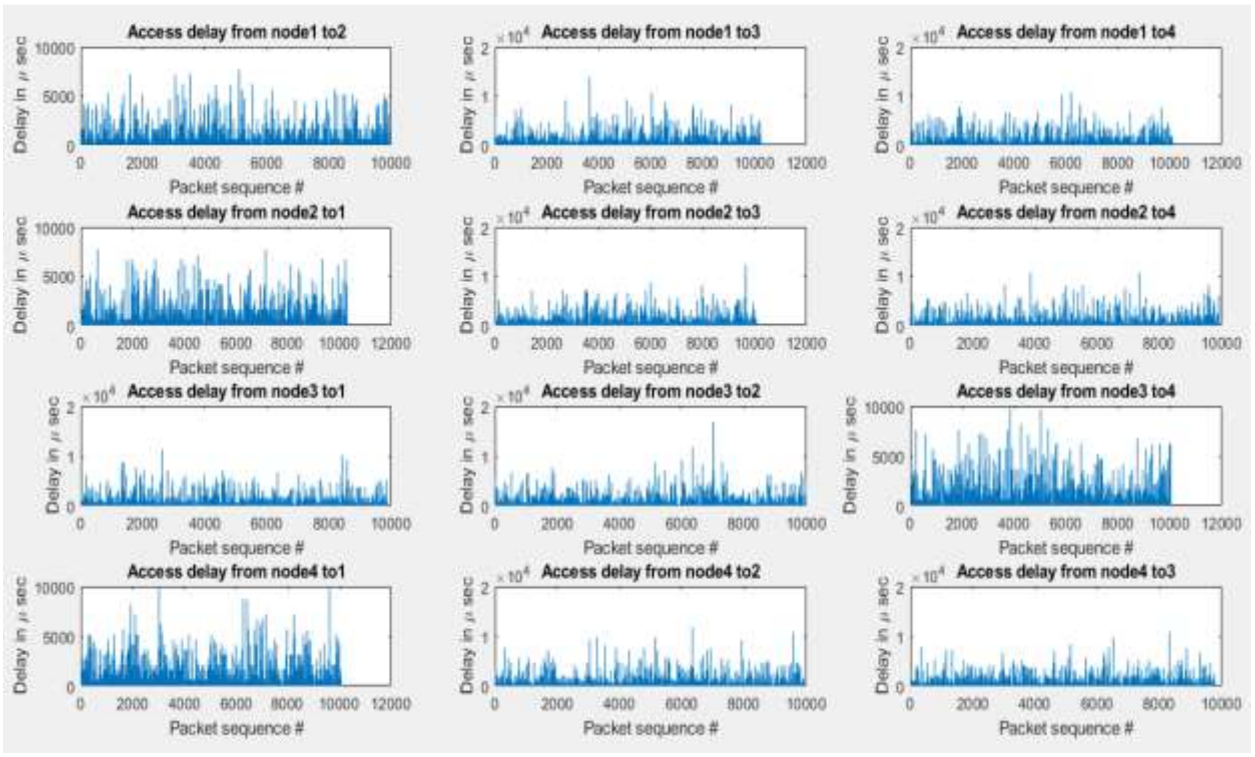


Fig. 3

From the Fig. 3 above we can see the difference between the access delays in packets sent from different source nodes to destination nodes. Here node 1 corresponds to node A, node 2 corresponds to node B and so on.

Part 3

In order to better understand the performance and relate to the network statistics, for the final implementation, we have included data which corresponds to simulation time of 3 sec. When we simulate for a period of 30 secs, the number of packets are too large for every source and destination pair of nodes and that makes us difficult to understand the graph better.

Source	Destination	No. of Packets Sent	Throughput
A	B	14	1.1197e+06
A	C	13	6.778e+05
A	D	18	1.0453e+06
B	A	10	2.9326e+05
B	C	14	3.4447e+05
B	D	14	3.0625e+05
C	A	18	3.5652e+05

C	B	13	4.7047e+05
C	D	16	3.5804e+05
D	A	8	1.6636e+05
D	B	13	2.4437e+05
D	C	10	2.391e+05

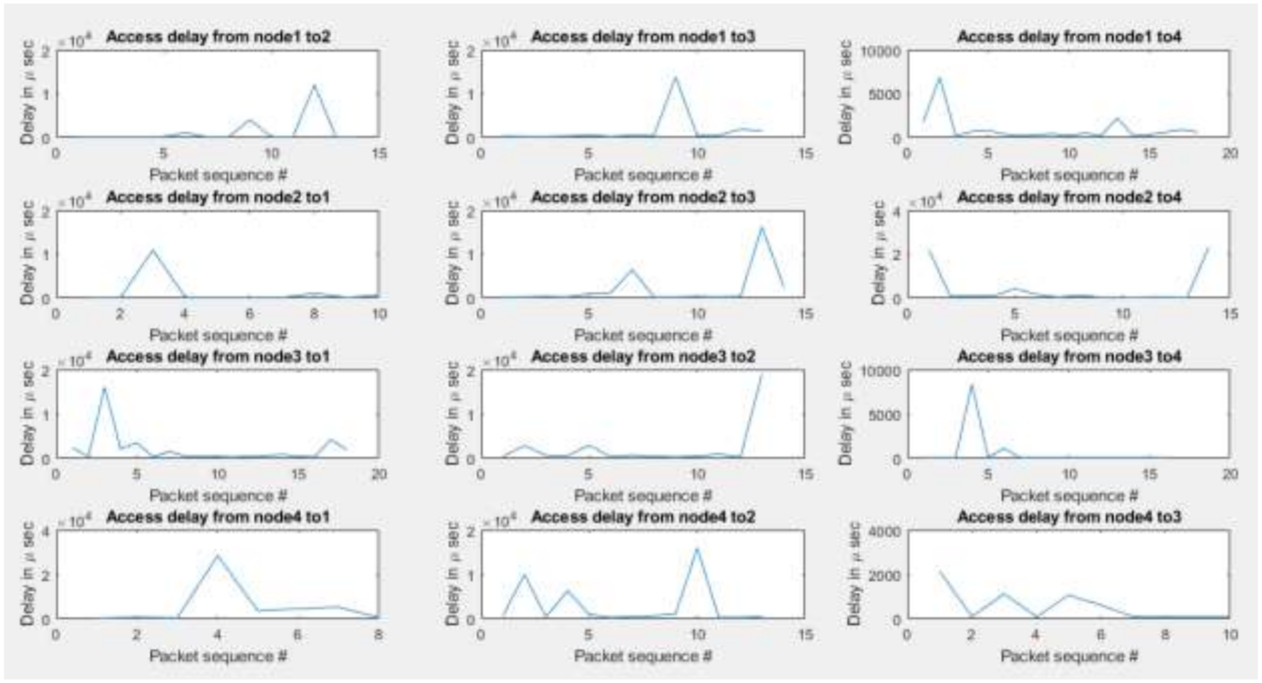


Fig. 4

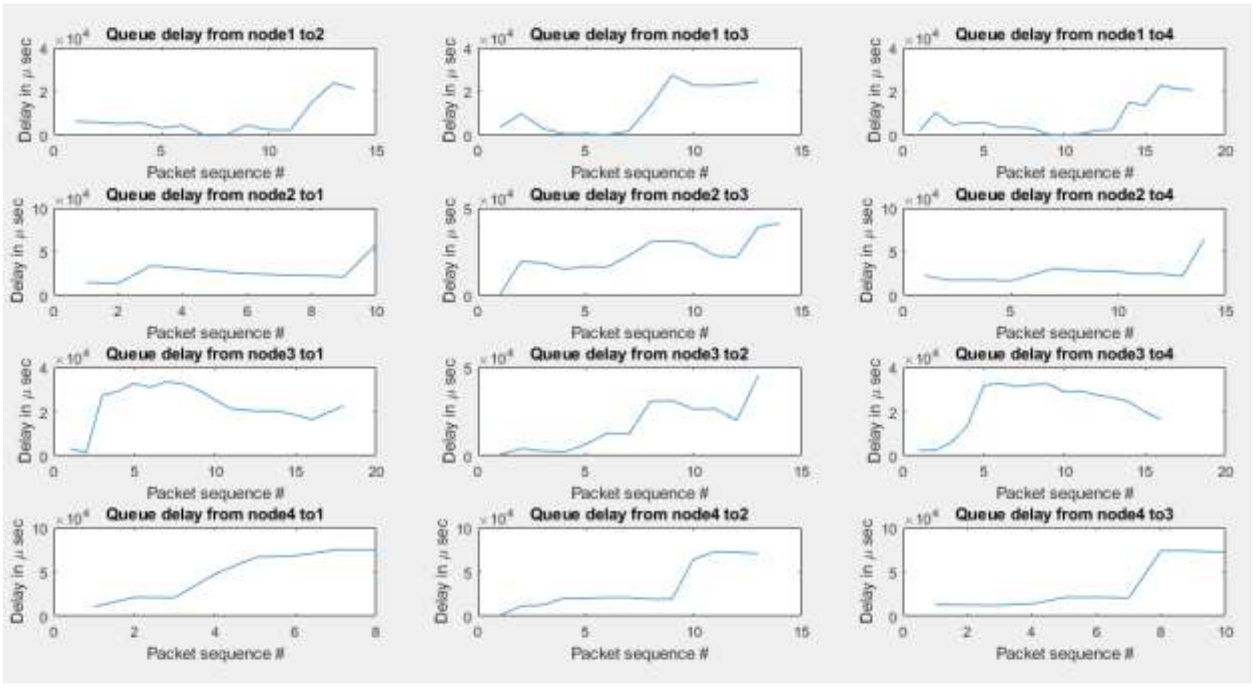


Fig. 5

The waiting time of the packets increases when there is increase in the number of collisions. Each collision results in retransmission of packets which in turn increases the delay of packet transmission.
In the Fig. 4 and Fig. 5 above we can see that there is a spike in the graph corresponding to the packet transmission from Node A (node1) to Node C (node 3) near packet sequence number 10. That indicates that packet 10 from A to C suffered more collisions as compared to the other packets that were transmitted for same pair of nodes.

OBSERVATIONS

In this subsection we will see how some network statistics like no. of collisions, inter-arrival time, and throughput, directly or indirectly depend on the value of frame slot time. And discuss some issue which have effect on the performance of the network.

We know that in CSMA/CD the node which is transmitting acquires the channel a time equal to that of frame slot time and starts transmitting the frames. If the slot time is larger than the total delay (transmission and propagation) we will not see any collisions. So, for us to observe collisions we have to take the time slot which is less than the total delay. So if the a collision occur in the shared bus, all the nodes which are sharing that bus will have to stop transmitting data, wait for a random back-off time which is calculated by the back-off algorithm and start retransmitting after that time. The retransmission is delayed by an amount of time obtained from the frame slot time and the number of attempts to retransmit and this algorithm has to be followed by the nodes which share the same Ethernet bus not by the nodes which are outside that LAN. For example let’s say there is a collision between packets with source = A and destination = B and source = A and destination = D, then A, B which are in the same LAN have to follow the back-off algorithm not D because it is in different LAN.

The frames at each node are generated using the Poisson distribution. So the generation time between two successive frames which is the inter-arrival time is larger initially and eventually decreases following exponential distribution. As a result, the subsequent frames are generated with less inter-arrival time. We can observe this criteria in fig.1. So the generation of the number of packets and the inter-arrival time depends on the frame slot we used in the Poisson distribution. But irrespective of the length of frame slot the graph between no. of frames and frame interval follows exponential distribution.

Throughput by definition is the number of successful transmissions per time of transmission. That is ratio of no. of bits sent and the time taken (idle +busy). So, the throughput is more when the frame slot time is small (the idle time will be small as the node's access time over the medium will be small). So the value of the throughput also depends on the frame slot time.

INSIGHTS

Impact of Lambda value:
The inter-arrival time depends on the value of mean (λ). We are using λ=0.5 in our implementation that is for every two frame slots one packet is generated. If the mean value (λ) decreases, let's say to λ=0.1, then for every 10 frame slots only one packet is generated. So the number of packets generated will decrease and the channel idle time will be more and as a result the throughput value decreases. We can also try keeping different frame slots for different nodes, this mean value has to be selected without affecting the throughput significantly.

Impact of Packet Size:
Another factor that significantly affects the throughput value is packet size. Throughput is defined as the number of successfully transmitted bits sent upon total end-to-end delay. If the packet size increases the numerator in above equation increase which increases the overall throughput. But while using an Ethernet cable we have a limitation on the size of packet which is 1518 bytes. For obtaining high throughputs the packets size must be large and in Ethernet the ideal size is 1518 bytes.

Rate of Collision:
The exponential backoff algorithm used in CSMA/CD does not reduce the possibility of collisions but it minimizes the recovery time when a collision happens. Once the collision occurs and the nodes started following backoff algorithm, there will be a disadvantage to the node which gets to transmit later, because the probability of transmission of the second transmitting node will always be less than that of the first transmitting node. It happens because this is a single channel system. Only one node gets to transmit a packet while others have to wait for the channel to be idle.

Throughput Degradation:
The throughput of the system reduces due to the excessive collisions and back-off delays. It has been taken care of to some extent by reducing the maximum backoff limit to 3 which guarantees a tolerable delay. We can also set a buffer size at each of the nodes to ensure that the nodes transmit only when they have enough data to send. This will reduce the number of nodes contending for the channel at the same time which in turn will reduce the bandwidth loss and collision rate.

PROS AND CONS

As mentioned earlier, our simulation is based on a Single threaded system which is more systematic to implement and also quickens the speed of the simulation. The program can be easily scaled for a large number of nodes, since there is no cost associated with handling separate threads at every node. The program is centrally controlled which gives us the flexibility to control and pause the execution by suspending or resuming the thread.

However, it is more realistic to have a multi-threaded system which would allow different nodes to have one thread that deals with packet generation while the other takes care of the transmission of the packets in the queue. It might result in slow down of the simulation speed and also require more memory space corresponding to every thread.

One of the main limitations or drawbacks of using CSMA/CD is that it becomes inefficient for a larger network due to attenuation. As the distance between two nodes (connected in a LAN) increases, the delay in collision detection also increases.

CHALLENGES

We faced several challenges during the whole project. The code was developed in different modules, creation of packet, backoff algorithm in case of collision, Dijkstra’s algorithm for routing, calculating statistics. Putting all individual pieces together to make the whole simulation work was challenging.
One of the other challenges after completing the whole flow was debugging the code to find which edge case and scenario was missed during implementation. During the testing phase, verifying the output against expected values was the most difficult task.

FUTURE IMPROVEMENTS

The single-threaded approach used to implement the protocol can be improved by using a parallel processor for separate event processing. This multi-threaded approach will help in making the implementation more real-time.
Along with collision detection, we can also expand the scope by implementing collision avoidance which will work in case of wireless LAN as well.
Due to limitation of time and of course the complexity of the task at hand, we focussed on developing a running version of our program that gives us correct results.

CONCLUSION

Simulation of CSMA/CD for the given architecture was helpful in improving our understanding of the MAC layer protocols. On evaluating various performance parameters, we observed that the maximum utilization depends on the length of the frame and on the propagation time, the longer the frame or the shorter the propagation time, the higher the utilization.

With CSMA/CD implementation in the Ethernet, it suffers capture effect which makes it unsuitable for transmitting real multimedia traffic. Because of the random number of collisions and associated backoff time it does not guarantee any delay bound for the traffic and behaves poorly under real traffic conditions, where the knowledge of the end to end delay is important for various reasons. Another reason for this MAC protocol to not work efficiently in real time traffic is traffic load. In case of multimedia traffic the offered traffic load will be more as a result the number of packets are more. If the number of packets are more, the number of collisions are more. With the increased percentage of collisions the performance of the network will be very poor.

ACKNOWLEDGMENT

Our sincere thanks to Prof. Wenye Wang for her unwavering assistance throughout the project.

REFERENCES

https://en.wikipedia.org/wiki/Discrete_event_simulation#Clock
https://www.mathworks.com/discovery/queuing-theory.html
https://en.wikipedia.org/wiki/Dijkstra%27s_algorithm

TEAM MEMBERS

Arpita Shekhar
Sriram Guddati