NETWORK LAB REPORT

CO3: Implement 1-persistent, non-persistent and p-persistent CSMA techniques.

RITABROTO GANGULY

BCSE-III (2019-2023) 5th sem, Section: A-3

Roll: 001910501090 Date: 27/10/2021

ASSIGNMENT-3

Implement 1-persistent, non-persistent and p-persistent CSMA techniques.

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

One-persistent CSMA:

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.

The problem with this method is that there are a large number of chances for the collision because there is a chance when two or more stations found a channel in an idle state and the transmit frames at the same time. At the time when a collision occurs, the station has to wait for the random time for the channel to be idle and to start all again.

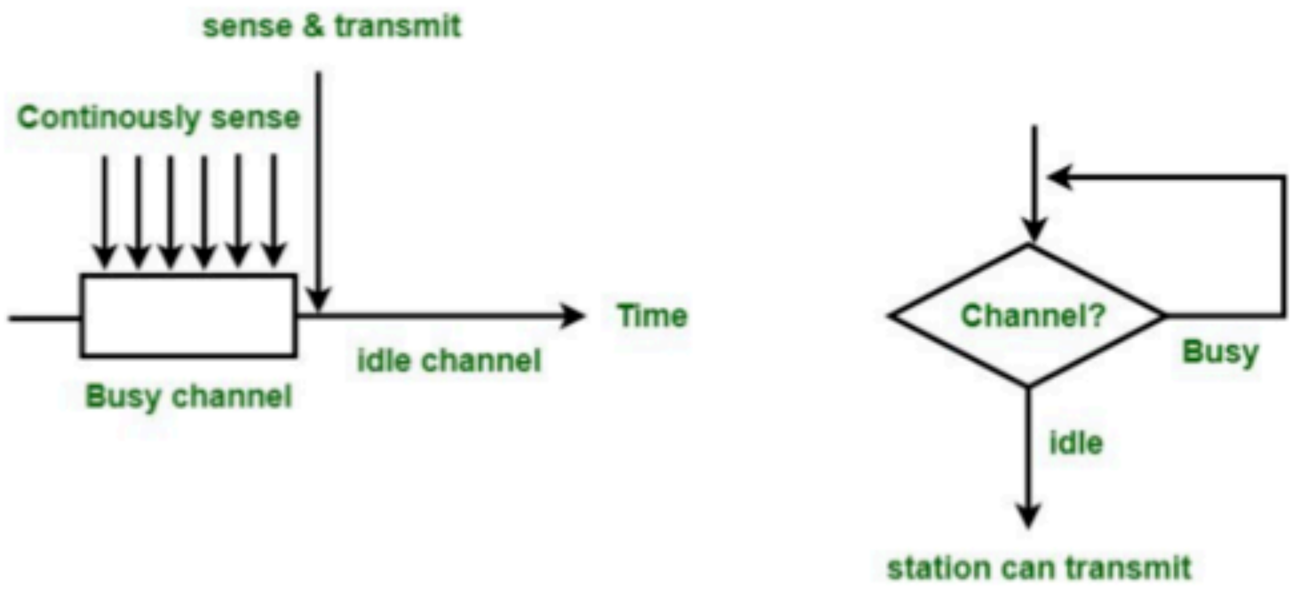
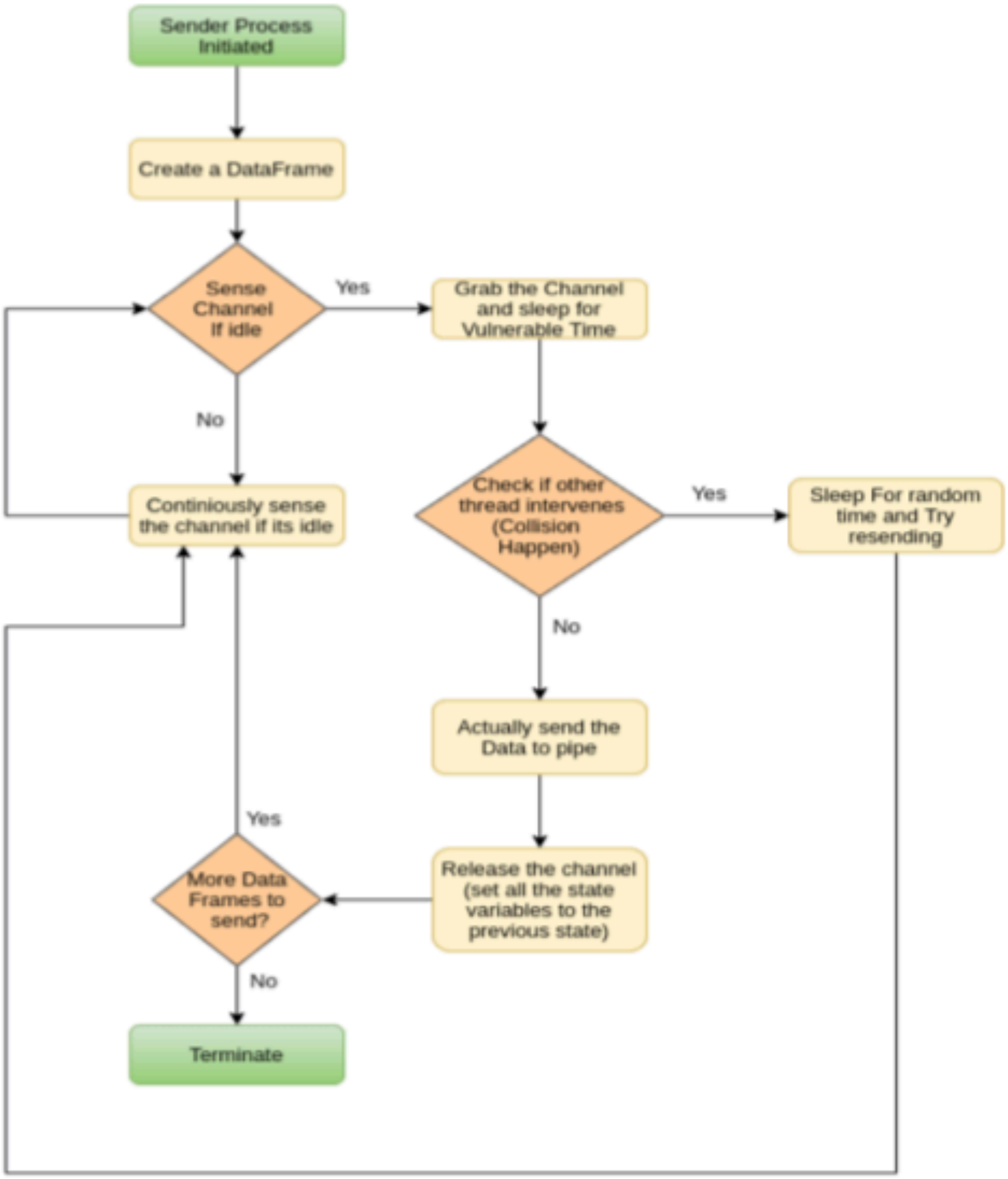


FIg : One persistent CSMA

FLOWCHART :



Sender Process Initiated
Create a DataFrame
Sense Channel If idle
Yes
Grab the Channel and sleep for Vulnerable Time
No
Continiously sense the channel if its idle
Check if other thread intervenes (Collision Happen)
Yes
Sleep For random time and Try resending
No
Actually send the Data to pipe
Yes
More Data Frames to send?
Release the channel (set all the state variables to the previous state)
No
Terminate

Fig : Implementation of One-persistent CSMA

Non-persistent CSMA:

In this method, the station which has frames to send, only that station senses the channel. In case of an idle channel, it will send a frame immediately to that channel. In case when the channel is found busy, it will wait for the random time and again sense the state of the station whether idle or busy. In this method, the station does not immediately 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.

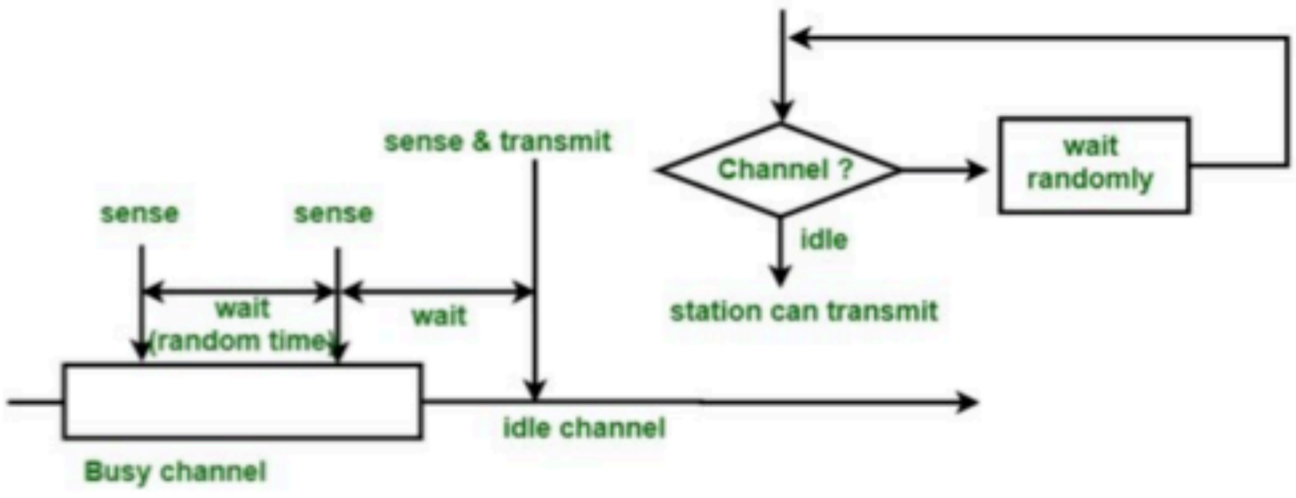


FIg : Non persistent CSMA

FLOWCHART :

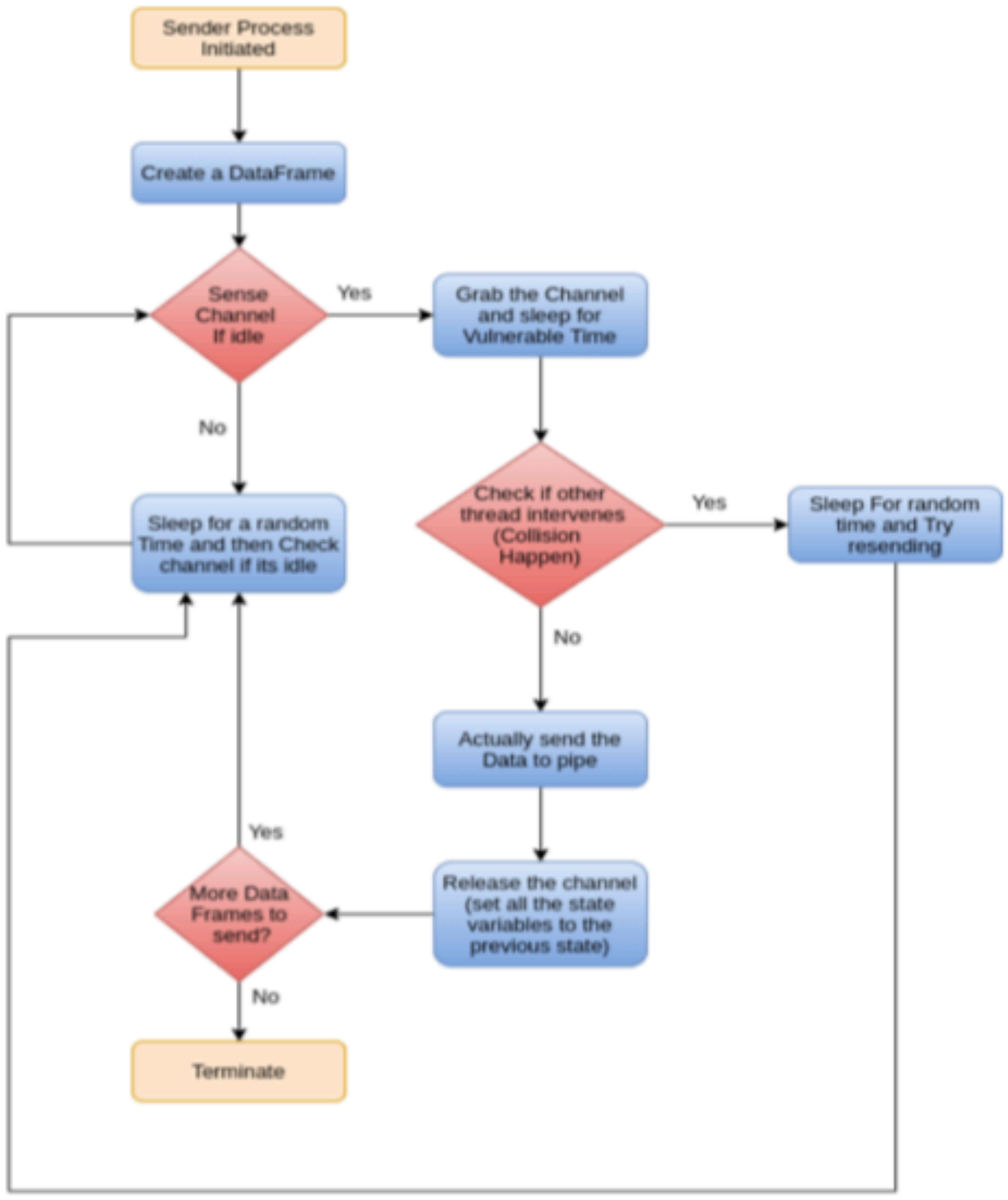

Sender Process Initiated
Create a DataFrame
Sense Channel If idle
Yes
No
Grab the Channel and sleep for Vulnerable Time
Sleep for a random Time and then Check channel if its idle
Check if other thread intervenes (Collision Happen)
Yes
Sleep For random time and Try resending
No
Actually send the Data to pipe
Release the channel (set all the state variables to the previous state)
Yes
More Data Frames to send?
No
Terminate

Fig : Implementation of Non-persistent CSMA

P (1/n) -persistent CSMA:

This is the method that is used when the 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 is found to be busy, the channel will wait for the next slot. If the channel is 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.

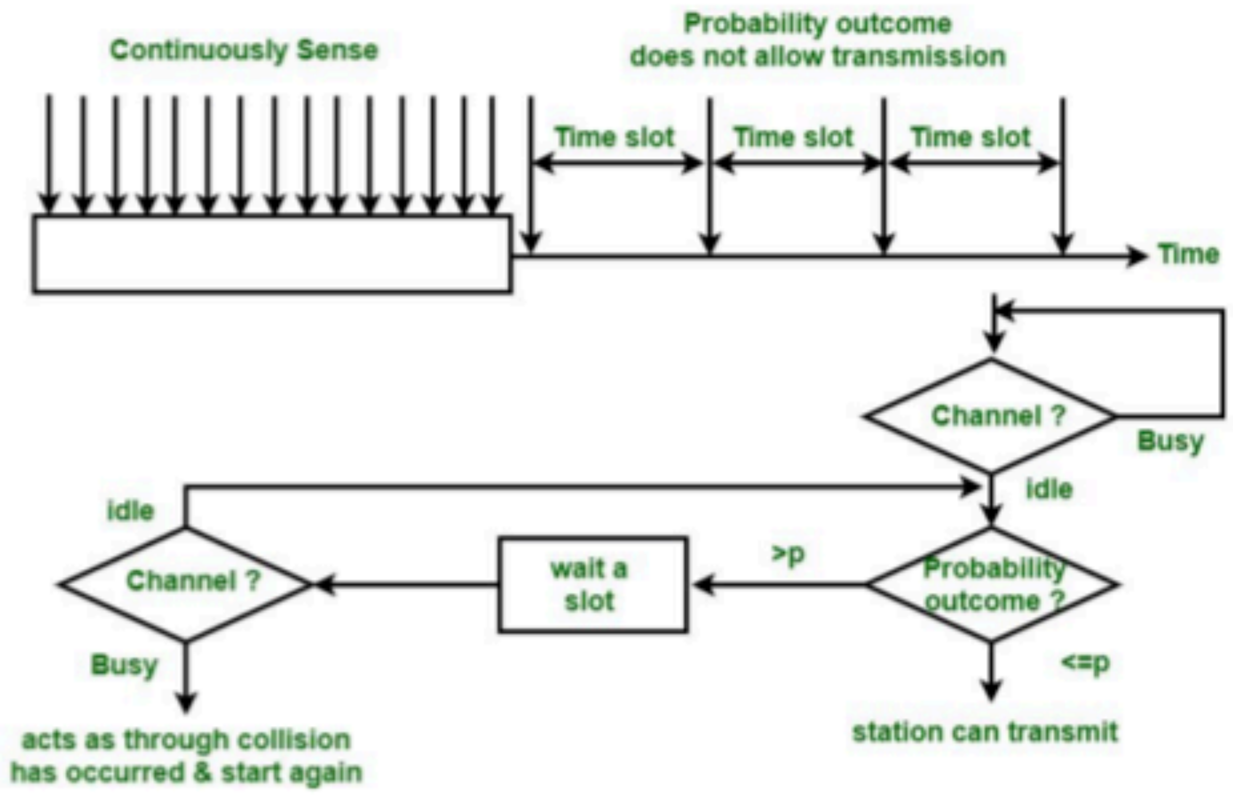


FIg : P persistent CSMA

FLOWCHART :

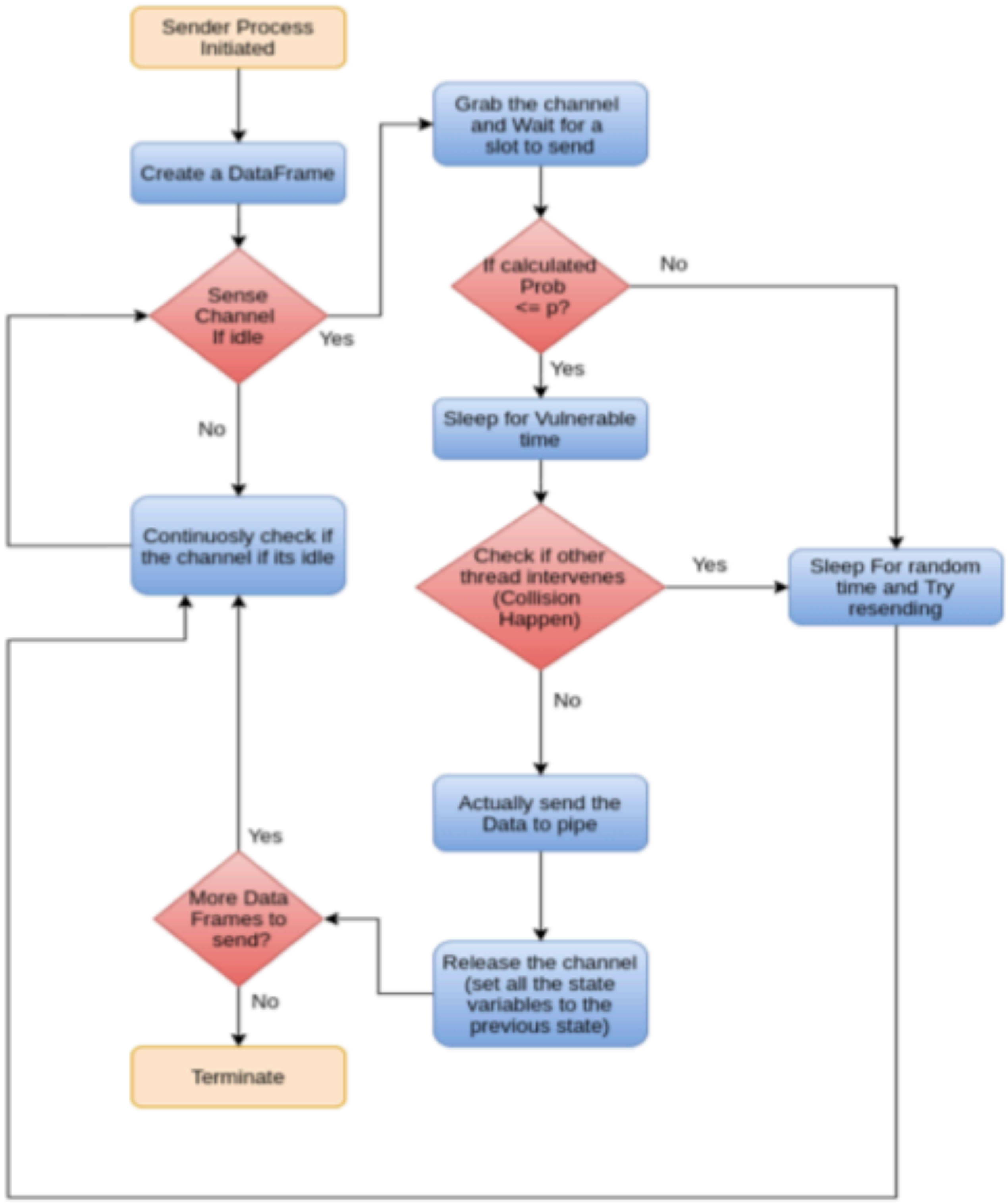

Sender Process Initiated
Create a DataFrame
Sense Channel If idle
Yes
No
Continuosly check if the channel if its idle
Grab the channel and Wait for a slot to send
If calculated Prob <= p?
No
Yes
Sleep for Vulnerable time
Check if other thread intervenes (Collision Happen)
Yes
Sleep For random time and Try resending
No
Actually send the Data to pipe
Release the channel (set all the state variables to the previous state)
More Data Frames to send?
Yes
No
Terminate

Fig : Implementation of P-persistent CSMA

SCHEMATIC DIAGRAM (as a part of DESIGN)

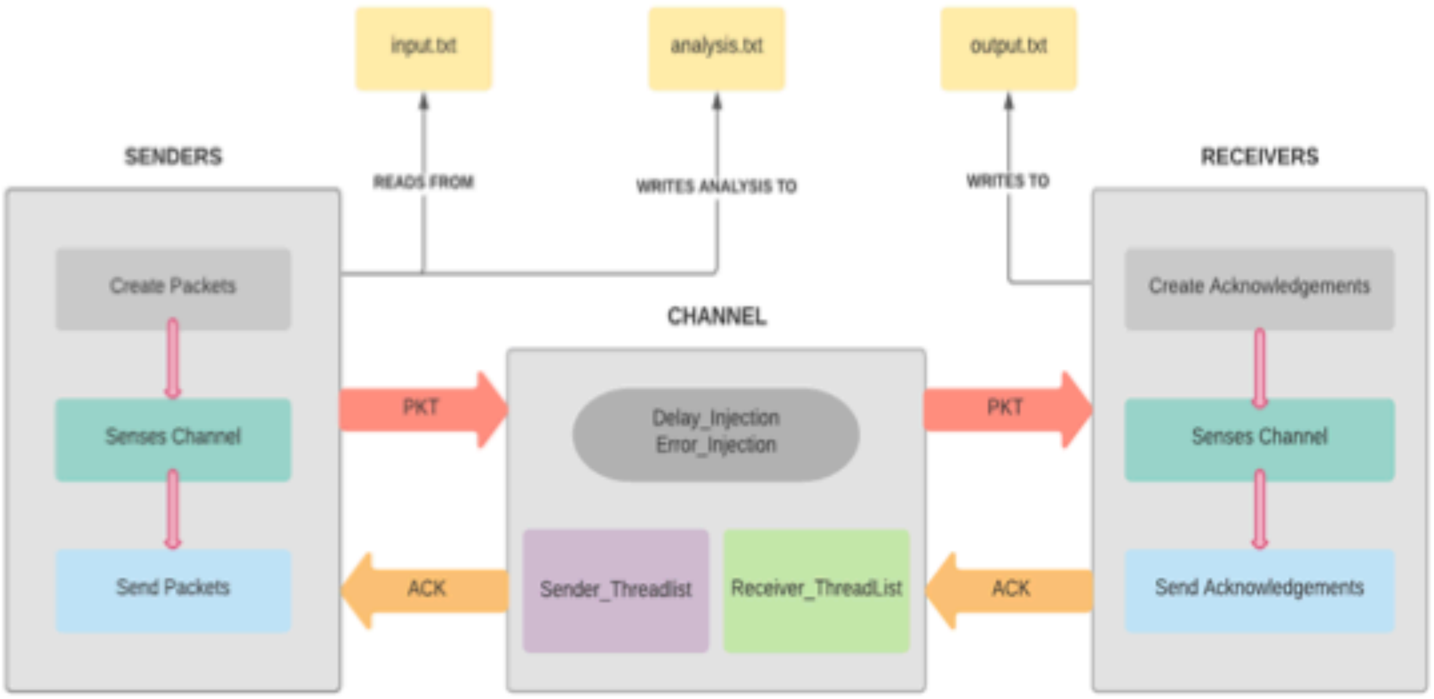


Fig : Schematic diagram to understand the flow of code for CSMA

IMPLEMENTATION

A. Packet Structure:

For better generalization and uniformity, **IEEE 802.3** frame format has been used throughout the communication. The fields are **`Preamble + SFD + DEST_MAC + SRC_MAC + Node_Info (custom addition) + DATA + CheckSum`** **(7 + 1 + 6 + 6 + 2 +46 + 4 = 72 Bytes).** Each data packet is encoded with **CheckSum-32** for detecting transmission errors.

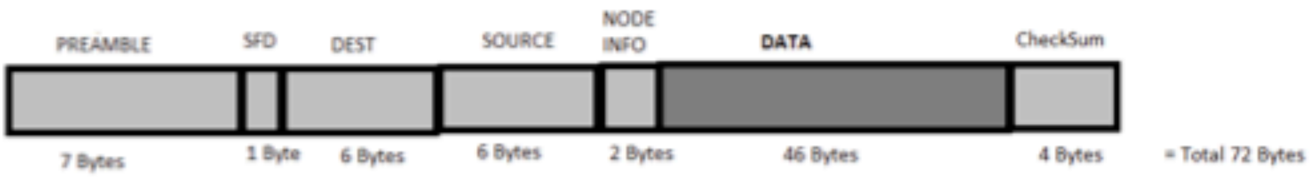


Fig : IEEE 802.3 Frame Format

B. Algorithm for Sender:

```
select_receiver( )
make_packets_to_send_by_one/non/P-persistent( )
sense_signal_to_check_if_busy_or_not( )
start_sending_packets( )
receive_confirmation_from_reciver( )
generate_report_for_performance_analysis( )
```

C. Algorithm for Channel:

```
channelize_packets_from_sender_to_receiver( )
channelize_response_from_receiver_to_sender( )
start_the_threads_for_data_sending_and_response_receiving( )
```

D. Algorithm for Receiver:

```
decode_which_sender_the_packet_is_coming_from( )
receive_the_packet( )
send_packet_reception_confirmation_to_specific_address( )
```

SOURCE CODE STRUCTURE

The Source-Code folder (`src`) contains the following scripts-

- `const.py:`

  All the constants (number of threads, time-out, packet-size, etc) are defined here.

- `checker.py:`

  Contains functions for CheckSum generation and ErrorChecking.

- `gen_packet.py:`

  Packet class containing methods for generating packets maintaining IEEE 802.3 format, extracting the original data from the generated packets, decode the source and destination mac addresses, checking errors by CheckSum algorithm, etc.

- `sender.py:`

  Sender Class containing methods for selecting a receiver, reading data from the input file(s), selecting the sending method (one/non/p-persistent), adding the packets into stream/channel after sensing whether the channel is busy or not, etc.

- `channel.py:`

  Channel Class for channelizing packets from the sender, channelizing response from the receiver, starting the sender and receiving threads one by one in a queued manner.

- `receiver.py:`

  Receiver Class for extracting the data from the receiver packet, writing the confirmation into the output file, etc.

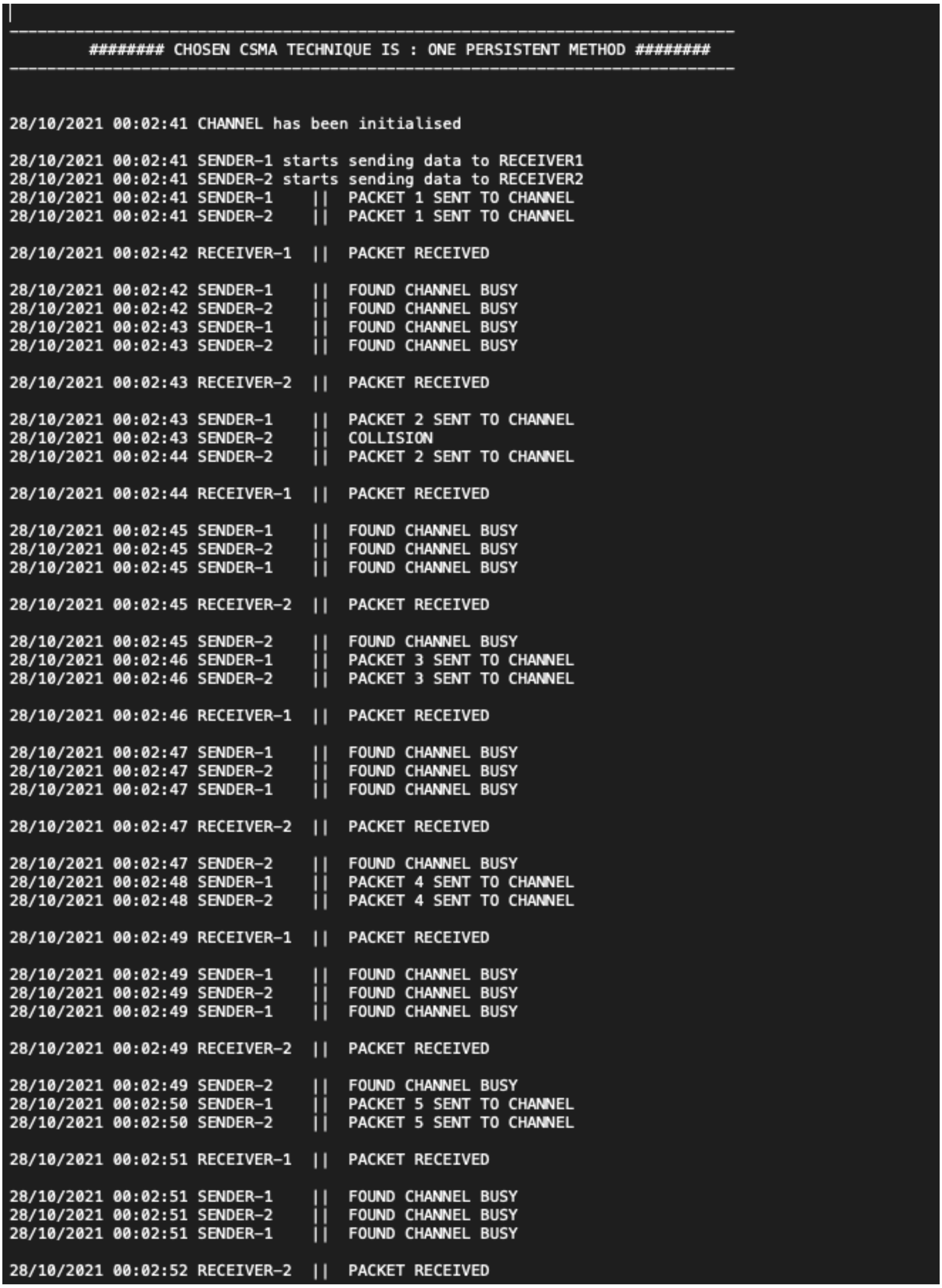
- `main.py:`

  Contains support for adding the sender and receiver connections to separate lists one by one, adding the sender and receiver threads to separate lists one by one, and starting the sending of packets based on the chosen option and generating a performance report.

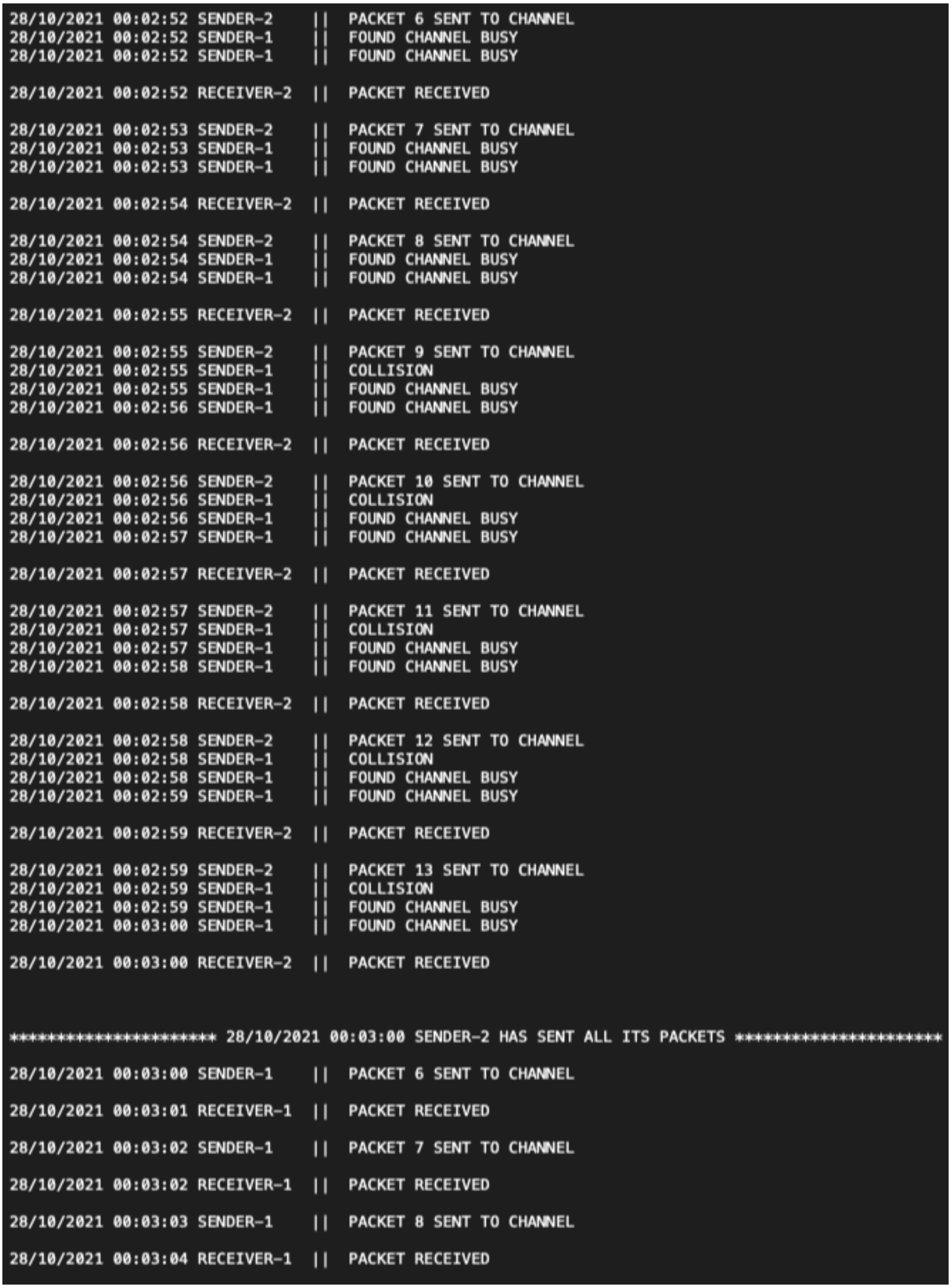
- `/textfiles :`

  Folder containing input and output files, and the report file (written number of packets sent, received and total delay and calculated throughput).

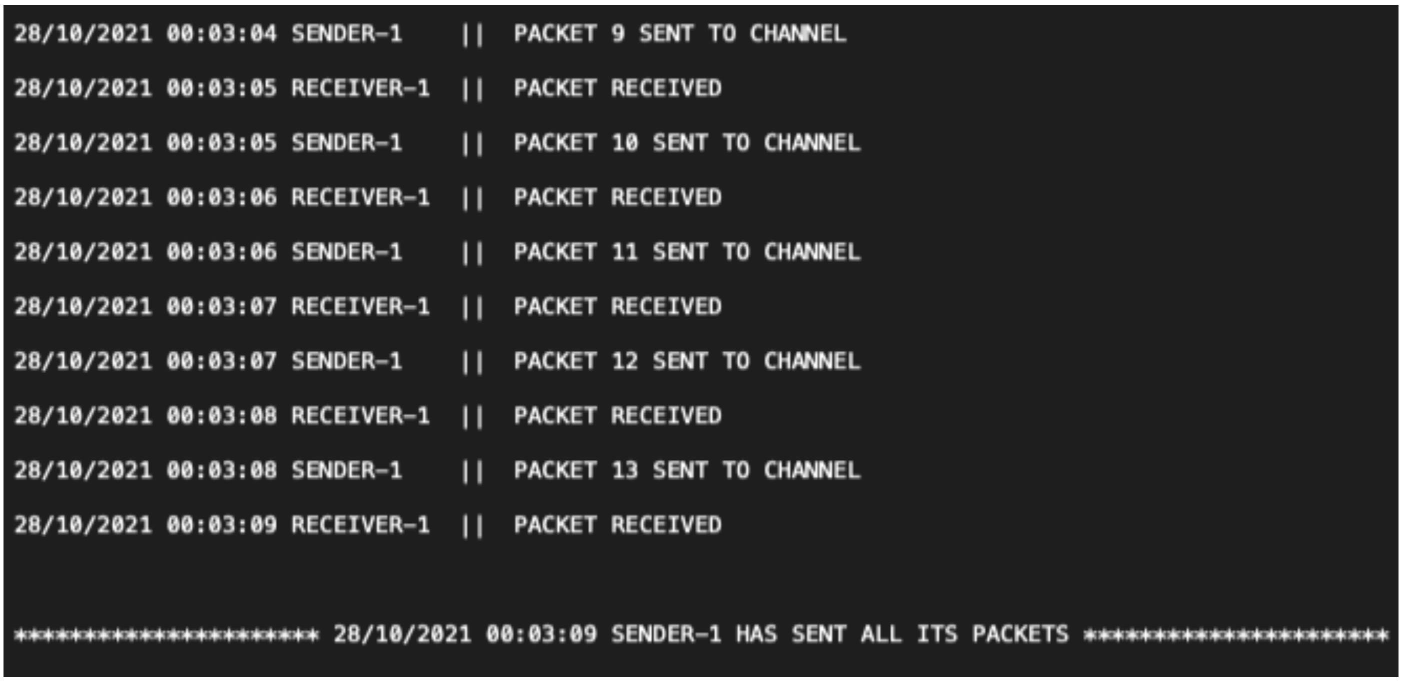
Outputs and Screenshots:

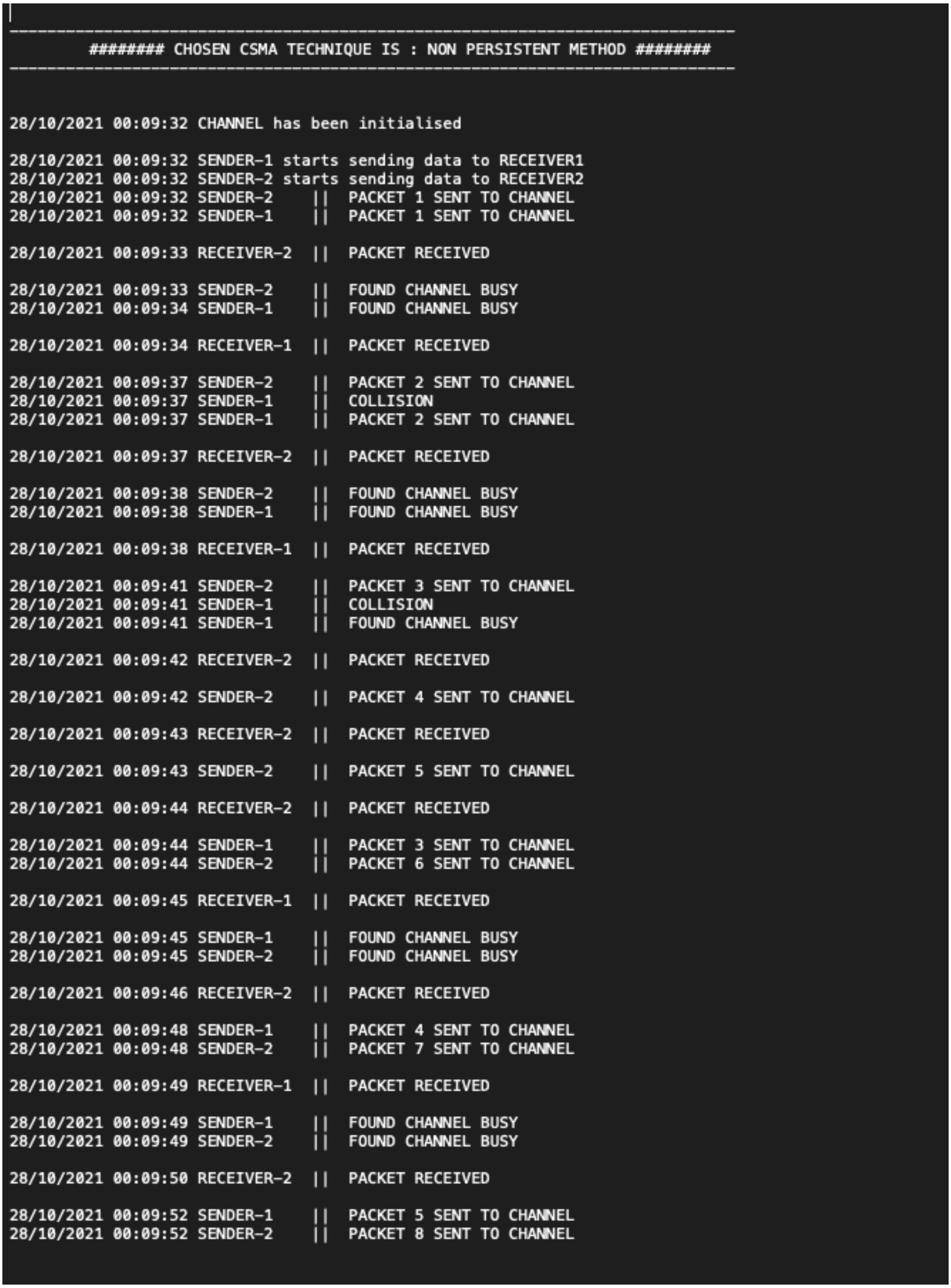


```
---------------------------------------------------------------------------
        ######## CHOSEN CSMA TECHNIQUE IS : ONE PERSISTENT METHOD ########
---------------------------------------------------------------------------


28/10/2021 00:02:41 CHANNEL has been initialised

28/10/2021 00:02:41 SENDER-1 starts sending data to RECEIVER1
28/10/2021 00:02:41 SENDER-2 starts sending data to RECEIVER2
28/10/2021 00:02:41 SENDER-1    ||  PACKET 1 SENT TO CHANNEL
28/10/2021 00:02:41 SENDER-2    ||  PACKET 1 SENT TO CHANNEL

28/10/2021 00:02:42 RECEIVER-1  ||  PACKET RECEIVED

28/10/2021 00:02:42 SENDER-1    ||  FOUND CHANNEL BUSY
28/10/2021 00:02:42 SENDER-2    ||  FOUND CHANNEL BUSY
28/10/2021 00:02:43 SENDER-1    ||  FOUND CHANNEL BUSY
28/10/2021 00:02:43 SENDER-2    ||  FOUND CHANNEL BUSY

28/10/2021 00:02:43 RECEIVER-2  ||  PACKET RECEIVED

28/10/2021 00:02:43 SENDER-1    ||  PACKET 2 SENT TO CHANNEL
28/10/2021 00:02:43 SENDER-2    ||  COLLISION
28/10/2021 00:02:44 SENDER-2    ||  PACKET 2 SENT TO CHANNEL

28/10/2021 00:02:44 RECEIVER-1  ||  PACKET RECEIVED

28/10/2021 00:02:45 SENDER-1    ||  FOUND CHANNEL BUSY
28/10/2021 00:02:45 SENDER-2    ||  FOUND CHANNEL BUSY
28/10/2021 00:02:45 SENDER-1    ||  FOUND CHANNEL BUSY

28/10/2021 00:02:45 RECEIVER-2  ||  PACKET RECEIVED

28/10/2021 00:02:45 SENDER-2    ||  FOUND CHANNEL BUSY
28/10/2021 00:02:46 SENDER-1    ||  PACKET 3 SENT TO CHANNEL
28/10/2021 00:02:46 SENDER-2    ||  PACKET 3 SENT TO CHANNEL

28/10/2021 00:02:46 RECEIVER-1  ||  PACKET RECEIVED

28/10/2021 00:02:47 SENDER-1    ||  FOUND CHANNEL BUSY
28/10/2021 00:02:47 SENDER-2    ||  FOUND CHANNEL BUSY
28/10/2021 00:02:47 SENDER-1    ||  FOUND CHANNEL BUSY

28/10/2021 00:02:47 RECEIVER-2  ||  PACKET RECEIVED

28/10/2021 00:02:47 SENDER-2    ||  FOUND CHANNEL BUSY
28/10/2021 00:02:48 SENDER-1    ||  PACKET 4 SENT TO CHANNEL
28/10/2021 00:02:48 SENDER-2    ||  PACKET 4 SENT TO CHANNEL

28/10/2021 00:02:49 RECEIVER-1  ||  PACKET RECEIVED

28/10/2021 00:02:49 SENDER-1    ||  FOUND CHANNEL BUSY
28/10/2021 00:02:49 SENDER-2    ||  FOUND CHANNEL BUSY
28/10/2021 00:02:49 SENDER-1    ||  FOUND CHANNEL BUSY

28/10/2021 00:02:49 RECEIVER-2  ||  PACKET RECEIVED

28/10/2021 00:02:49 SENDER-2    ||  FOUND CHANNEL BUSY
28/10/2021 00:02:50 SENDER-1    ||  PACKET 5 SENT TO CHANNEL
28/10/2021 00:02:50 SENDER-2    ||  PACKET 5 SENT TO CHANNEL

28/10/2021 00:02:51 RECEIVER-1  ||  PACKET RECEIVED

28/10/2021 00:02:51 SENDER-1    ||  FOUND CHANNEL BUSY
28/10/2021 00:02:51 SENDER-2    ||  FOUND CHANNEL BUSY
28/10/2021 00:02:51 SENDER-1    ||  FOUND CHANNEL BUSY

28/10/2021 00:02:52 RECEIVER-2  ||  PACKET RECEIVED
```

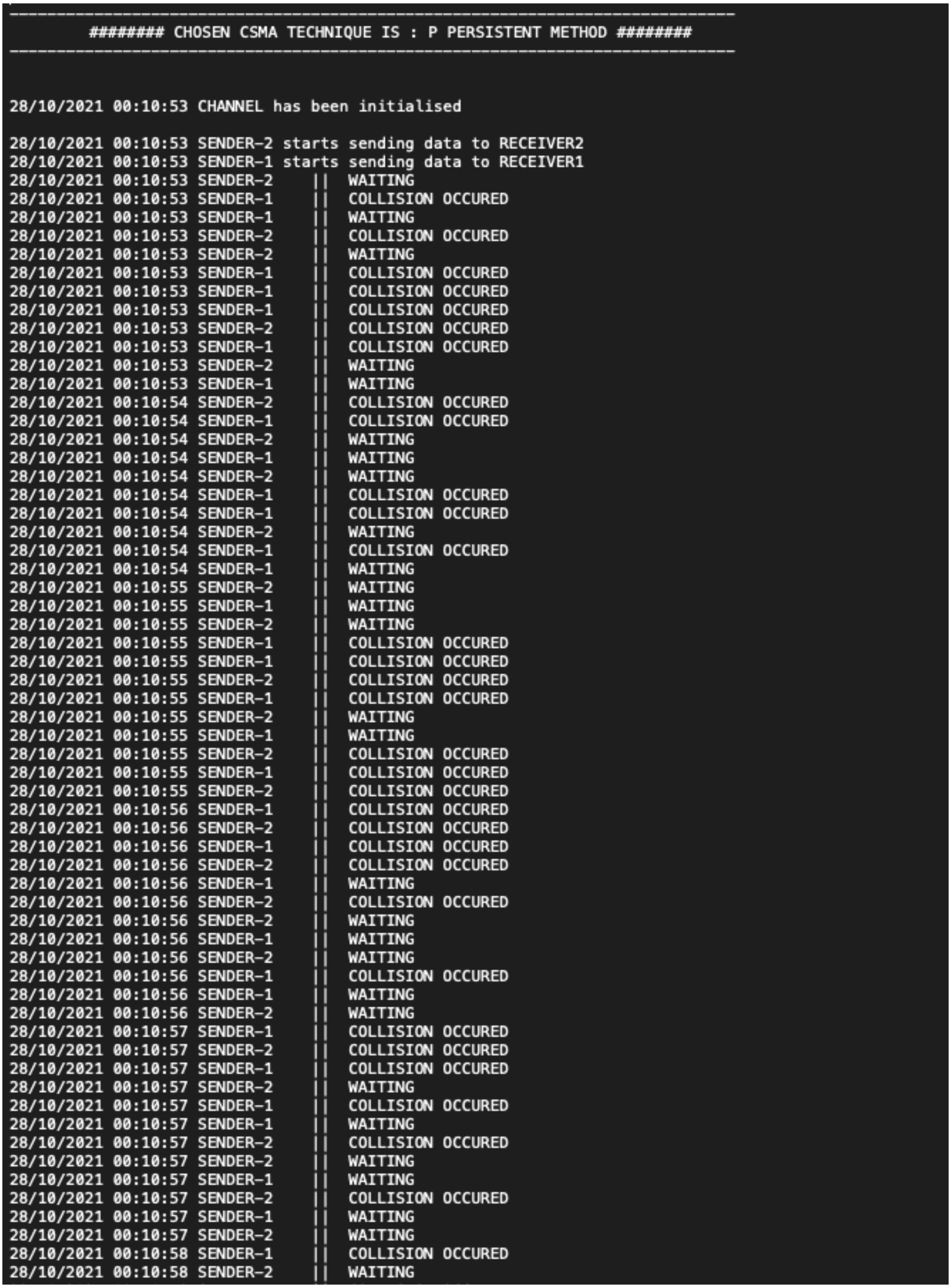


```
28/10/2021 00:02:52 SENDER-2    ||  PACKET 6 SENT TO CHANNEL
28/10/2021 00:02:52 SENDER-1    ||  FOUND CHANNEL BUSY
28/10/2021 00:02:52 SENDER-1    ||  FOUND CHANNEL BUSY

28/10/2021 00:02:52 RECEIVER-2  ||  PACKET RECEIVED

28/10/2021 00:02:53 SENDER-2    ||  PACKET 7 SENT TO CHANNEL
28/10/2021 00:02:53 SENDER-1    ||  FOUND CHANNEL BUSY
28/10/2021 00:02:53 SENDER-1    ||  FOUND CHANNEL BUSY

28/10/2021 00:02:54 RECEIVER-2  ||  PACKET RECEIVED

28/10/2021 00:02:54 SENDER-2    ||  PACKET 8 SENT TO CHANNEL
28/10/2021 00:02:54 SENDER-1    ||  FOUND CHANNEL BUSY
28/10/2021 00:02:54 SENDER-1    ||  FOUND CHANNEL BUSY

28/10/2021 00:02:55 RECEIVER-2  ||  PACKET RECEIVED

28/10/2021 00:02:55 SENDER-2    ||  PACKET 9 SENT TO CHANNEL
28/10/2021 00:02:55 SENDER-1    ||  COLLISION
28/10/2021 00:02:55 SENDER-1    ||  FOUND CHANNEL BUSY
28/10/2021 00:02:56 SENDER-1    ||  FOUND CHANNEL BUSY

28/10/2021 00:02:56 RECEIVER-2  ||  PACKET RECEIVED

28/10/2021 00:02:56 SENDER-2    ||  PACKET 10 SENT TO CHANNEL
28/10/2021 00:02:56 SENDER-1    ||  COLLISION
28/10/2021 00:02:56 SENDER-1    ||  FOUND CHANNEL BUSY
28/10/2021 00:02:57 SENDER-1    ||  FOUND CHANNEL BUSY

28/10/2021 00:02:57 RECEIVER-2  ||  PACKET RECEIVED

28/10/2021 00:02:57 SENDER-2    ||  PACKET 11 SENT TO CHANNEL
28/10/2021 00:02:57 SENDER-1    ||  COLLISION
28/10/2021 00:02:57 SENDER-1    ||  FOUND CHANNEL BUSY
28/10/2021 00:02:58 SENDER-1    ||  FOUND CHANNEL BUSY

28/10/2021 00:02:58 RECEIVER-2  ||  PACKET RECEIVED

28/10/2021 00:02:58 SENDER-2    ||  PACKET 12 SENT TO CHANNEL
28/10/2021 00:02:58 SENDER-1    ||  COLLISION
28/10/2021 00:02:58 SENDER-1    ||  FOUND CHANNEL BUSY
28/10/2021 00:02:59 SENDER-1    ||  FOUND CHANNEL BUSY

28/10/2021 00:02:59 RECEIVER-2  ||  PACKET RECEIVED

28/10/2021 00:02:59 SENDER-2    ||  PACKET 13 SENT TO CHANNEL
28/10/2021 00:02:59 SENDER-1    ||  COLLISION
28/10/2021 00:02:59 SENDER-1    ||  FOUND CHANNEL BUSY
28/10/2021 00:03:00 SENDER-1    ||  FOUND CHANNEL BUSY

28/10/2021 00:03:00 RECEIVER-2  ||  PACKET RECEIVED



********************** 28/10/2021 00:03:00 SENDER-2 HAS SENT ALL ITS PACKETS **********************

28/10/2021 00:03:00 SENDER-1    ||  PACKET 6 SENT TO CHANNEL

28/10/2021 00:03:01 RECEIVER-1  ||  PACKET RECEIVED

28/10/2021 00:03:02 SENDER-1    ||  PACKET 7 SENT TO CHANNEL

28/10/2021 00:03:02 RECEIVER-1  ||  PACKET RECEIVED

28/10/2021 00:03:03 SENDER-1    ||  PACKET 8 SENT TO CHANNEL

28/10/2021 00:03:04 RECEIVER-1  ||  PACKET RECEIVED
```

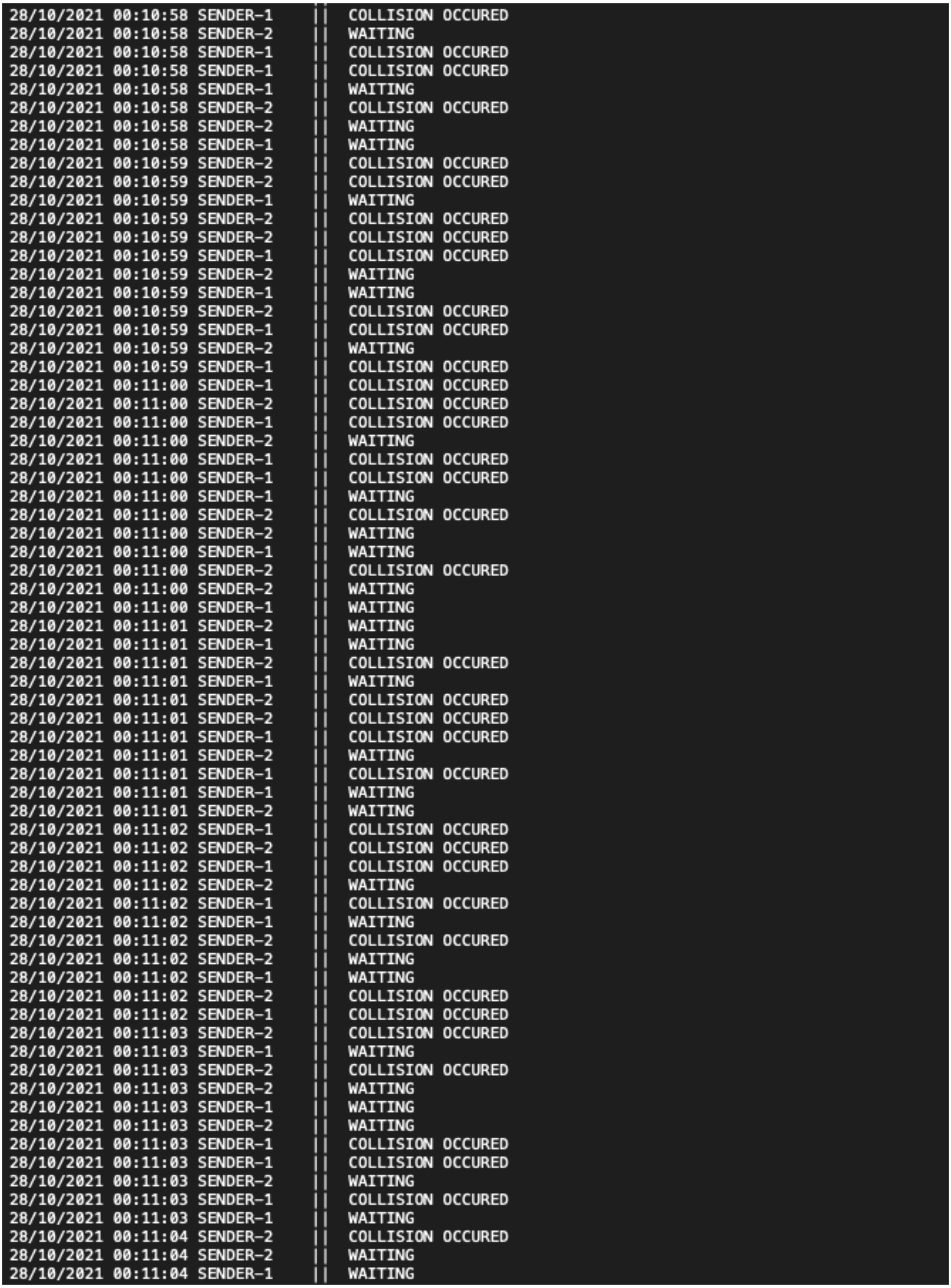

```
28/10/2021 00:03:04 SENDER-1    ||  PACKET 9 SENT TO CHANNEL

28/10/2021 00:03:05 RECEIVER-1  ||  PACKET RECEIVED

28/10/2021 00:03:05 SENDER-1    ||  PACKET 10 SENT TO CHANNEL

28/10/2021 00:03:06 RECEIVER-1  ||  PACKET RECEIVED

28/10/2021 00:03:06 SENDER-1    ||  PACKET 11 SENT TO CHANNEL

28/10/2021 00:03:07 RECEIVER-1  ||  PACKET RECEIVED

28/10/2021 00:03:07 SENDER-1    ||  PACKET 12 SENT TO CHANNEL

28/10/2021 00:03:08 RECEIVER-1  ||  PACKET RECEIVED

28/10/2021 00:03:08 SENDER-1    ||  PACKET 13 SENT TO CHANNEL

28/10/2021 00:03:09 RECEIVER-1  ||  PACKET RECEIVED


********************** 28/10/2021 00:03:09 SENDER-1 HAS SENT ALL ITS PACKETS **********************
```



```
--------------------------------------------------------------------------------
        ######## CHOSEN CSMA TECHNIQUE IS : NON PERSISTENT METHOD ########
--------------------------------------------------------------------------------


28/10/2021 00:09:32 CHANNEL has been initialised

28/10/2021 00:09:32 SENDER-1 starts sending data to RECEIVER1
28/10/2021 00:09:32 SENDER-2 starts sending data to RECEIVER2
28/10/2021 00:09:32 SENDER-2    ||  PACKET 1 SENT TO CHANNEL
28/10/2021 00:09:32 SENDER-1    ||  PACKET 1 SENT TO CHANNEL

28/10/2021 00:09:33 RECEIVER-2  ||  PACKET RECEIVED

28/10/2021 00:09:33 SENDER-2    ||  FOUND CHANNEL BUSY
28/10/2021 00:09:34 SENDER-1    ||  FOUND CHANNEL BUSY

28/10/2021 00:09:34 RECEIVER-1  ||  PACKET RECEIVED

28/10/2021 00:09:37 SENDER-2    ||  PACKET 2 SENT TO CHANNEL
28/10/2021 00:09:37 SENDER-1    ||  COLLISION
28/10/2021 00:09:37 SENDER-1    ||  PACKET 2 SENT TO CHANNEL

28/10/2021 00:09:37 RECEIVER-2  ||  PACKET RECEIVED

28/10/2021 00:09:38 SENDER-2    ||  FOUND CHANNEL BUSY
28/10/2021 00:09:38 SENDER-1    ||  FOUND CHANNEL BUSY

28/10/2021 00:09:38 RECEIVER-1  ||  PACKET RECEIVED

28/10/2021 00:09:41 SENDER-2    ||  PACKET 3 SENT TO CHANNEL
28/10/2021 00:09:41 SENDER-1    ||  COLLISION
28/10/2021 00:09:41 SENDER-1    ||  FOUND CHANNEL BUSY

28/10/2021 00:09:42 RECEIVER-2  ||  PACKET RECEIVED

28/10/2021 00:09:42 SENDER-2    ||  PACKET 4 SENT TO CHANNEL

28/10/2021 00:09:43 RECEIVER-2  ||  PACKET RECEIVED

28/10/2021 00:09:43 SENDER-2    ||  PACKET 5 SENT TO CHANNEL

28/10/2021 00:09:44 RECEIVER-2  ||  PACKET RECEIVED

28/10/2021 00:09:44 SENDER-1    ||  PACKET 3 SENT TO CHANNEL
28/10/2021 00:09:44 SENDER-2    ||  PACKET 6 SENT TO CHANNEL

28/10/2021 00:09:45 RECEIVER-1  ||  PACKET RECEIVED

28/10/2021 00:09:45 SENDER-1    ||  FOUND CHANNEL BUSY
28/10/2021 00:09:45 SENDER-2    ||  FOUND CHANNEL BUSY

28/10/2021 00:09:46 RECEIVER-2  ||  PACKET RECEIVED

28/10/2021 00:09:48 SENDER-1    ||  PACKET 4 SENT TO CHANNEL
28/10/2021 00:09:48 SENDER-2    ||  PACKET 7 SENT TO CHANNEL

28/10/2021 00:09:49 RECEIVER-1  ||  PACKET RECEIVED

28/10/2021 00:09:49 SENDER-1    ||  FOUND CHANNEL BUSY
28/10/2021 00:09:49 SENDER-2    ||  FOUND CHANNEL BUSY

28/10/2021 00:09:50 RECEIVER-2  ||  PACKET RECEIVED

28/10/2021 00:09:52 SENDER-1    ||  PACKET 5 SENT TO CHANNEL
28/10/2021 00:09:52 SENDER-2    ||  PACKET 8 SENT TO CHANNEL
```



```
-------------------------------------------------------------------------------
        ######## CHOSEN CSMA TECHNIQUE IS : P PERSISTENT METHOD ########
-------------------------------------------------------------------------------



28/10/2021 00:10:53 CHANNEL has been initialised

28/10/2021 00:10:53 SENDER-2 starts sending data to RECEIVER2
28/10/2021 00:10:53 SENDER-1 starts sending data to RECEIVER1
28/10/2021 00:10:53 SENDER-2      ||  WAITING
28/10/2021 00:10:53 SENDER-1      ||  COLLISION OCCURED
28/10/2021 00:10:53 SENDER-1      ||  WAITING
28/10/2021 00:10:53 SENDER-2      ||  COLLISION OCCURED
28/10/2021 00:10:53 SENDER-2      ||  WAITING
28/10/2021 00:10:53 SENDER-1      ||  COLLISION OCCURED
28/10/2021 00:10:53 SENDER-1      ||  COLLISION OCCURED
28/10/2021 00:10:53 SENDER-1      ||  COLLISION OCCURED
28/10/2021 00:10:53 SENDER-2      ||  COLLISION OCCURED
28/10/2021 00:10:53 SENDER-1      ||  COLLISION OCCURED
28/10/2021 00:10:53 SENDER-2      ||  WAITING
28/10/2021 00:10:53 SENDER-1      ||  WAITING
28/10/2021 00:10:54 SENDER-2      ||  COLLISION OCCURED
28/10/2021 00:10:54 SENDER-1      ||  COLLISION OCCURED
28/10/2021 00:10:54 SENDER-2      ||  WAITING
28/10/2021 00:10:54 SENDER-1      ||  WAITING
28/10/2021 00:10:54 SENDER-2      ||  WAITING
28/10/2021 00:10:54 SENDER-1      ||  COLLISION OCCURED
28/10/2021 00:10:54 SENDER-1      ||  COLLISION OCCURED
28/10/2021 00:10:54 SENDER-2      ||  WAITING
28/10/2021 00:10:54 SENDER-1      ||  COLLISION OCCURED
28/10/2021 00:10:54 SENDER-1      ||  WAITING
28/10/2021 00:10:55 SENDER-2      ||  WAITING
28/10/2021 00:10:55 SENDER-1      ||  WAITING
28/10/2021 00:10:55 SENDER-2      ||  WAITING
28/10/2021 00:10:55 SENDER-1      ||  COLLISION OCCURED
28/10/2021 00:10:55 SENDER-1      ||  COLLISION OCCURED
28/10/2021 00:10:55 SENDER-2      ||  COLLISION OCCURED
28/10/2021 00:10:55 SENDER-1      ||  COLLISION OCCURED
28/10/2021 00:10:55 SENDER-2      ||  WAITING
28/10/2021 00:10:55 SENDER-1      ||  WAITING
28/10/2021 00:10:55 SENDER-2      ||  COLLISION OCCURED
28/10/2021 00:10:55 SENDER-1      ||  COLLISION OCCURED
28/10/2021 00:10:55 SENDER-2      ||  COLLISION OCCURED
28/10/2021 00:10:56 SENDER-1      ||  COLLISION OCCURED
28/10/2021 00:10:56 SENDER-2      ||  COLLISION OCCURED
28/10/2021 00:10:56 SENDER-1      ||  COLLISION OCCURED
28/10/2021 00:10:56 SENDER-2      ||  COLLISION OCCURED
28/10/2021 00:10:56 SENDER-1      ||  WAITING
28/10/2021 00:10:56 SENDER-2      ||  COLLISION OCCURED
28/10/2021 00:10:56 SENDER-2      ||  WAITING
28/10/2021 00:10:56 SENDER-1      ||  WAITING
28/10/2021 00:10:56 SENDER-2      ||  WAITING
28/10/2021 00:10:56 SENDER-1      ||  COLLISION OCCURED
28/10/2021 00:10:56 SENDER-1      ||  WAITING
28/10/2021 00:10:56 SENDER-2      ||  WAITING
28/10/2021 00:10:57 SENDER-1      ||  COLLISION OCCURED
28/10/2021 00:10:57 SENDER-2      ||  COLLISION OCCURED
28/10/2021 00:10:57 SENDER-1      ||  COLLISION OCCURED
28/10/2021 00:10:57 SENDER-2      ||  WAITING
28/10/2021 00:10:57 SENDER-1      ||  COLLISION OCCURED
28/10/2021 00:10:57 SENDER-1      ||  WAITING
28/10/2021 00:10:57 SENDER-2      ||  COLLISION OCCURED
28/10/2021 00:10:57 SENDER-2      ||  WAITING
28/10/2021 00:10:57 SENDER-1      ||  WAITING
28/10/2021 00:10:57 SENDER-2      ||  COLLISION OCCURED
28/10/2021 00:10:57 SENDER-1      ||  WAITING
28/10/2021 00:10:57 SENDER-2      ||  WAITING
28/10/2021 00:10:58 SENDER-1      ||  COLLISION OCCURED
28/10/2021 00:10:58 SENDER-2      ||  WAITING
```



```
28/10/2021 00:10:58 SENDER-1     ||  COLLISION OCCURED
28/10/2021 00:10:58 SENDER-2     ||  WAITING
28/10/2021 00:10:58 SENDER-1     ||  COLLISION OCCURED
28/10/2021 00:10:58 SENDER-1     ||  COLLISION OCCURED
28/10/2021 00:10:58 SENDER-1     ||  WAITING
28/10/2021 00:10:58 SENDER-2     ||  COLLISION OCCURED
28/10/2021 00:10:58 SENDER-2     ||  WAITING
28/10/2021 00:10:58 SENDER-1     ||  WAITING
28/10/2021 00:10:59 SENDER-2     ||  COLLISION OCCURED
28/10/2021 00:10:59 SENDER-2     ||  COLLISION OCCURED
28/10/2021 00:10:59 SENDER-1     ||  WAITING
28/10/2021 00:10:59 SENDER-2     ||  COLLISION OCCURED
28/10/2021 00:10:59 SENDER-2     ||  COLLISION OCCURED
28/10/2021 00:10:59 SENDER-1     ||  COLLISION OCCURED
28/10/2021 00:10:59 SENDER-2     ||  WAITING
28/10/2021 00:10:59 SENDER-1     ||  WAITING
28/10/2021 00:10:59 SENDER-2     ||  COLLISION OCCURED
28/10/2021 00:10:59 SENDER-1     ||  COLLISION OCCURED
28/10/2021 00:10:59 SENDER-2     ||  WAITING
28/10/2021 00:10:59 SENDER-1     ||  COLLISION OCCURED
28/10/2021 00:11:00 SENDER-1     ||  COLLISION OCCURED
28/10/2021 00:11:00 SENDER-2     ||  COLLISION OCCURED
28/10/2021 00:11:00 SENDER-1     ||  COLLISION OCCURED
28/10/2021 00:11:00 SENDER-2     ||  WAITING
28/10/2021 00:11:00 SENDER-1     ||  COLLISION OCCURED
28/10/2021 00:11:00 SENDER-1     ||  COLLISION OCCURED
28/10/2021 00:11:00 SENDER-1     ||  WAITING
28/10/2021 00:11:00 SENDER-2     ||  COLLISION OCCURED
28/10/2021 00:11:00 SENDER-2     ||  WAITING
28/10/2021 00:11:00 SENDER-1     ||  WAITING
28/10/2021 00:11:00 SENDER-2     ||  COLLISION OCCURED
28/10/2021 00:11:00 SENDER-2     ||  WAITING
28/10/2021 00:11:00 SENDER-1     ||  WAITING
28/10/2021 00:11:01 SENDER-2     ||  WAITING
28/10/2021 00:11:01 SENDER-1     ||  WAITING
28/10/2021 00:11:01 SENDER-2     ||  COLLISION OCCURED
28/10/2021 00:11:01 SENDER-1     ||  WAITING
28/10/2021 00:11:01 SENDER-2     ||  COLLISION OCCURED
28/10/2021 00:11:01 SENDER-2     ||  COLLISION OCCURED
28/10/2021 00:11:01 SENDER-1     ||  COLLISION OCCURED
28/10/2021 00:11:01 SENDER-2     ||  WAITING
28/10/2021 00:11:01 SENDER-1     ||  COLLISION OCCURED
28/10/2021 00:11:01 SENDER-1     ||  WAITING
28/10/2021 00:11:01 SENDER-2     ||  WAITING
28/10/2021 00:11:02 SENDER-1     ||  COLLISION OCCURED
28/10/2021 00:11:02 SENDER-2     ||  COLLISION OCCURED
28/10/2021 00:11:02 SENDER-1     ||  COLLISION OCCURED
28/10/2021 00:11:02 SENDER-2     ||  WAITING
28/10/2021 00:11:02 SENDER-1     ||  COLLISION OCCURED
28/10/2021 00:11:02 SENDER-1     ||  WAITING
28/10/2021 00:11:02 SENDER-2     ||  COLLISION OCCURED
28/10/2021 00:11:02 SENDER-2     ||  WAITING
28/10/2021 00:11:02 SENDER-1     ||  WAITING
28/10/2021 00:11:02 SENDER-2     ||  COLLISION OCCURED
28/10/2021 00:11:02 SENDER-1     ||  COLLISION OCCURED
28/10/2021 00:11:03 SENDER-2     ||  COLLISION OCCURED
28/10/2021 00:11:03 SENDER-1     ||  WAITING
28/10/2021 00:11:03 SENDER-2     ||  COLLISION OCCURED
28/10/2021 00:11:03 SENDER-2     ||  WAITING
28/10/2021 00:11:03 SENDER-1     ||  WAITING
28/10/2021 00:11:03 SENDER-2     ||  WAITING
28/10/2021 00:11:03 SENDER-1     ||  COLLISION OCCURED
28/10/2021 00:11:03 SENDER-1     ||  COLLISION OCCURED
28/10/2021 00:11:03 SENDER-2     ||  WAITING
28/10/2021 00:11:03 SENDER-1     ||  COLLISION OCCURED
28/10/2021 00:11:03 SENDER-1     ||  WAITING
28/10/2021 00:11:04 SENDER-2     ||  COLLISION OCCURED
28/10/2021 00:11:04 SENDER-2     ||  WAITING
28/10/2021 00:11:04 SENDER-1     ||  WAITING
```

RESULTS and ANALYSIS

To calculate and get an idea about the avg bandwidth, I turn off all the delay elements in the channel and set the Channel to transmit data with a frame size of 72 bytes. It came out to be approximately 500 kbps.

- **BandWIdth Calculated = 500 Kbps**
- **Avg packets send from each sender = 13**

Number of Senders = n = 2

CSMA Technique	Avg. Delay Per Sender	Avg. Collision Per Sender	Avg. Throughput
One Persistent	23.795	2.50	0.8471
None Persistent	50.73	0.00	1.0000
1/n Persistent	26.86	0.00	1.0000

Number of Senders = n = 4

CSMA Technique	Avg. Delay Per Sender	Avg. Collision Per Sender	Avg. Throughput
One Persistent	44.807	2.00	0.8825
None Persistent	69.97	1.75	0.8825
1/n Persistent	42.27	2.50	0.8432

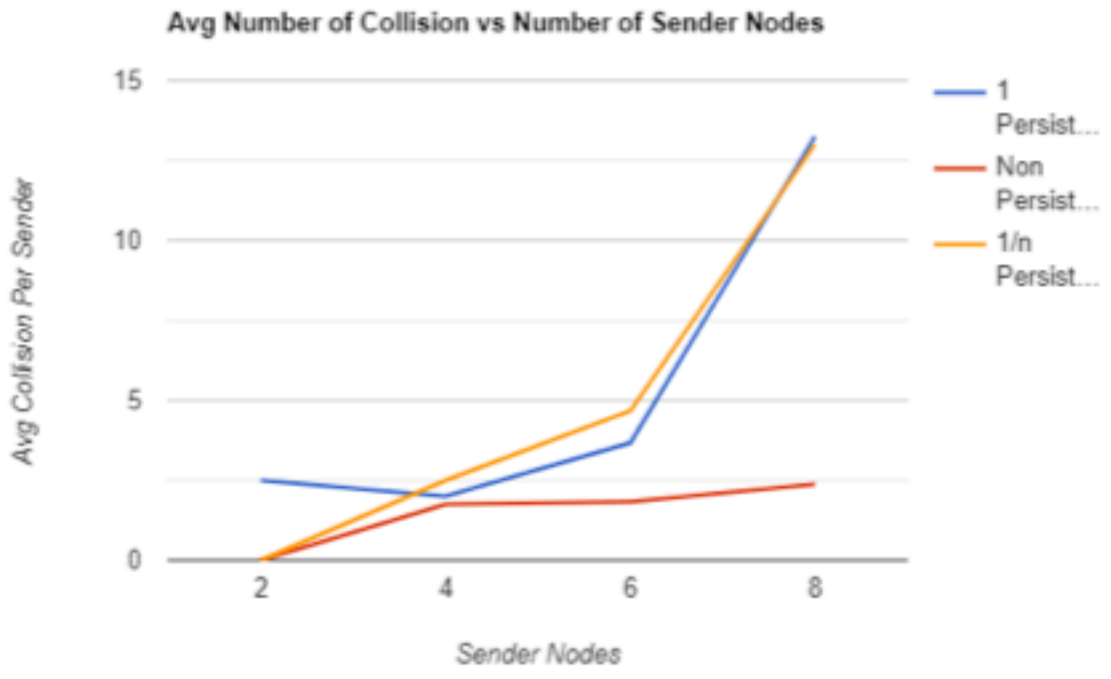
Number of Senders = n = 6

CSMA Technique	Avg. Delay Per Sender	Avg. Collision Per Sender	Avg. Throughput
One Persistent	61.30	3.67	0.7963
None Persistent	84.06	1.83	0.8800
1/n Persistent	66.93	4.67	0.749

Number of Senders = n = 8

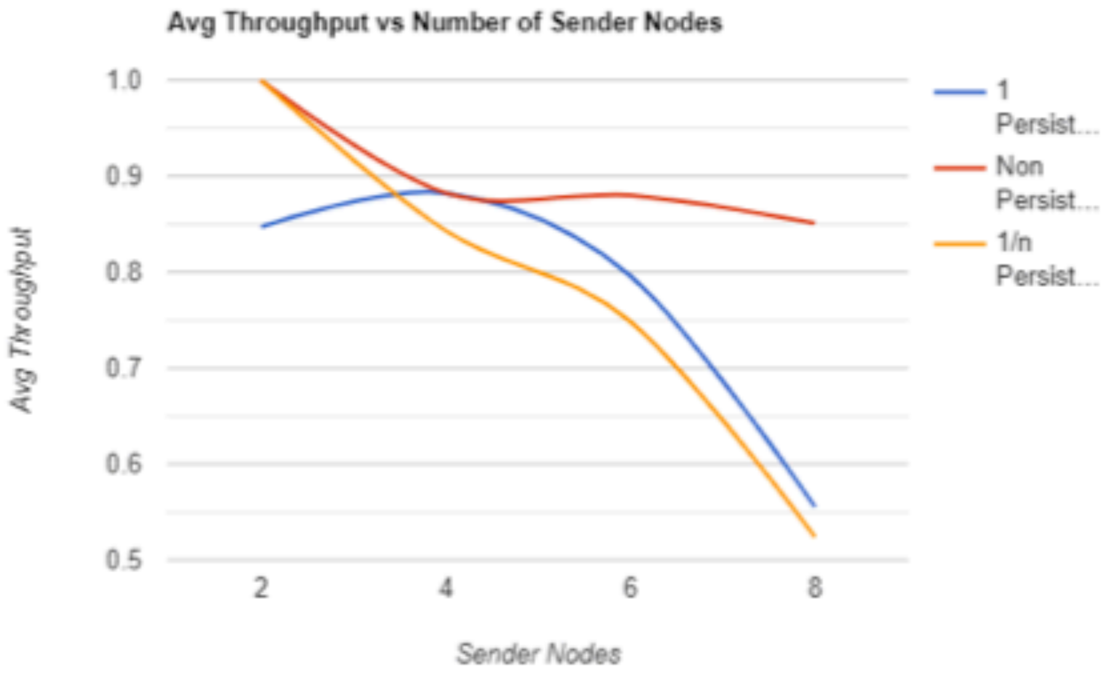
CSMA Technique	Avg. Delay Per Sender	Avg. Collision Per Sender	Avg. Throughput
One Persistent	79.74	13.25	0.5557
None Persistent	115.74	2.37	0.8511
1/n Persistent	81.31	13.00	0.5245

Comparison of Avg Number of Collision:



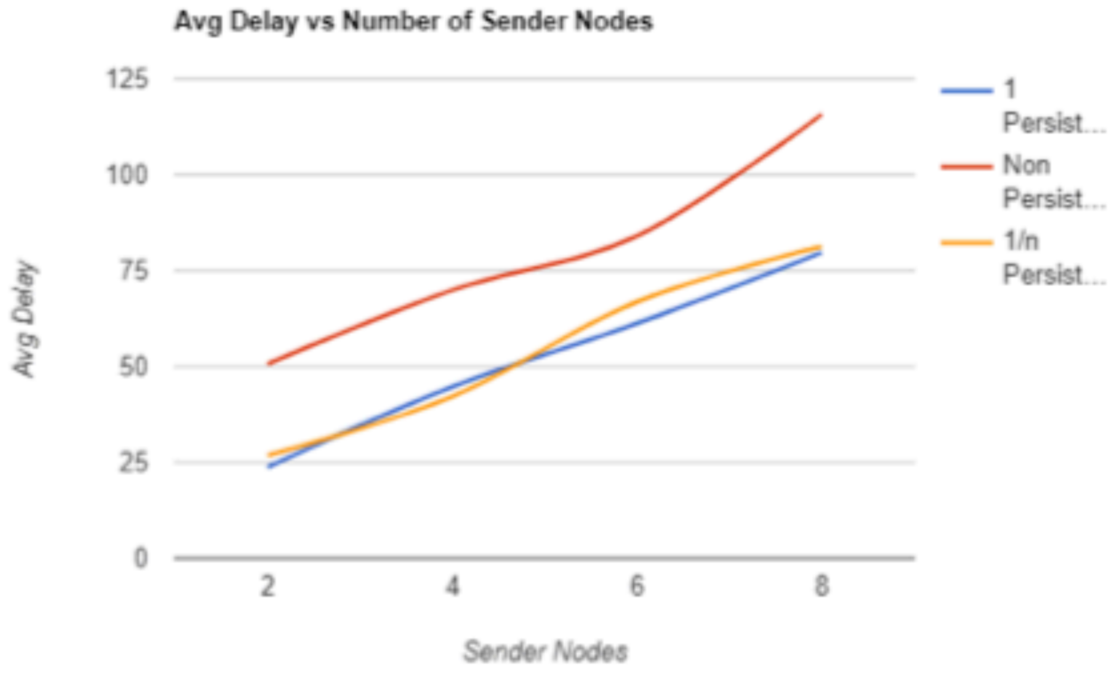
1/n persistent method shows the significant number of collisions and with a greater number of senders it just abruptly increases and almost shows **exponential growth**. The **One-persistent** method actually performs best at the medium number of senders and increases with the increasing number of senders but the **1/n persistent** method outperformed it gradually in the long run. The **Non-persistent** graph actually shows greater consistency throughout and managed to end up having the least number of collisions.

Comparison of Throughputs:



The throughput of a network system is defined as the number of successful transmissions per frame time. The **one persistent** method shows the worst possible throughput and quickly dies to almost zero as the number of senders increases. Interestingly enough **non-persistent** method works pretty well at higher numbers of senders. **1/n persistent** method shows similar performance to **non-persistent** in less number of senders but performs much better than the rest two with the increasing number of senders.

Comparison of Avg Delay:



The **Non-persistent** method shows the greatest delay of the bunch. For lower numbers of nodes, **One persisten**t method is the best. In the lower region **One-persistent** method actually able to beat the **1/n persisten**t method, however with increasing sender number it slightly underperforms but in the longer run, the **1/n persistent** method shows the best result and actually is the best of the bunch.

CONCLUSION

In the Collison comparison and Throughput comparison, The **Non-persistent** method actually outperforms both of the rest two methods but in the case of Delay comparison, it abruptly shows huge delay compared to the other two. When we compare the **One-persistent** and **1/n persistent** methods, we can see that the first one always outperforms the latter one in all of the above scenarios.

Scopes of Improvement

- In a relatively small network, the number of nodes falls somewhere between 10-30. So I should've considered more data points at a higher number of nodes.
- A more real-life approach would have been taken into consideration, the receiver nodes are also a part of this sensing algorithms. It would've provided more data points.
- I should've considered more data points regarding **non-persistent** CSMA varying the range of random sleep.

COMMENTS

This assignment has helped me in understanding the different CSMA protocols immensely, by researching and implementing them. It has also helped in understanding the demerits of a protocol, and how such demerits are overcome by other protocols.