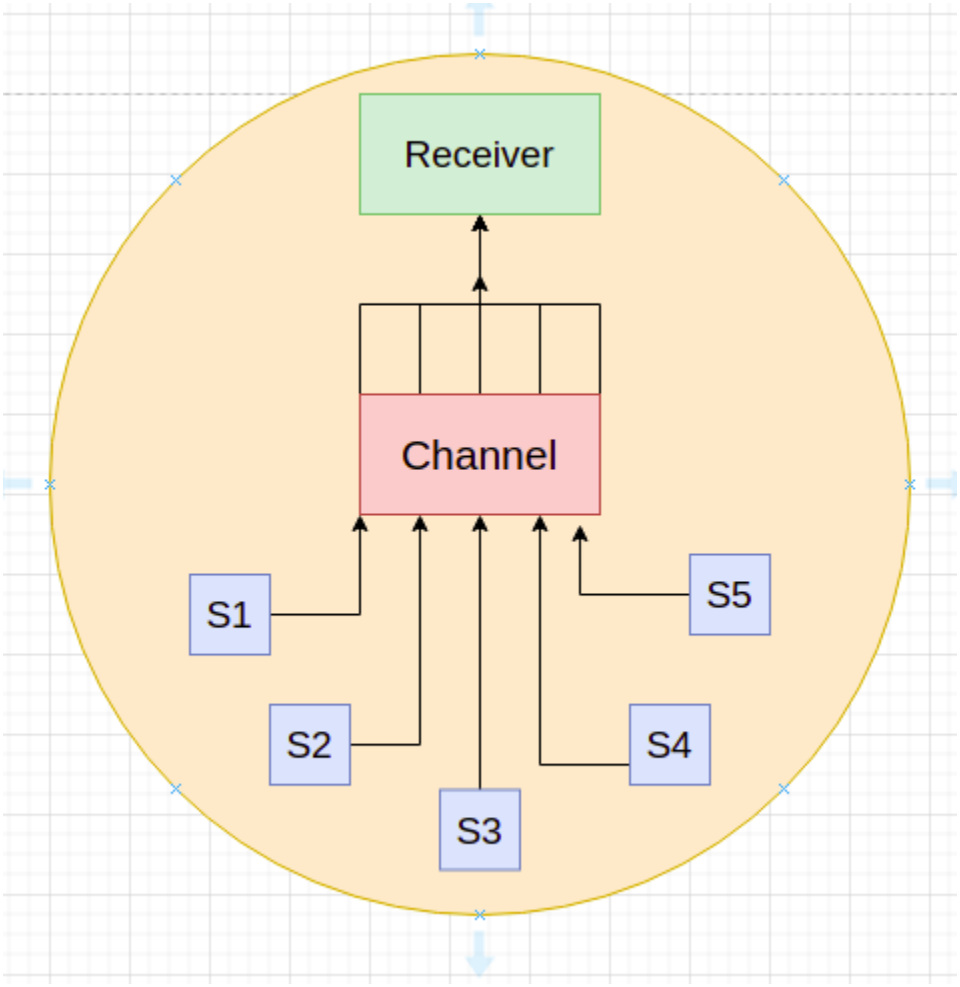
Network Assignment 3

Name :- Debargha Mukherjee Roll :- 001910501067

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.

Solution :

System Design

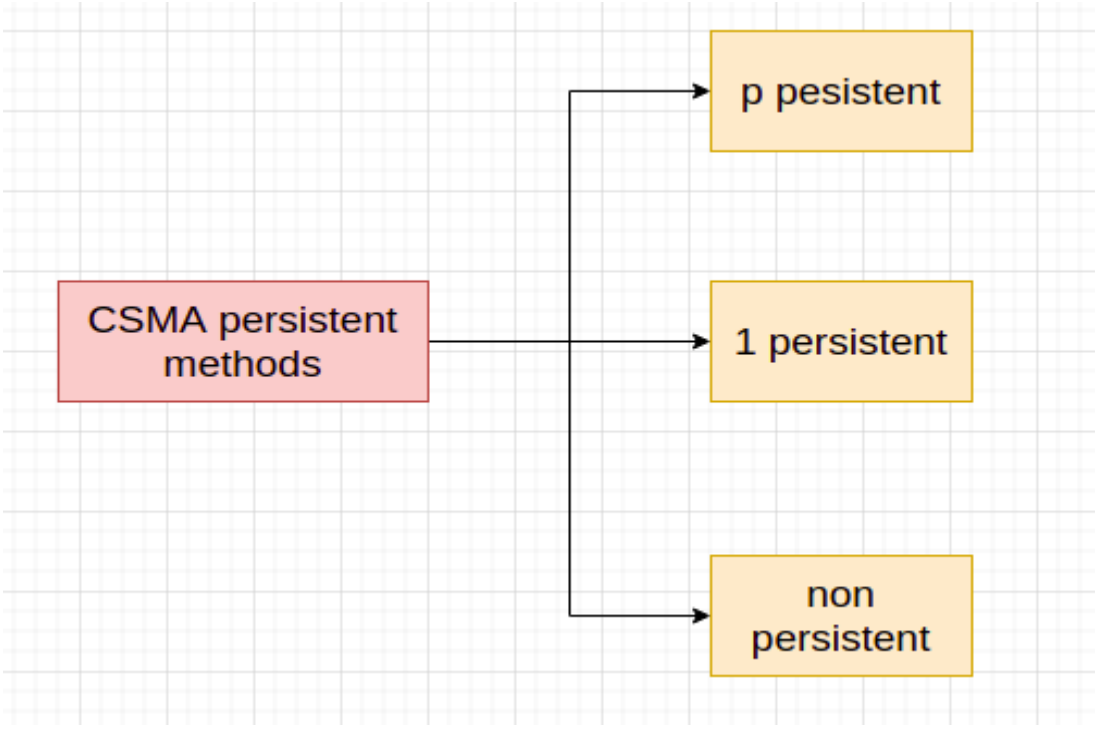


Simulated Network Environment designed using Simpy

Simpy :- SimPy is a **process-based discrete-event simulation** framework based on standard Python. Processes in SimPy are defined by **Python generator functions** and may, for example, be used to model active components like customers, vehicles or agents. SimPy also provides various types of shared resources to model limited capacity congestion points (**like servers, checkout counters and tunnels**).

Assignment Architecture (designed using Simpy) :- This entire assignment has been designed exploiting the features of simpy , to create a simulated network environment having a single receiver, a channel and option to create multiple stations which can act as senders to the receiver. There is no actual transfer of packet, but whenever a packet is required to be sent, the channel is kept busy by a particular station having an unique id.

CSMA persistent methods tested using this architecture



1-Persistent:

In 1-persistent CSMA, the station continuously senses the channel to check its state i.e. idle or busy so that it can transfer data or not. In case when the channel is busy, the station will wait for the channel to become idle. When a station finds an idle channel, 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.

Non-Persistent:

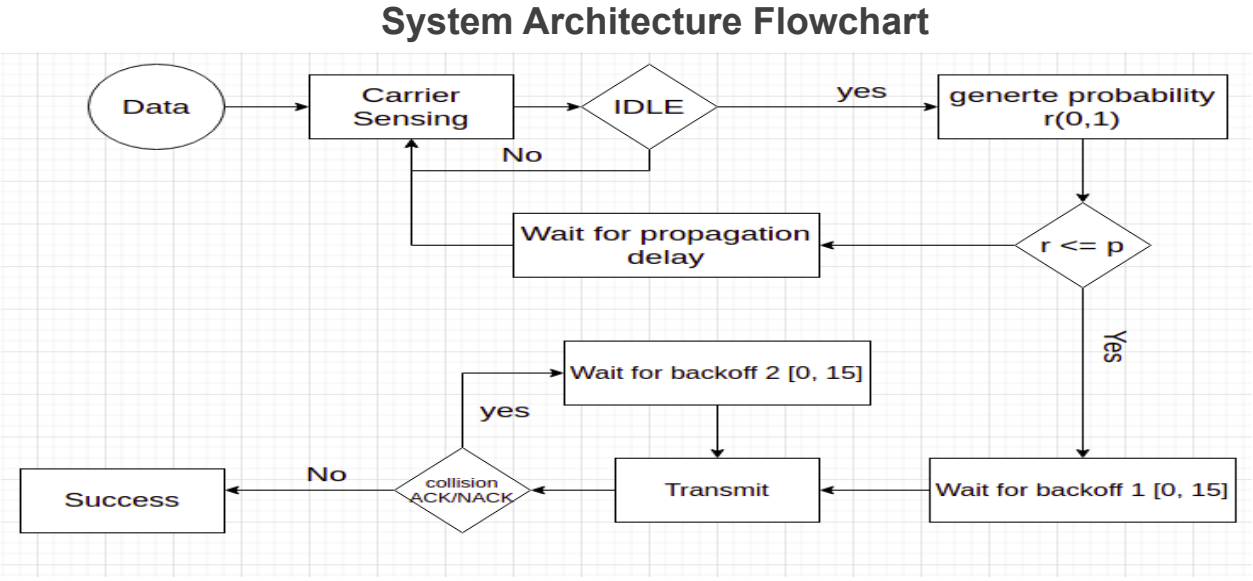
This is the method that is used when channel has time-slots and that time-slot duration is equal to or greater than the maximum propagation delay time. When the station is ready to send the frames, it will sense the channel. If the channel found to be busy, the channel will wait for the next slot. If 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.

P persistent

p-persistent CSMA is used when a channel has time-slots and that time-slot duration is equal to or greater than maximum propagation delay time for that channel. When station is ready to send frames, it will sense channel. If channel found to be busy, station will wait for next time-slot. But if channel is found to be idle, station transmits frame immediately with a probability p. The station thus waits for left probability i.e. q which is equal to 1-p, for beginning of next time-slot. If the next time-slot is also found idle, station transmits or waits
again with probabilities p and q. This process repeats until either frame gets transmitted or another station starts transmitting.

P-persistent CSMA is an approach of Carrier Sense Multiple Access (CSMA) protocol that combines the advantages of 1-persistent CSMA and non-persistent CSMA. Using CSMA protocols, more than one users or nodes send and receive data through a shared medium that may be a single cable or optical fiber connecting multiple nodes, or a portion of the wireless spectrum.

In p-persistent CSMA, when a transmitting station has a frame to send and it senses a busy channel, it waits for the end of the transmission, and then transmits with a probability p. Since, it sends with a probability p, the name p – persistent CSMA is given.



Components of the system architecture

channel class :- It is a discrete event used to establish a connection between sender and the receiver. It can have two states (True : Busy, False : Idle) , depending on which a sender can acquire it for transferring packets to the receiver. It is also responsible for maintaining the count of successful counts and collision counts. The receiver is present as part of the channel, since the main objective of this assignment is to study the effectiveness of CSMA protocols, we can avoid creating a real receiver process and use the channel instead. Instead of transferring real packets we can simulate packet transfer using the channel.

```
class Channel:
   # define message events
   succeedMsgEvt = 0
   failMsgEvt = 0
   noReplyMsgEvt = 0

   # collision count
   colCount = 0

   # successful read count
   readCount = 0

   # channel state
   channel = False  # True : Busy, False : Idle

   def __init__(self, env, tSlot):
       # initialize message event
       Channel.succeedMsgEvt = env.event()
       Channel.failMsgEvt = env.event()
       Channel.noReplyMsgEvt = env.event()

       # initialize collision count
       Channel.colCount = 0

       self.env = env
       self.tSlot = tSlot  # slot time

       # schedule process
       env.process(self.run())

   def run(self):
```

```
        global Count
        Count = 0
        while True:
            # one slot passed
            yield slotSignal.slotEvt

            # receiving the packets

            # check the collision
            tEpsilon = 0.1

            yield self.env.timeout(self.tSlot - tEpsilon)

            # send the feedback 0.1 time unit before the next slot
            if Channel.colCount == 0:
                Channel.noReplyMsgEvt.succeed(value='no_reply')
                Channel.noReplyMsgEvt = env.event()

            if Channel.colCount == 1:
                Count = Count + 1
                Channel.channel = True
                Channel.succeedMsgEvt.succeed(value='ACK')
                Channel.succeedMsgEvt = env.event()
                # print("\nACK : success at t = %4.1f\n" % self.env.now)

            elif Channel.colCount > 1:
                Channel.failMsgEvt.succeed(value='NACK')
                Channel.failMsgEvt = env.event()
                print("\nNACK : fail at t = %4.1f\n" % int(self.env.now))
                # reset collision count
                Count = Count + Channel.colCount
                Channel.colCount = 0
```

packet generator class :- It is responsible for generating packets and placing them in a queue for transfer, by inducing a certain waiting time between the launch of two packets.

mobile class :- It is acting as the sender over here having multiple methods defined within it in order to transmit the packets and for carrier sensing. The carrier sensing checks whether the channel is in a busy state or not and the transmit method is targeting to send the packet as soon as the channel becomes free.

```
def carrierSense(self):
        if Channel.channel:
            return True
        else:
            return False

    def transmit(self):
        global throughput
        global throughputA
        global throughputB
        global throughputC
        print("ID = %d, trasnmitting at t = %4.1f" % (self.mID,
int(self.env.now)))
        del self.Que[0]
        self.count += 1
        throughput += 1
        if self.mID == 0:
            throughputA += 1
        elif self.mID == 1:
            throughputB += 1
        elif self.mID == 2:
            throughputC += 1
```

Result And Analysis

Snapshot of p persistent running.

```
ID = 38, trasnmitting at t = 9472.0
ID = 38, trasnmitting at t = 9473.0
ID = 38, trasnmitting at t = 9474.0
ID = 38, trasnmitting at t = 9475.0
ID = 38, trasnmitting at t = 9476.0
ID = 38, trasnmitting at t = 9477.0
ID = 38, trasnmitting at t = 9478.0
ID = 38, trasnmitting at t = 9479.0

ACK : ID = 38, success at t = 9479.0

throughput : 0.039
offered load : 0.039
Collision Count :- 351
(base) debargha@debargha-Vostro-3480:~/BCSE_Network_Assignment/network_assignment3$
```

Snapshot of 1 persistent running.

```
ID = 10, trasnmitting at t = 9585.0
ID = 10, trasnmitting at t = 9586.0
ID = 10, trasnmitting at t = 9587.0
ID = 10, trasnmitting at t = 9588.0
ID = 10, trasnmitting at t = 9589.0
ID = 10, trasnmitting at t = 9590.0
ID = 10, trasnmitting at t = 9591.0
ID = 10, trasnmitting at t = 9592.0
ID = 10, trasnmitting at t = 9593.0
ID = 10, trasnmitting at t = 9594.0

ACK : ID = 10, success at t = 9594.0

throughput : 0.051
offered load : 0.051
Collision Count :- 461
(base) debargha@debargha-Vostro-3480:~/BCSE_Network_Assignment/network_assignment3$
```

Snapshot of non persistent running.

```
ID = 29, trasnmitting at t = 9917.0
ID = 29, trasnmitting at t = 9918.0
ID = 29, trasnmitting at t = 9919.0
ID = 29, trasnmitting at t = 9920.0
ID = 29, trasnmitting at t = 9921.0
ID = 29, trasnmitting at t = 9922.0
ID = 29, trasnmitting at t = 9923.0
ID = 29, trasnmitting at t = 9924.0

ACK : ID = 29, success at t = 9924.0

throughput : 0.048
offered load : 0.048
Collision Count :- 432
(base) debargha@debargha-Vostro-3480:~/BCSE_Network_Assignment/network_assignment3$
```

Analysis of CSMA persistent methods collision probability

From the above results, it is clearly visible that in 1 persistent CSMA the probability of collision is most and in p persistent it is least. Non persistent has an average behaviour which lies somewhere between one persistent and p persistent.

Justification

1. In one persistent the frame is sent immediately after the sender senses the channel idle, therefore it has the maximum chances of collision.
2. In non persistent, even if the sender sends the packet immediately as soon as the channel is detected idle, but it waits for a random time interval to send the packet as it detects the channel is busy.
3. P persistent method uses combination of above two methods. When it senses an idle channel, it doesn't transmit immediately. It generates a random value 'x' which must be less than p (= 1/n) to transmit the frame. If 'x' exceeds p, then it waits for a timeslot (Tp) then again senses the channel and repeats the above process. It is unlikely for different senders to generate 'x' (<p) in the same slot which reduces collision probability.

Collision :- 1 persistent > non persistent > p persistent

Analysis of CSMA persistent methods throughput

1. The channel utilization is best for p persistent and then non persistent and finally comes 1 persistent. Therefore the throughput is best for the persistent CSMA protocol.

2. As in one persistent method, as collisions increases with increasing number of sending stations throughput decreases drastically.

Tabular Data of throughputs, collision, and offered load

P persistent CSMA protocol

Number Of stations	Throughput	collision	Offered load	Simulation times
50	0.044	396	0.044	10000
100	0.093	837	0.093	10000
200	0.187	1687	0.187	10000
300	0.2756	2507	0.276	10000
400	0.3672	3338	0.3682	10000

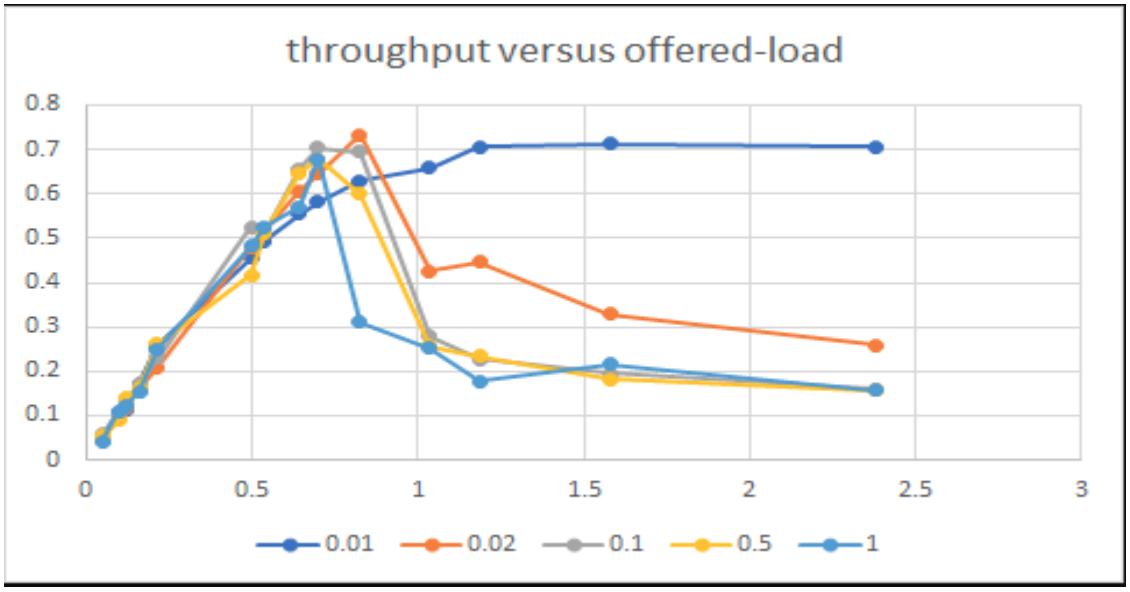
1 persistent CSMA protocol

Number Of stations	Throughput	collision	Offered load	Simulation times
50	0.039	351	0.039	10000
100	0.106	954	0.106	10000
200	0.170	2057	0.174	10000
300	0.234	3078	0.230	10000
400	0.3529	3558	0.3582	10000

Non persistent CSMA protocol

Number Of stations	Throughput	collision	Offered load	Simulation times
50	0.039	380	0.039	10000
100	0.101	902	0.101	10000
200	0.181	1887	0.181	10000
300	0.244	3024	0.243	10000
400	0.353	3488	0.3542	10000

Graph Throughput vs Offered Load



Conclusion

This lab assignment gives insights to different CSMA techniques. Implementation of the schemes give greater understanding of the topic (how the protocols are different from each other in terms of performance, collision avoidance, throughput). As the entire assignment is developed using a simulated network environment creator (simpy), it can be tested against multiple stations (even 500) to get a proper idea of how the protocol works. The transfer of actual packets is not necessary here, because it's a simple check of the efficiency of a channel which can be done by blocking the resource for a certain amount of time. The other benefit of using this discrete event dispatch module is that it is much more efficient as compared to implementing it by multiprocessing or multithreading which involves intervention of the OS and CPU clock cycles.