Department of Electrical and Computer Engineering
ECE 358 – Computer Networks

Project 2: CSMA/CD Performance Evaluation

Group #45
Names: Shrey Kavi and Chris Huynh
User IDs: skavi (20613111) and cv2huynh (20616110)

Table of Contents

1.

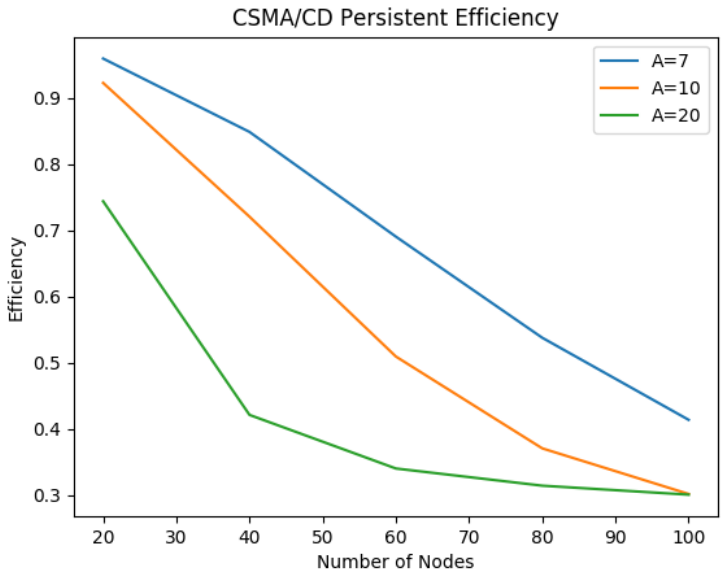

Figure 1: CSMA/CD Persistent Efficiency

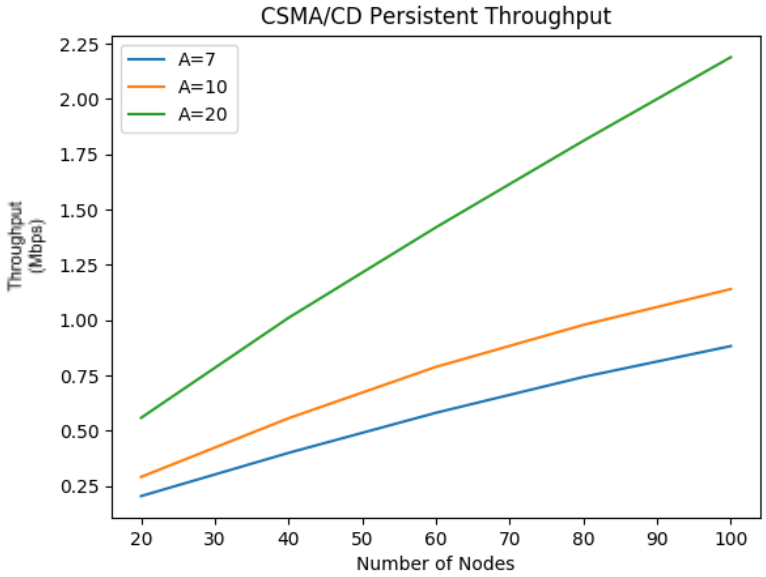

Figure 2: CSMA/CD Persistent Throughput

It can be observed in Figure 1 that the efficiency decreases as more nodes are added, this is because more nodes cause more collisions to occur. The increase in arrival rate (A) causes a decrease in efficiency, because it also catalysts collisions; more packets at a time so more can potentially collide. It can also be observed from Figure 2 that the throughput increases as number of nodes increase, this is because the more nodes have more packets to transmit. The throughput increases for an increasing arrival rate, this is explained because more packets are transmitting at a time.

2.

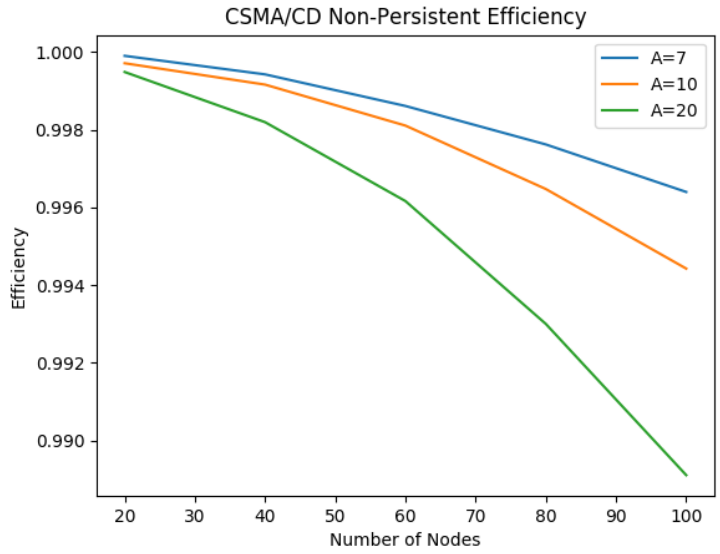

Figure 3: CSMA/CD Non-Persistent Efficiency

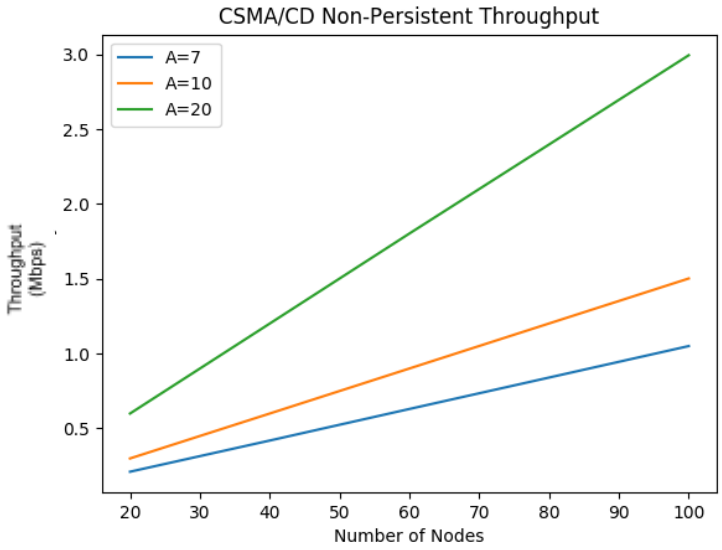

Figure 4: CSMA/CD Non-Persistent Throughput

In the non-persistent case instead of packets just waiting for the transmission to finish (not be busy anymore), it goes through an exponential back-off. The efficiency and throughput are generally better than the persistent case which is as expected because the exponential back-off in a non-persistent process helps avoid collisions of nodes by giving them more randomized wait times.

Looking at Figure 3, the efficiency drops as number of nodes increases, and it drops when arrival rate increases as well. This is because both of these variables increasing cause a increase of collision potential.
Looking at Figure 4, the throughput trends the same way as persistent; increase in number of node increases throughput and the throughput increases for an increasing arrival.

Source Code Design

In the zip file submitted is a python file called main.py. This file contains all the code needed to simulate a persistent and non-persistent CSMA/CD LAN. The code follows a discrete event paradigm similar to that discussed in lab 1. Therefore, the simulator program starts by building N queues for each node until the simulation time T. After the queues are then popped by minimum first, and the necessary calculations are done to see whether or not collisions occur, a bus busy is detected, or a packet is successfully transmitted. The conditions for these three events were defined in the lab manual, and were carried out in the program code. The provided code does not generate the graphs shown in figures 1 to 4, however the program does return the results of simulating the network with the same set of parameters for the graphs. There, if anyone wishes to graph the results of this program, they can use the returned data.