NAME: RASHMI ANANTHAN

ID: A04165399

SIMULATION OF THE ETHERNET

INTRODUCTION:

The goal of this project is to simulate the classic Ethernet using socket API programming through client / server application. Stations are connected through the communication bus and the information is passed from one station to another through the bus. Collisions have been simulated and the wait time before transmitting has been implemented using the Binary Exponential Backoff algorithm.

ASSUMPTION:

Following are the assumptions on which this project is based:

1. Maximum number of stations allowed is 10.
2. The stations will start sending their frames only when they receive the ‘start’ message from the CBP.
3. The processing of the frames is strictly based on the input order and hence the space, alignment everything accounts for a successful transmission.
4. The stations will exit when the ‘End Of File’ is reached. Similarly, the CBP will automatically exit if the total number of stations is 0.
5. User will be asked to give the total number of stations required for the simulation.

DESCRIPTION:

Following is the detailed description of how this project has been implemented:

Language used: C

CBP side description:

1. The CBP is allowed to listen to a maximum of 10 stations.
2. I have used Select() function to accept and read from the stations.
3. An initial message is sent to all the stations involved in the communication set to telling them to initiate the process.
4. A structure having the ID, data received is stored for each and every station.
5. All the receiving data are stored in the corresponding stations.
6. Initially we check if more than one station is trying to send the data to CBP at the same time, if so we inform the stations about the collision.
7. If no collision is present, then we proceed to process the frames.
8. For each frame from the stations we do the following:
   1. The received details about the frame are stored in corresponding locations.
   2. We check if the CBP’s buffer is free if so we store part 1 of the received frame in it.
   3. Here the collision condition is possible when another frame from different station arrives while the first part of the frame is still in the buffer. In this case the clients are informed about the collision.
   4. If the next part of the same frame from same sender if received, CBP will send the buffer to the respective sender, waits for some time and then sends the next part of the frame to the same sender. The waiting here is established using sleep function.
   5. Collisions are checked every time before transmitting to check if another sender is sending the data.
   6. Every time the collision is checked by using the non-blocking select function call to check if more than one station is trying to send the data.
   7. In this case collision message is sent to the stations.
   8. All these details are stored in the CBP’s log file.

Station Process side description:

1. The station process receives two command line parameters – the ip address and the path of the input file.
2. For each station socket is opened, signal handler for I/O is registered using the Signal() function. The system calls fcntl() and ioctl() is used to handle various operations for the station’s socket descriptor.
3. The station will start processing the frames once it receives the signal from the CBP.
4. Once it receives the signal, it reads line by line from the input file and sends it to CBP in two parts.
5. After the first part has been transferred successfully, occurrence of collision is checked.
6. Collisions are detected by the stations using the message received through signals from the CBP.
7. The signal functions handle whenever the CBP sends any message. This message inside the signal function is used to initiate the frame sequence or to inform the sender about the existence of collision.
8. In case of no collision, the station would send the next part of the frame.
9. Select() function is used to check for the presence of input from the CBP. The time value is set in the last parameter to wait for the input from the CBP.
10. If collision is present, waiting time is determined using BEBO algorithm.
11. The station will exit when it has encountered end of file or if any error has occurred while reading the input file.

RESULTS:

The sample input file I have used has been attached. Similar file has to be given as input.

I have been able to simulate the collision scenario. Also frames transmitted from the station to the CBP is successful. However I am unable to make the CBP send the successful frames to the corresponding destinations. Otherwise for the given input, the program can transmit the frames from the station to the CBP, check for the collision and wait if the collision occurs. I am able to execute the project in different computers.

However I have observed the following behavior:

When I execute the project with two stations alone, I am able to generate the results properly. But when third station is given as input, the third one finishes execution but the remaining two hangs for a longer time. I was unable to sort this problem.

Following are the files required:

1. cbp\_new.c
2. network.h
3. sp\_new.c
4. station1.txt
5. station2.txt
6. station3.txt

Following are the steps for compiling and executing the file:

1. compile the cbp.c and sp\_new.c files and store them in cbp and sp executables respectively.
2. Execute the CBP first by using ./cbp command
3. User will be asked to enter the number of stations required.
4. Then start the specified number of stations by using the following format :

<executable> <IP\_address\_of\_CBP> <Input\_file\_name>

1. Then the data transfer between the stations and CBP takes place.
2. Once they are completed we can view the results in the following log files:
   1. cbp\_log.txt – to view CBP’s log
   2. sp1.txt , sp2.txt, sp3.txt – for the respective stations’ log