ECE 358: Computer Networks

Winter 2013

Project 2: CSMA/CD Simulation

March 13, 2013

Submitted by:

20334465	Shahen, Jonathan	jmshahen@uwaterloo.ca
20337152	Carlton, Kevin	kcarlton@uwaterloo.ca

Marks received:

Marked by:

Contents

1 Description of Simulation Design

The program consists of 4 packages, each is responsible for a specific functionality in the simulation. These are shown graphically in **Figure 1**

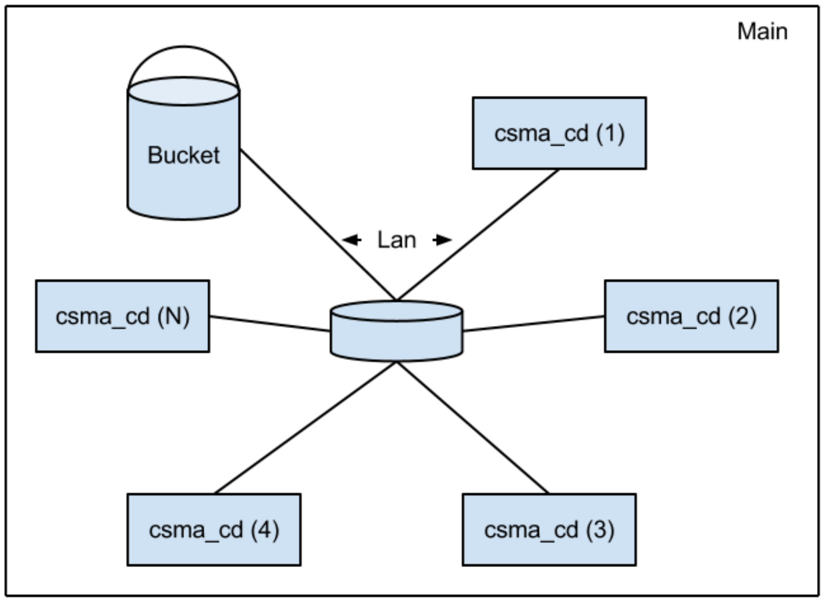

Figure 1: Layout of components

Main Responsible for reading in input variables, running the main loop of the simulation, and writing out results. It also holds variables to keep track of the averages needed over M runs of the simulation.

Lan Functions as the network between all computers. It allows computers (csma_cd) to put packets onto it. It keeps track of where the packets are on the Lan and where they are going. Additionally it allows computers to query if a packet is currently arriving at their node. Once a packet has reached it's destination successfully, Lan puts it in the bucket (Stats/bucket).

csma_cd This is a computer, or node on the network. It provides the functionality of the CSMA/CD Protocol. This includes sensing the medium (Lan), standard waits, transmitting frames, expeonential backoffs, and sending jamming signals.

Stats Provides structures to keep track of all the neccessary data. These include:

- Bucket: Accepts packets from the Lan once they have reached their destination. Holds all info and functionality relating to the delays (queue, csma and full) experienced by a successfully generated and recieved packet. This is per computer and on average for the whole. It also provides the functionality to determine the throughput of the system and throughput of each computer.
- Packets: Structure for a packet. Holds the Id of the computer it was generated by, and at what times it was generated, exited the queue, and arrived at its destination.
- Queue: Structure that implements an infinite queue that stores Packets. Each computer has its own instance of a Queue.

The program flow is as follows:

- Main reads in input variables from an input csv file. Upon reading all necessary variables, the outermost loop is entered. This is a loop to to run the simulation with N computers, from N_Start to N_End, in Steps of N_Step.

- Enter a loop to run the simulation M times. N computers are initialized, as well as the Lan and Bucket with the appropriate parameters.

- Enter a loop to run the simulation for T ticks. For each tick, the Tick() function is called on every computer, the the Complete_Tick() function is called on the lan.

- Each computer is a steady state machine that runs the CSMA/CD protocol. Upon Tick() being called, the Steady State Machine starts again in the current state. This steady state machine is shown in **Figure 2**

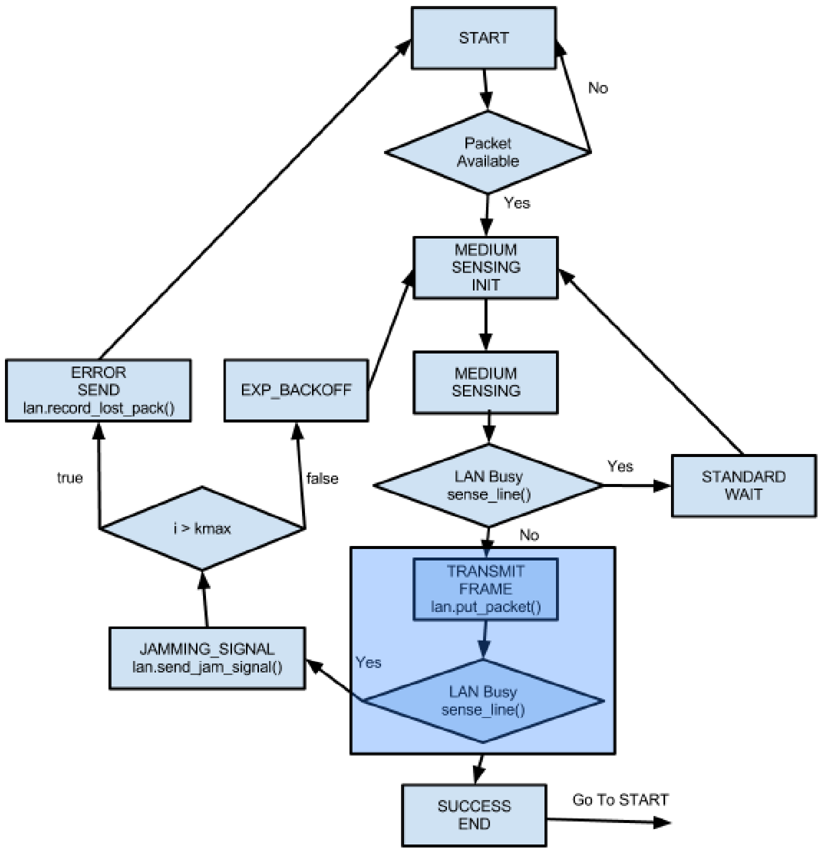

Figure 2: CSMA Steady State Machine

The lan keeps track of packets with the Packet_Arrival stuct. It holds the packet, as well as a start and end time. These are the times at which the packet will start and stop arriving at the other nodes (all nodes are the same distance apart- see assumptions). The lan keeps track of when packets are arriving at each node in a Slice (array) of Packet_Arrivals. It also holds an array of boolean values (sense_flags), one for each computer, showing if a packet is currently arriving at that node. Finally it holds two buckets, one for successfull packet arrivals, and one wo keep track of any unsuccessfully transmitted packets.

The Complete_Tick() function updates the sense_flags array by looking at the Packet_arrival arrival. If a packets end time is equal to the current tick, and no collisions were deteched, then the lan sends the packet to the successful bucket. If a computer has tried k_{max} times to send a packet,m the computer will disgard the packet to the lost bucket.

As seen in **Figure 2** The computer calls 4 functions of lan:

put_packet()	This puts a packet in a Packet_Arrival Struct with the appropriate start and end times.
sense_line()	Returns whether a packet is arriving at that computer at the current tick.
send_jam_signal()	Same as put_packet, except there is no packet as it is a jam signal.
record_lost_packet()	Tells the lan to send the packet to the lost_bucket.

Finally, The bucket has the function accept_packet() to accept a packet from the lan, as well as neccessary functions to compute statistics that deal with adding individual packets to the buckets (i.e. incrementing counters for throughput, adding to averages for delays, etc).

2 Assumptions

The following assumptions were made:

1. Network is in a star topology.
2. All computers are exactly 100 meters away from every other computer.
3. All packets are recieved without distortion; The medium is error free.
4. Sensing if the line is busy is an instantaneous operation (requires no time)
5. The packet buffer queue is of infinite size.

3 Performance Graphs and Discussion

Input Variables Used:

M	5
TICKS	10e6
TICK Time	10ns / Tick
A	300 packet/s
W	100 Mb/s
L	1500 bits

3.1 Throuhgput of Lan vs Number of Computers

Figure 3 is a plot of system's throughput vs number of computers. The graph is showing that the system is plateauing as the number of N increases. This means that the system is reaching it's maximum throughput.

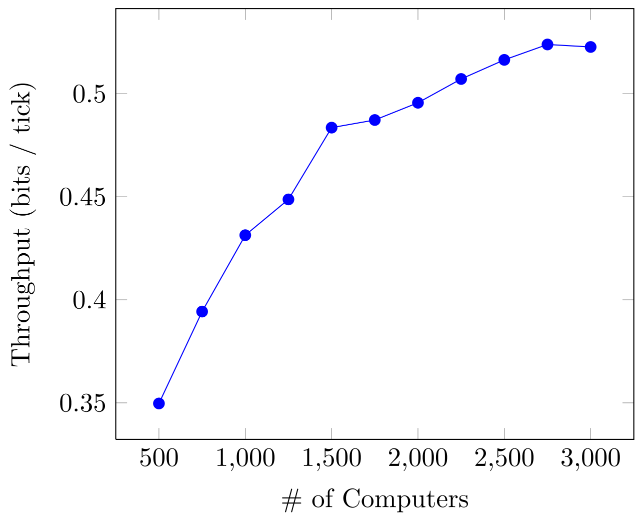

Figure 3: Plot of throughput vs. number of computer nodes

3.2 Average Delay over Lan vs Number of Computers

The system has three types of delays:

Figure 4 is a plot of queue delay vs number of computers. The Queue delay is the average number of ticks the packet was in the queue. The packet is taken out of the queue when the system is trying to send the packet.

Figure 5 is a plot of csma delay vs number of computers. The CSMA delay is the average number of ticks the packet spends in the CSMA protocol.

Figure 6 is a plot of the full delay vs number of computers. The full delay is the delay between when the packet was generated and when the packet reaches the Bucket.

We can see in Figure 4 that the Queue delay is neglible, and the packet does not spend long waiting to be picked up. From Figure 5 and Figure 6 we see a 4.5% drop in average delay ticks between 500 computers to 3000 computers, this change is also neglible. We expect that as the number of computers increases the average delay to also increase, as their is less bandwidth to be shared amoung all the computers, also as more computers are added the probability of collisions increases, thuis increasing the CSMA delay; because of a more probable increase in CSMA delay, this would also increase the Queue delay, as packets have to wait longer for the previous packet to be sent.

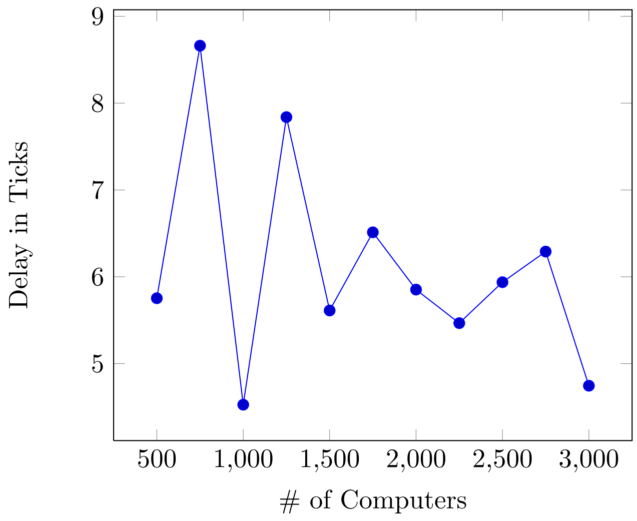

Figure 4: Queue delay vs. number of computer nodes

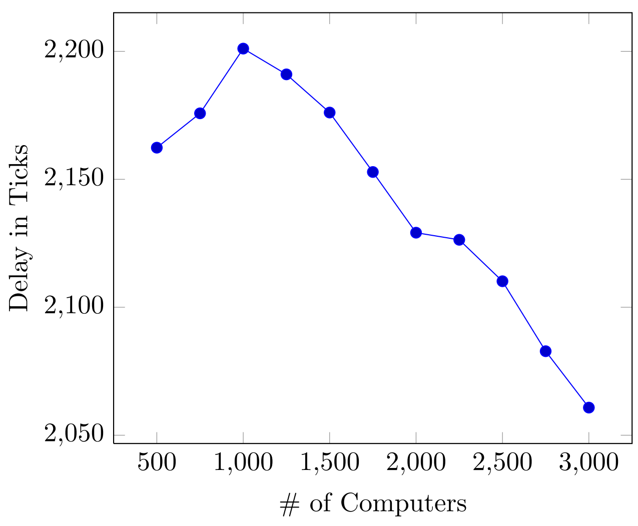

Figure 5: CSMA delay vs. number of computer nodes

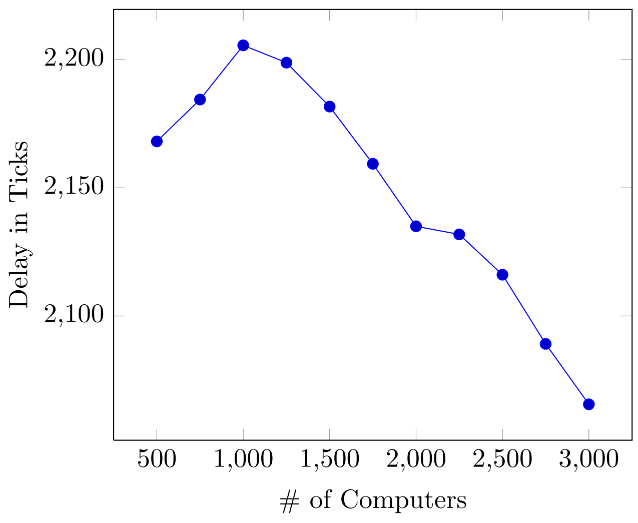

Figure 6: Full delay vs. number of computer nodes