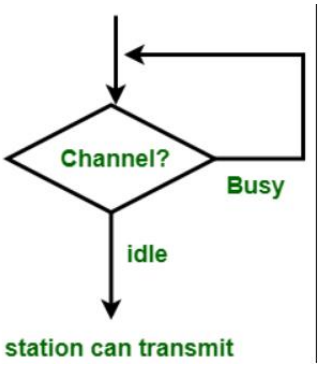
ASSIGNMENT 3

Name: Imon Raj
Class: BCSE III
Roll: 002010501098
Section: A3
Subject: Computer Networks Lab Report

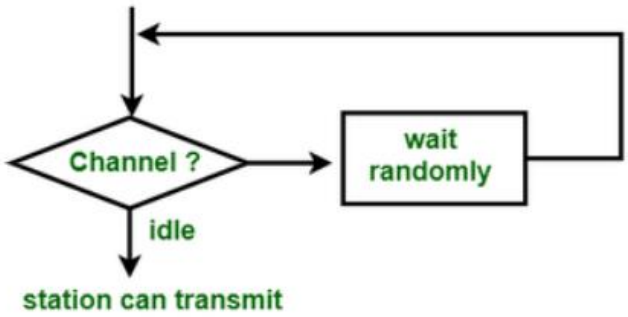
PROBLEM STATEMENT: In this assignment, you have to implement 1-persistent, non-persistent and p-persistent CSMA techniques. Measure the performance parameters like throughput (i.e., average amount of data bits successfully transmitted per unit time) and forwarding delay (i.e., average end-to-end delay, including the queuing delay and the transmission delay) experienced by the CSMA frames (IEEE 802.3). Plot the comparison graphs for throughput and forwarding delay by varying p. State your observations on the impact of performance of different CSMA techniques.

DESIGN: CSMA stands for Carrier Sense Multiple Access. It is one of the randomized MAC layer protocols. Like other MAC layer protocols, it is used in situations where multiple stations are using shared media. No station is superior to another. Any station can send frames whenever it wants, with some restrictions to be followed. Carrier sense indicates that a station must sense the medium before sending a frame. It can only send(using some persistence methods) a frame if the medium is idle. Carrier sensing reduces the chance of collision and improves throughput. There are three types of persistence methods for CSMA.

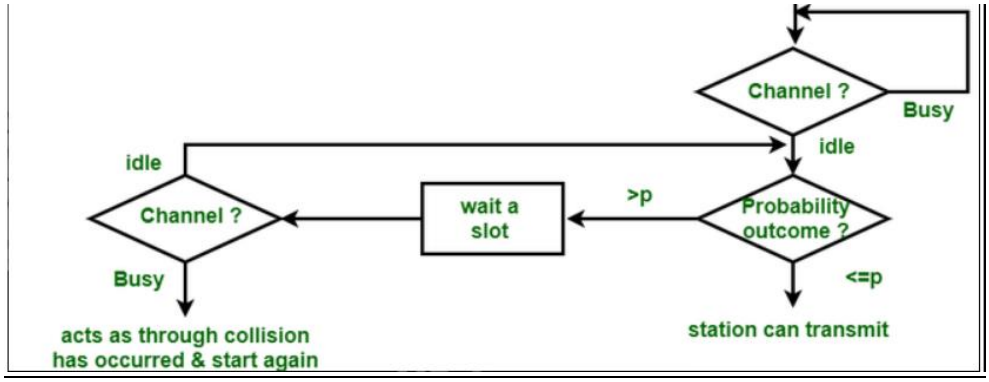
1. One-persistent: The station continuously senses the channel to check its state i.e. idle or busy so that it can transfer data. In case the channel is busy, the station will wait for the channel to become idle. When the channel is found idle, it transmits the frame to the channel without any delay. It transmits the frame with probability 1. Due to probability 1, it is called 1-persistent CSMA. The problem with this method is that there are a large number of chances for a collision to occur because two or more stations might find the channel in an idle state and the transmit frames at the same time. When a collision occurs the station has to wait for back-off time duration and then sense again.



2. Non-persistent: When there is a frame to send, a station senses the medium. If the channel is idle, station will immediately send frame. In case the channel is busy, it will wait for the random time and again senses the station whether idle or busy. In this method, the station does not continuously sense the channel for only the purpose of capturing it when it detects the end of the previous transmission. The main advantage of using this method is that it reduces the chances of collision. The problem with this is that it reduces the efficiency of the network. It may so happen that channel is completely idle but no station is sending, all are waiting, causes underutilization of the channel.



3. P-persistent: This is the method that is used when channel has time-slots and that time-slot duration is equal to or greater than the maximum propagation delay time. When the station is ready to send the frames, it will sense the channel. If the channel found to be busy, the channel will wait for the next slot. If the channel found to be idle, it transmits the frame with probability p, thus for the left probability i.e. q which is equal to 1-p the station will wait for the beginning of the next time slot. In case, when the next slot is also found idle it will transmit or wait again with the probabilities p and q. This process is repeated until either the frame gets transmitted or another station has started transmitting.



IMPLEMENTATION:

For the implementation of the three persistence methods of CSMA, I have used Java as a programming language. There will be two classes – Channel and Station. The Station class extends the Thread class so that multiple stations can run parallelly. Station class will have three methods for three persistence rules. The overridden run() method will execute one of these persistence methods. The Channel class will have a variable isBusy(to check whether channel is busy), a Semaphore mutex(to synchronize the access and modification of the isBusy).

All related codes written in Java are attached herewith (In the Drive-shared-folder).

TEST CASES:

The program should be able to assure that –

- No station should send when another is sending.
- The value of isBusy variable is consistent(synchronized).
- The persistence methods are obeyed properly by all stations.

RESULTS & ANALYSIS:

Program can fulfil all the test cases provided above. Currently, I have not found any bug, as the program is not very complicated. Snapshots of outputs of the three persistence methods are given below.

1-PERSISTENT:

```
Station 0 starts to sense medium..
Station 0 finds medium IDLE.. Medium Acquired..
Station 0 sends Frame..
Station 3 starts to sense medium..
Station 0 releases Medium..
==
Station 3 finds medium IDLE.. Medium Acquired..
Station 3 sends Frame..
Station 1 starts to sense medium..
Station 3 releases Medium..
==
Station 1 finds medium IDLE.. Medium Acquired..
Station 1 sends Frame..
Station 2 starts to sense medium..
Station 4 starts to sense medium..
Station 1 releases Medium..
==
Station 2 finds medium IDLE.. Medium Acquired..
Station 2 sends Frame..
Station 2 releases Medium..
==
Station 4 finds medium IDLE.. Medium Acquired..
Station 4 sends Frame..
Station 3 starts to sense medium..
Station 0 starts to sense medium..
Station 4 releases Medium..
```

NON-PERSISTENT:

```
Station 1 senses medium..
Station 1 finds medium IDLE.. Medium Acquired..
Station 1 sends Frame..
Station 2 senses medium..
Station 2 finds medium busy.. waits randomly..
Station 1 releases Medium..
==
Station 0 senses medium..
Station 0 finds medium IDLE.. Medium Acquired..
Station 0 sends Frame..
Station 4 senses medium..
Station 4 finds medium busy.. waits randomly..
Station 0 releases Medium..
==
Station 3 senses medium..
Station 3 finds medium IDLE.. Medium Acquired..
Station 3 sends Frame..
Station 2 senses medium..
Station 2 finds medium busy.. waits randomly..
Station 3 releases Medium..
==
Station 4 senses medium..
Station 4 finds medium IDLE.. Medium Acquired..
Station 4 sends Frame..
Station 4 releases Medium..
==
```

P-PERSISTENT:

```
Station 0 starts to sense medium..
Station 0 finds medium IDLE. generates -> (0.11) -> WILL SEND ACQUIRE MEDIUM..
Station 0 sends Frame..
Station 1 starts to sense medium..
Station 4 starts to sense medium..
Station 3 starts to sense medium..
Station 0 releases Medium..
==
Station 4 finds medium IDLE. generates -> (0.77) -> WILL NOT SEND..
Station 1 finds medium IDLE. generates -> (0.54) -> WILL NOT SEND..
Station 4 waits backoff..
Station 1 waits backoff..
Station 3 finds medium IDLE. generates -> (0.28) -> WILL NOT SEND..
Station 3 waits backoff..
Station 2 starts to sense medium..
Station 2 finds medium IDLE. generates -> (0.10) -> WILL SEND ACQUIRE MEDIUM..
Station 2 sends Frame..
Station 2 releases Medium..
==
Station 1 starts to sense medium..
Station 1 finds medium IDLE. generates -> (0.95) -> WILL NOT SEND..
Station 1 waits backoff..
Station 0 starts to sense medium..
Station 0 finds medium IDLE. generates -> (0.56) -> WILL NOT SEND..
Station 0 waits backoff..
Station 4 starts to sense medium..
Station 4 finds medium IDLE. generates -> (0.51) -> WILL NOT SEND..
Station 4 waits backoff..
Station 3 starts to sense medium..
```

I can see some improvement possibilities which are not present in my code. In my code, there is a global isBusy variable, but actually a station can only sense the medium in its point. Also in my code, no actual packet is being sent. My code is more about – station acquiring the medium, holding for sometimes and then release. In these mentioned topics, improvement can be done.

COMMENTS:

This assignment was not that lengthy. Like all the other assignments, it also encouraged me to think of an efficient solution. And coding this assignment was mostly fun and interesting.