# ECE 358: Computer Networks
**Fall 2014**
**Project 2: CSMA/CD**

**Date of Submission:** November 17, 2014

**Submitted by:**

| | | |
|---|---|---|
| 20414514: | Louisa Chong | l5chong@uwaterloo.ca |
| 20434209: | Simarpreet Singh | s244sing@uwaterloo.ca |

**Marks received:**

**Table of Contents**

## High Level Overview of the Simulator

The program we wrote accepts eight variables. The first variable is the `tick_duration`, which defines the time that a tick represents in real time. The second variable is the `total_ticks`, this variable represents the total number of ticks we want our program to run for. The variable `N` describes the total nodes/computer we are simulating with. `A` is used to represent the average packet arrival rate per second. `LAN_speed` is defined using `W` with a unit of bits per second. `L` is used to describe the length of the packet in bits. To determine the persistent scheme, we used the letter `P`. Finally the `variable_calc` is used to define what data we want to log.

Using these variables, the simulation will start from main. Main function will call the `tick_tock`, which defines how many discrete ticks that the function will run. The `tick_tock` functions runs a for loop for a defined variable `total_ticks` times. Each tick we run, we also loop through the logic for all nodes. During a tick, each node is responsible in determining its packet generation. If the node generates a packet, it will place the packet into a list which represents the buffer. If the node is generating a new packet, we then move on to the CSMA/CD logic.

In the CSMA/CD, we divided the logic based on the state each node can be in. The states are separated as following; idle, medium sensing, transmitting, sending jamming signal and in exponential backoff. When we don't have any packets in the buffer, the node is idle. During the idle state, the node does not have any packet in transit and does not have any packet to transmit. When the node is not idle, we then start the transmission logic. To start transmission, we will perform medium sensing. For 1 persistent and p-persistent, we reset the counter to 0 and restart the medium sensing logic immediately when medium was sensed to be busy. For non-persistent, we included a logic incorporate a random wait time. To calculate the wait time, we first obtain the node's more recent binary exponential wait time from a dictionary of node and binary exponential wait time mapping. With that value, we generate a uniformly generate random wait time. Using the wait time, we introduce a delay before we retry the medium sensing. If the medium is not busy, we then will increment the medium sending time for all persistent scheme. When the total medium sending time reaches 48 bit time, we then start transmitting the packet.

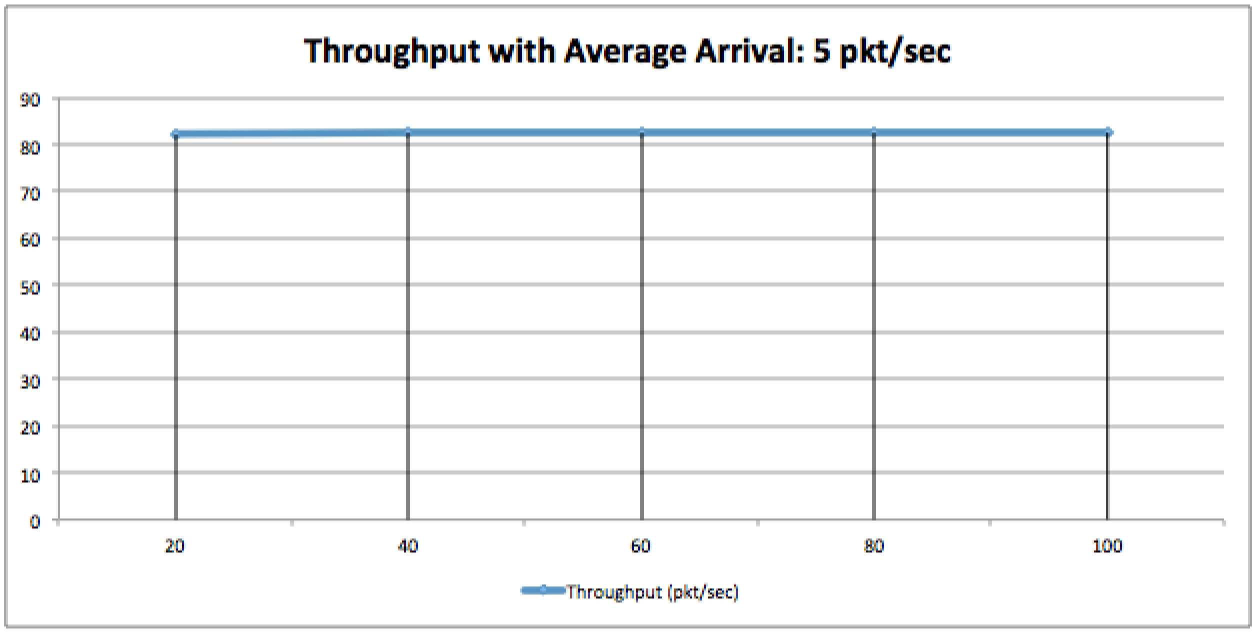
After the medium sensing state has completed, the server is then moved to transmission state. When we have a p-persistent system, we generate a random number from 0 to 1. If the randomly generated number is less than the p value we given, we then go through the binary exponential backoff state. Or else, we add the packet in transmission to a list which represents the medium. The packets in the medium lists represent any packet in transit. For anything other persistent system, one-persistent and non-persistent, we add the packet in the medium list directly. During the transmission state, we keep checking if the medium is busy. If we detected the busy state, we then determine if the signal comes from a jamming signal or a packet. If the signal was a packet, a collision occurred. We remove the item from the medium list and send a jamming signal. The server is then moved to the jamming state. If a jamming signal has occurred, we remove the item from the medium list and moved to the binary exponential backoff state. If the packet was fully transmitted without a sensing the medium is busy, we have a successful transmission. For a successful transmission, we reset  the values for binary exponential back off timer and the medium sensing time for the next transmission.

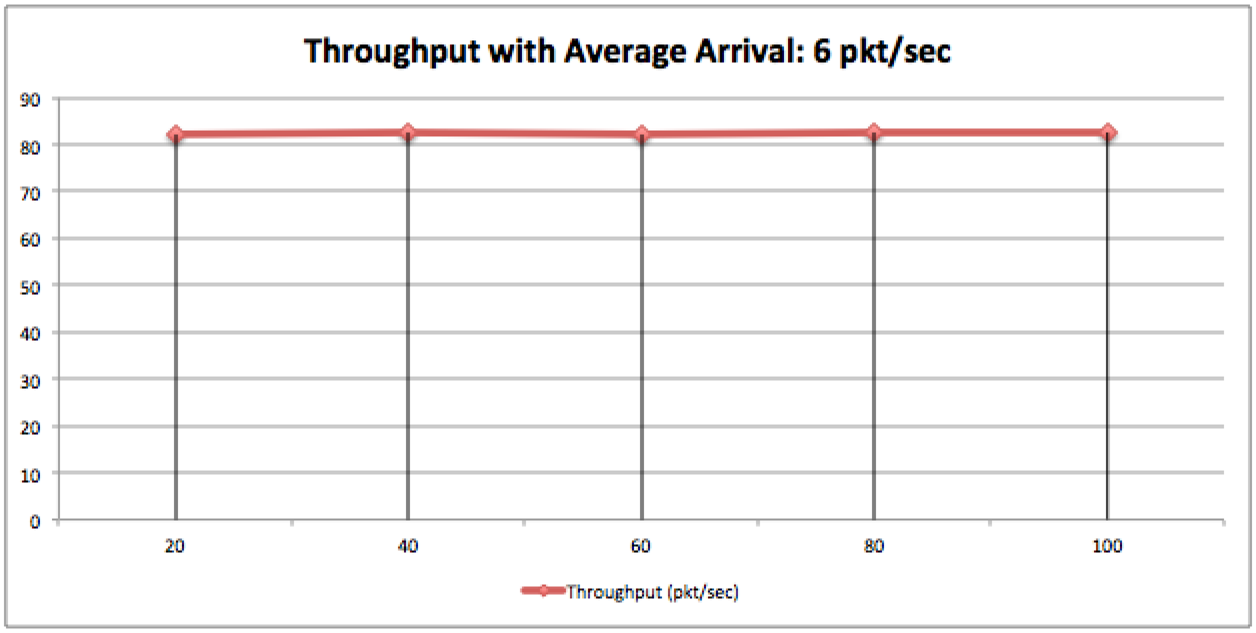
For the jamming state, the node will transmit a jamming signal by adding a jamming packet to the medium list. During the transmission time of the jamming packet, we do not perform any extra collision detection logic. After the jamming state has completed, the server is then moved to the binary exponential backoff stage.

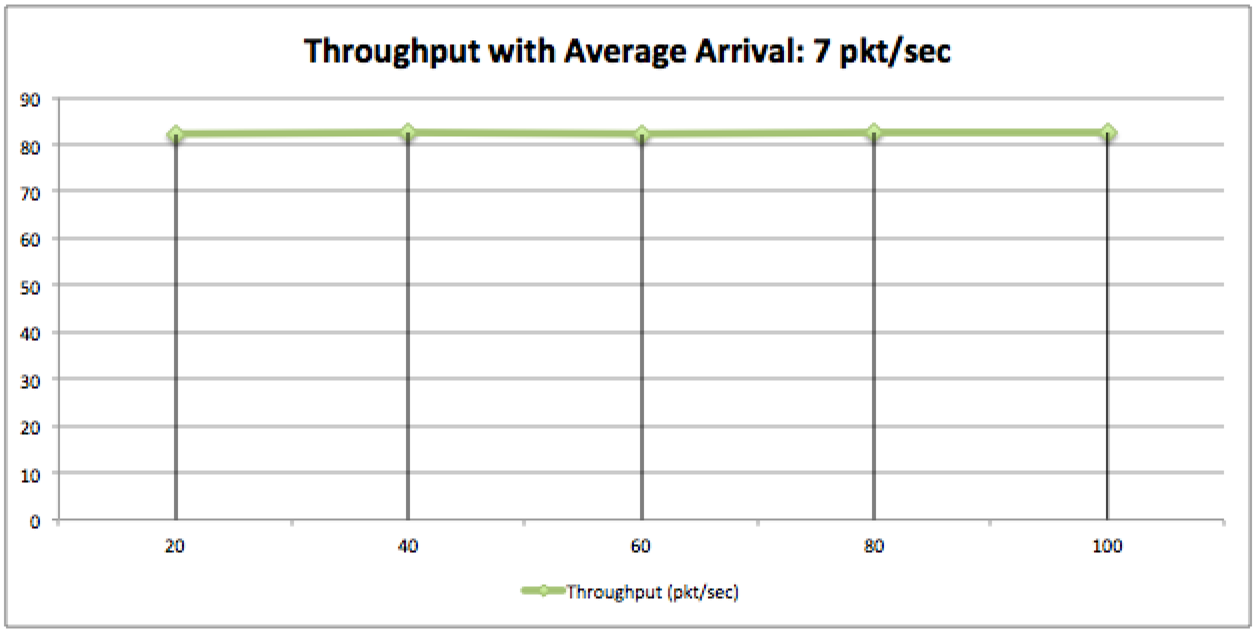
The exponential backoff state starts from looking up the exponential back off count dictionary. We then find out how many times did the packet go in to the exponential back off state. If the count is greater than 10, we increment the dropped packet count and reset the binary exponential count. If the counter is less than 10, we put the transmitting packet back to the computer's buffer and make the node wait for a randomly generated wait time based on the binary exponential backoff count and also increment the binary exponential backoff count. After the wait has completed we then resume the medium sensing state.

**Question 1: Throughput of the LAN as a function of N.**

From the plots below, we observed the throughput of the LAN is pretty consistent as a function of N. The throughput for different arrival rate looks approximately the same. As we increase the number of packets we generated in each N, the throughput remained constant. With 20 nodes and 5 packets per second, it requires a (5 packet * 1500 bytes * 8 bits/second * 20 nodes) = 1200000 bits/second = 1.2Mbps LAN to support the speed needed. With the increasing number of nodes and packets, the LAN speed required is even higher. The LAN speed we were given was only 1 Mbps, which is less than the total number for data we are trying to transmit with for all cases. Therefore, the throughput was maximized. Given a LAN of 1Mbps, the simulator can only transmitting 83 packets per second at most. This calculated value matches our collected data. We observed a near constant throughput of around 82 packets per second when we are generating 5 packets per second with 20, 40, 60, 80 and 100 nodes. We also observed a similar behaviour with the average generation rate of 6 packets per second and 7 packets per second.

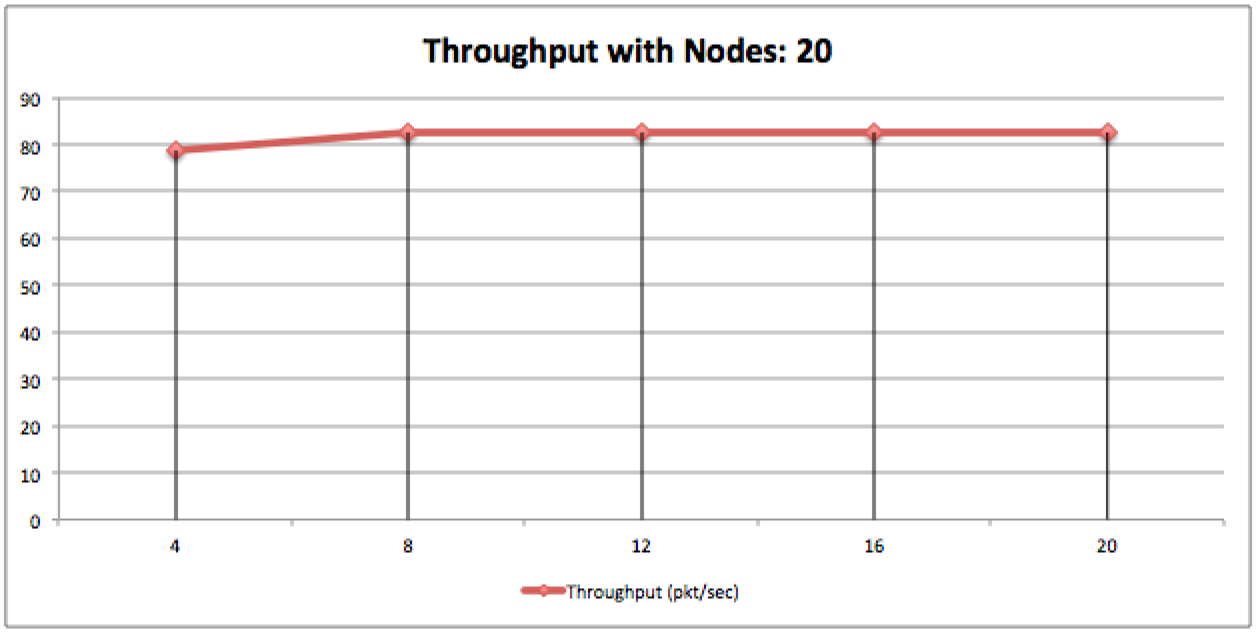


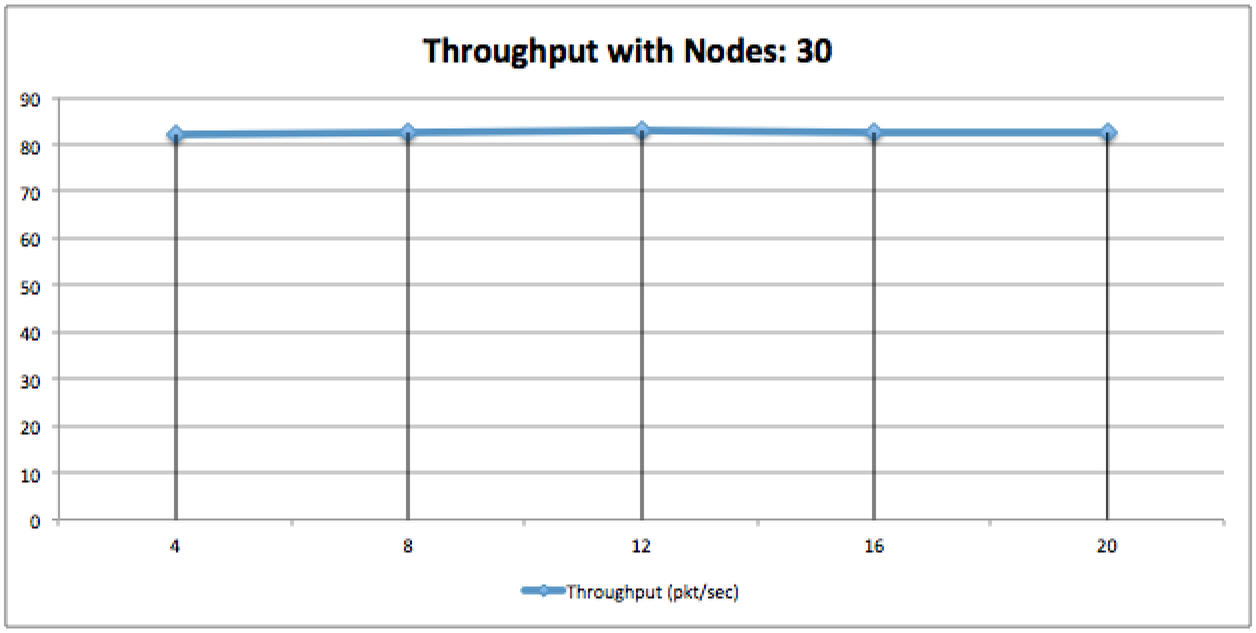

Throughput with Average Arrival: 6 pkt/sec
90
80
70
60
50
40
30
20
10
0
20
40
60
80
100
Throughput (pkt/sec)

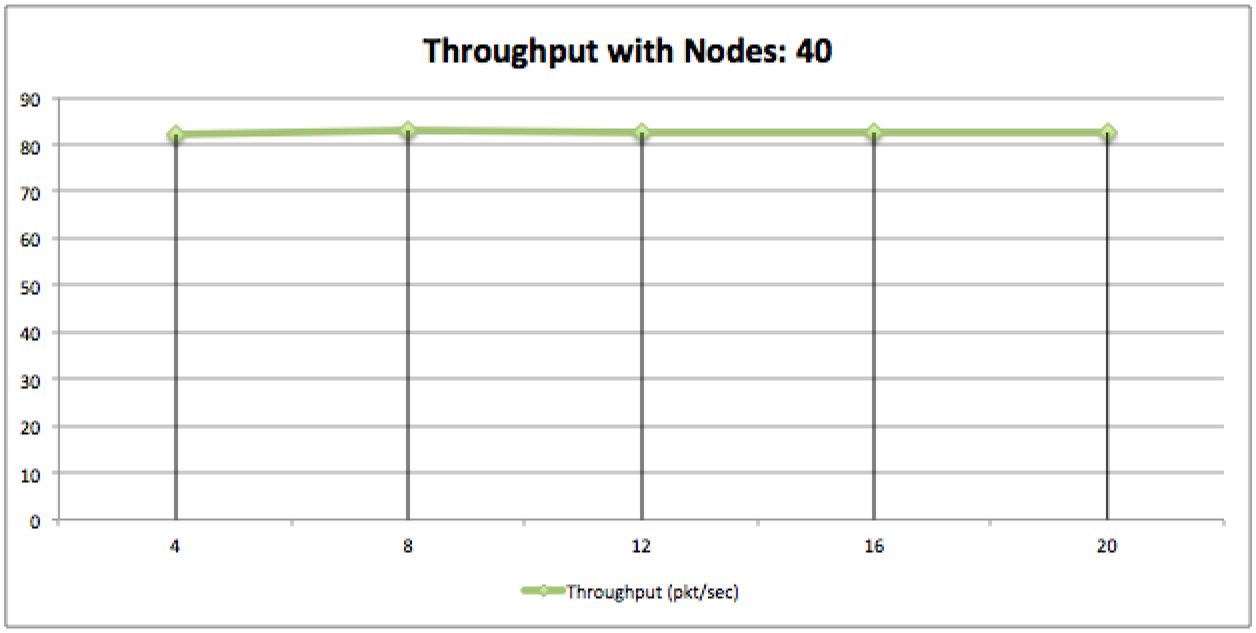

Throughput with Average Arrival: 7 pkt/sec
90
80
70
60
50
40
30
20
10
0
20
40
60
80
100
Throughput (pkt/sec)

**Question 2: Throughput of the LAN as a function of A**

Similar to question 1, the throughput data we collected was able to maximize the LAN throughput. Based on the number we calculated from the LAN we were given, it can only support a maximum of 83 packets per second. Therefore, we do not see an increase in the throughput as we increase the number of average packets generated from 4 packets to 20 packets per second. In the throughput as a function of node plots, the result is also as consistent as question 1. The values for all packet generated per second for different number of nodes were all around 82 packets per second. This observation is consistent in what we were expecting it to be.

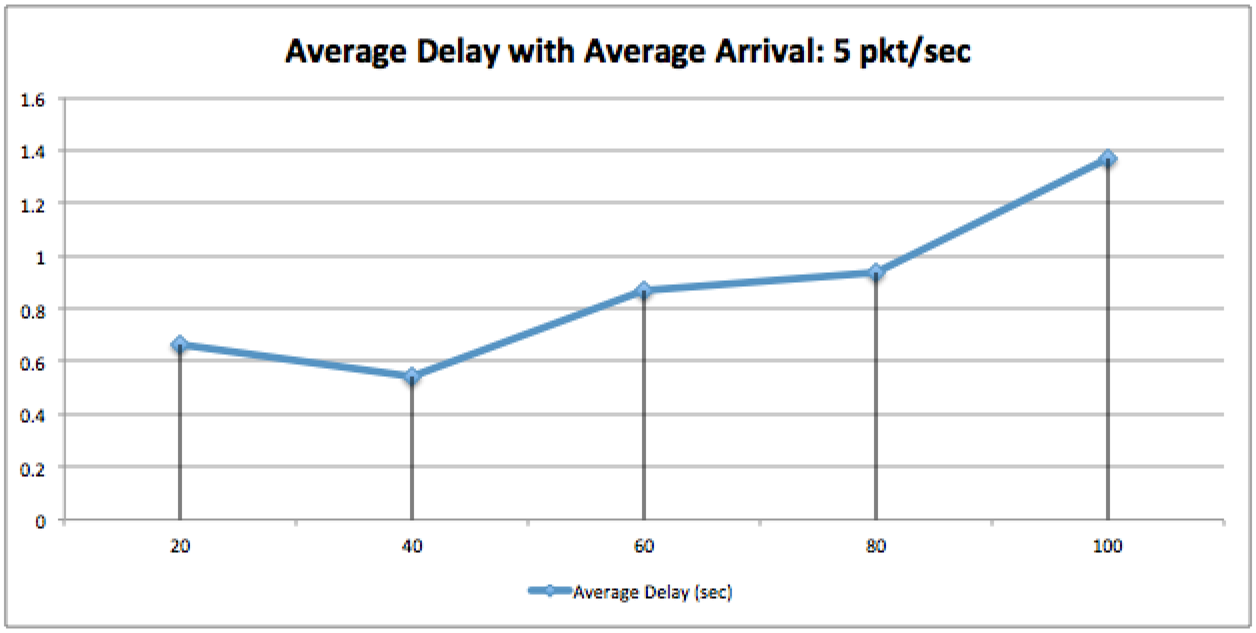


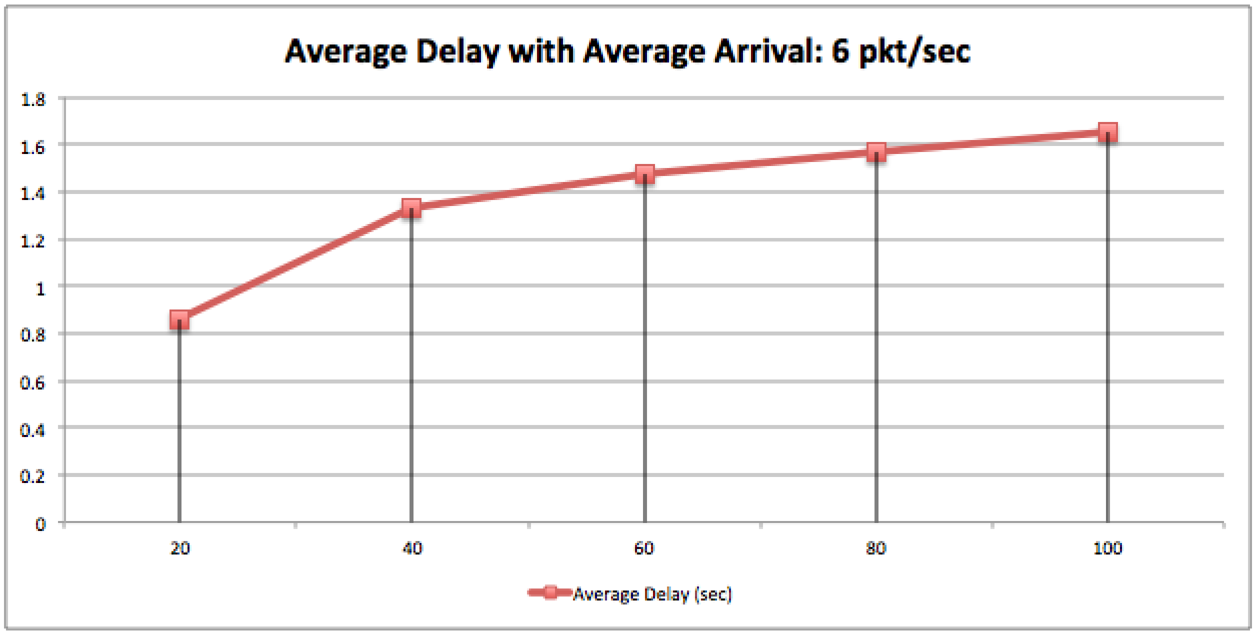

Throughput with Nodes: 30
90
80
70
60
50
40
30
20
10
0
4
8
12
16
20
Throughput (pkt/sec)

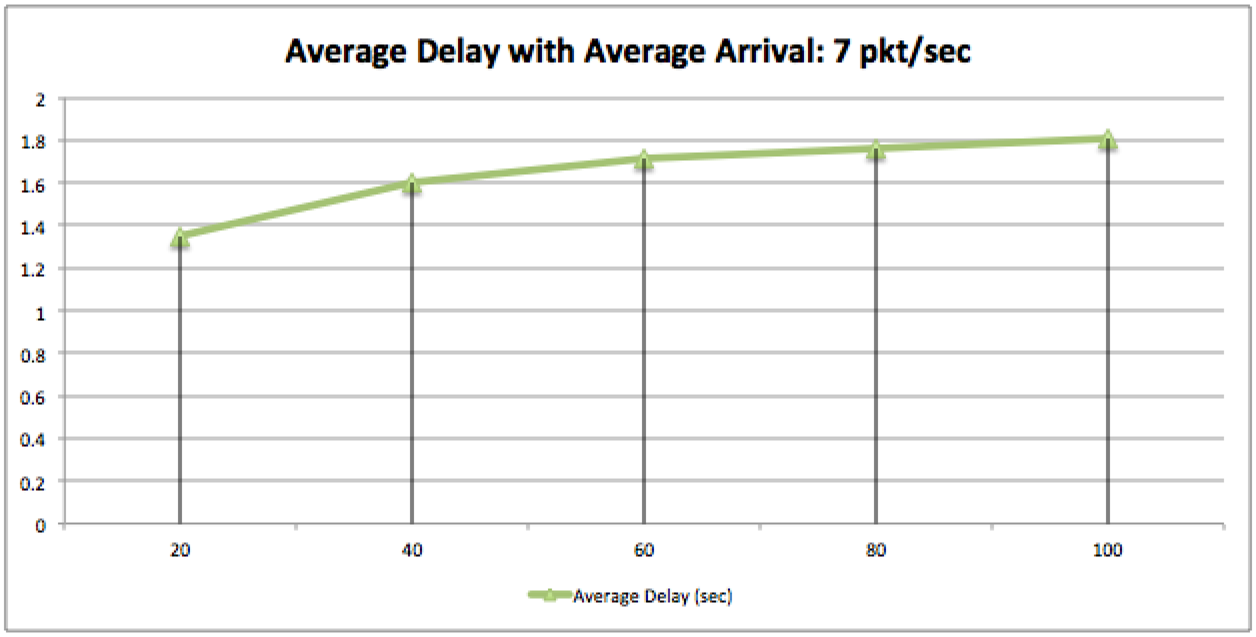

Throughput with Nodes: 40
90
80
70
60
50
40
30
20
10
0
4
8
12
16
20
Throughput (pkt/sec)

**Question 3: Average Delay of the LAN as a function of N.**

From the average delay graphs below, we observed an increasing average delay tread with the increase in the number of nodes. In the average delay plot for arrival rate of 5 packets/second, we observed a delay of 0.7 seconds. As we increase the number of nodes, the average delay increases with  the node count. This is because the total packets generated is a total of  100 packets per second at 20 nodes with 5 packets per second. The LAN only supports 83 packets per second. In the first second, we have 20 packets left in total for all nodes. As n seconds go by, there will be 20n packet left in total. In order to transmit the first 100 packets, we need at some additional time to process all packets. With the increasing nodes, the node buffer is getting more and more congested. Which leads to the increasing average delay. Additional to the congestion, each packet was also delayed due to the busy link. The busy link increases the number of collisions that result in the increased delay.

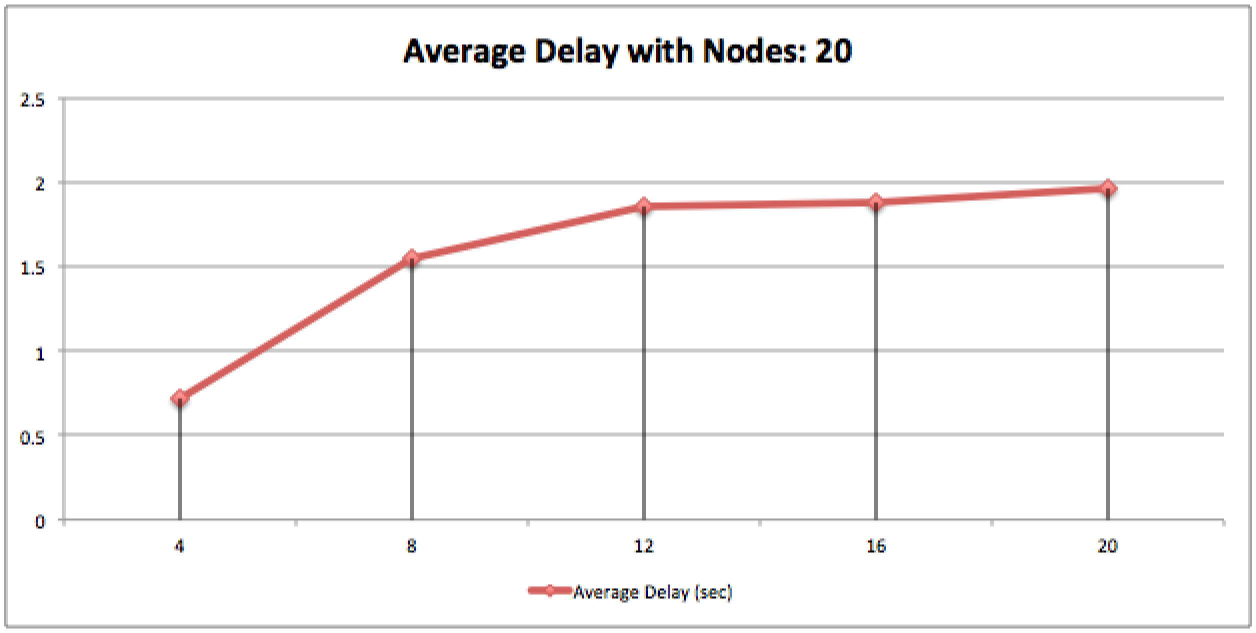


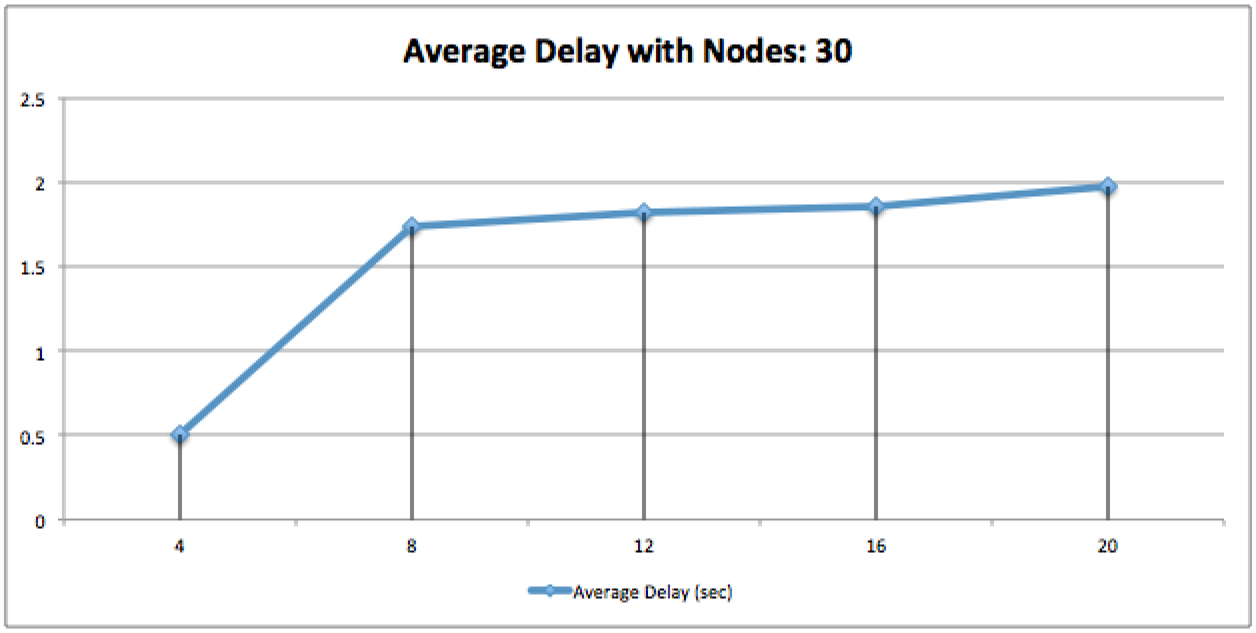

Average Delay with Average Arrival: 6 pkt/sec
1.8
1.6
1.4
1.2
1
0.8
0.6
0.4
0.2
0
20
40
60
80
100
Average Delay (sec)

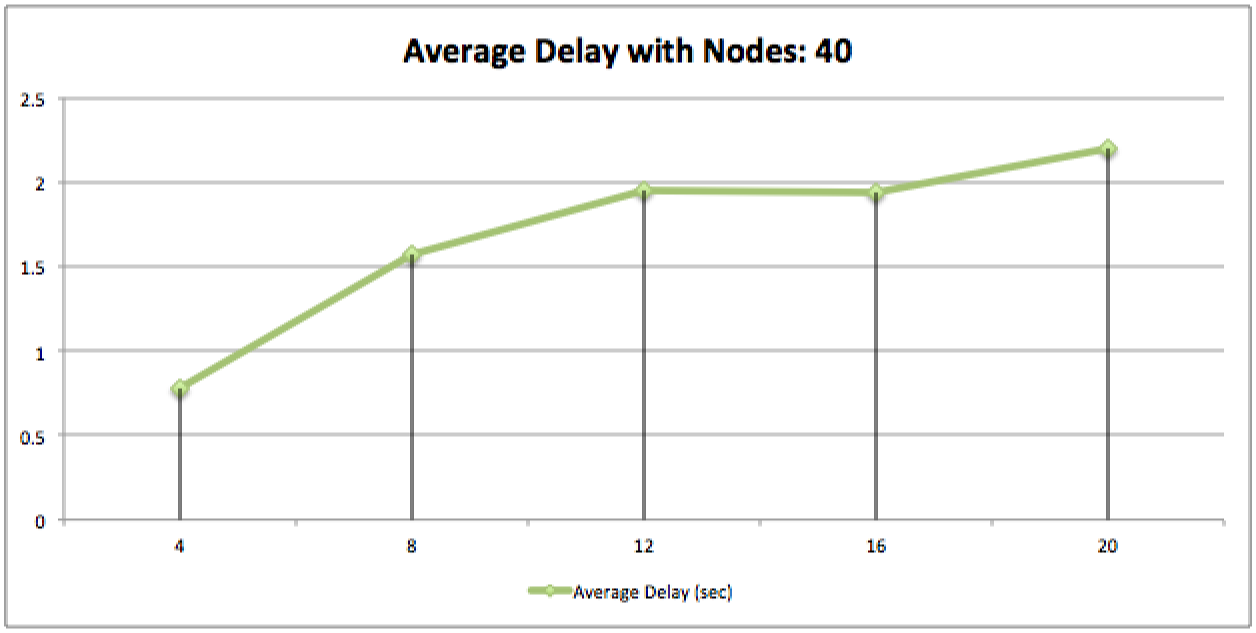

Average Delay with Average Arrival: 7 pkt/sec
2
1.8
1.6
1.4
1.2
1
0.8
0.6
0.4
0.2
0
20
40
60
80
100
Average Delay (sec)

**Question 4: Average Delay of the LAN as a function of A.**

From the average delay of LAN as a function of A, we also observed an increasing average delay trend with the increasing number of average arrival rate. When we have 20 nodes and we are generating at 4 packets per second, the average delay is around 0.7 seconds. As we increase the number of packets generated by each node, the total number of packets generated also increase. With the increasing number of packets generated the buffer for each link is filling up quicker. However, the link we were provided did not support the volume required and the whole system is all congested like the previous question. The congested system will then also lead to increasing number of collision.




Average Delay with Nodes: 30
2.5
2
1.5
1
0.5
0
4
8
12
16
20
Average Delay (sec)


Average Delay with Nodes: 40
2.5
2
1.5
1
0.5
0
4
8
12
16
20
Average Delay (sec)

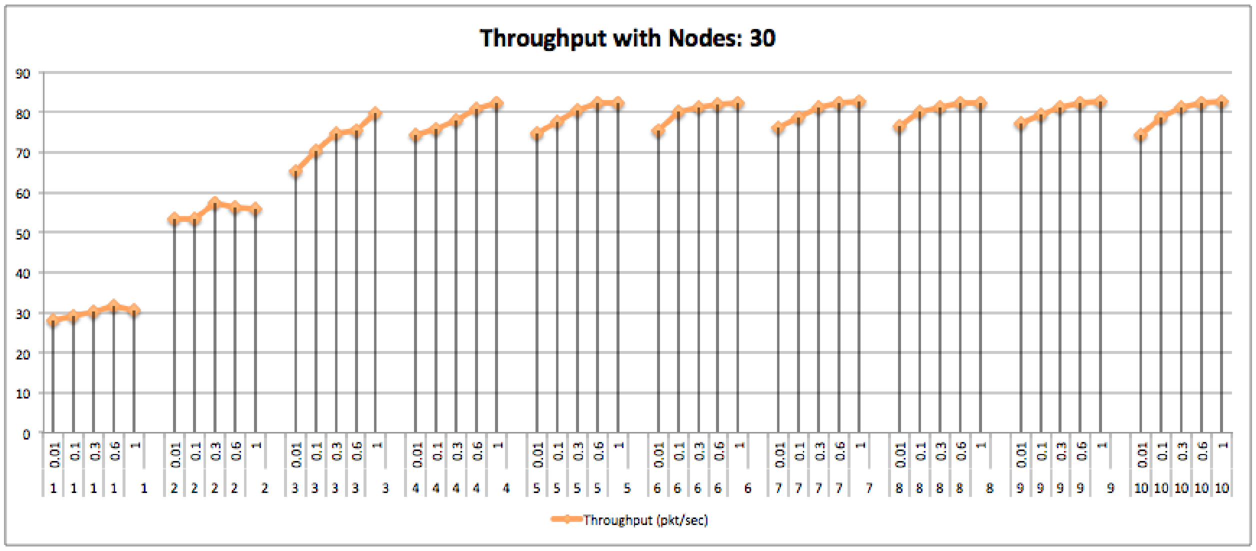
## Question 5

**Throughput and Delay of non-persistent and P-persistent CSMA protocols as a function of A.**

From the graphs below we can observe a few different things about the throughput and the average delay of both non-persistent and the persistent algorithms. In the case of the throughput graph we observe that the throughput tends to quickly saturate as the number of average packets per second generated is increased from 1 to 10 packets per second. We also observe that the increase in throughput is quite rapid around the jumps from 1 to 2 packet per second and also from 2 to 3 packet per second. The throughput, beyond the 5 packet per second mark tends to saturate fairly constantly. This is seen throughout the various P-persistent and non-persistent algorithms.

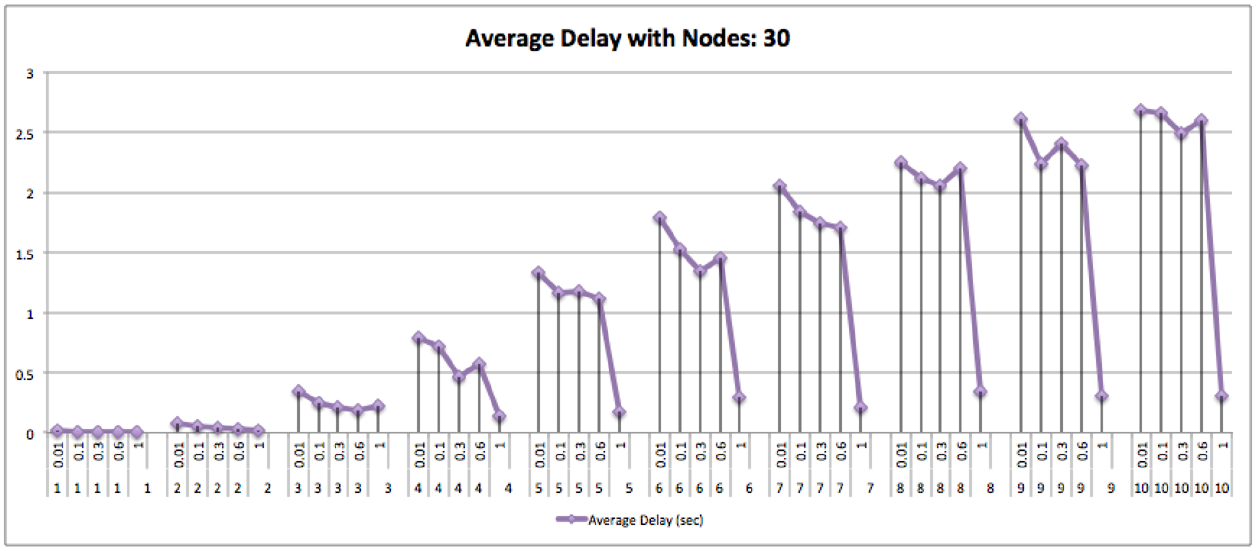
With 30 nodes and 5 packets per second, it requires a (5 packet * 1500 bytes * 8 bits/second * 20 nodes) = 1200000 bits/second = 1.2Mbps LAN to support the speed needed. With the increasing number of nodes and packets, the LAN speed required is even higher. The LAN speed we were given was only 1 Mbps, which is less than the total number for data we are trying to transmit with for all cases. Therefore, the throughput was maximized at around 5 packets per second.

This result also fits in perfectly with our previous observations made in earlier question as to why there's a saturation around the 5 packet per second mark. This graph clearly explains both phases.



The average delay curve is also within expectations. Here we can observe that as the persistence value is gradually increased from 0.01 to 1, the average delay incurred by a packet to reach its destination also tends to decrease. The descent in the average delay time per packet is fairly rapid as the transition is made from a 0.6 to a 1-persistent algorithm. The delay in most cases, falls by a factor of 10.

This fits in line with our understanding of how the various algorithms are designed and structured. In the case of all P-persistence algorithms the retry attempts before making another try are fairly spread out due to the probability outcome. As we increase the persistence value, this probability gets closer and closer to the value of 1, which being the 1-persistent case. The amount of packets deferred (not shown here) were also observed to go significantly lower as we approached the 1 persistence stage. This is again due to the fact that more packets get through, faster and in lower time, with a more aggressive algorithm like the 1-persistence algorithm.


Average Delay with Nodes: 30
Average Delay (sec)