National Institute Of Technology, Hamirpur

Department Of Computer Science and Engineering

Mobile Computing Lab (CSD 427)

Practical Assignment 4

Submitted to:

Dr. Naveen Chauhan

Submitted by:

Jahnvi Gupta

Roll No. 17MI503

CSE Dual Degree

Aim: Write a program for simulating following MAC Protocols: CSMA/CA, MACA and MACAW. Compare the performance of the protocols on various performance metrics.

CSMA/CA:

Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA) is a network protocol for carrier transmission that operates in the Medium Access Control (MAC) layer. In contrast to CSMA/CD (Carrier Sense Multiple Access/Collision Detection) that deals with collisions after their occurrence, CSMA/CA prevents collisions prior to their occurrence.

Algorithm for CSMA/CA:

The algorithm of CSMA/CA is:

- When a frame is ready, the transmitting station checks whether the channel is idle or busy.
- If the channel is busy, the station waits until the channel becomes idle.
- If the channel is idle, the station waits for an Inter-frame gap (IFG) amount of time and then sends the frame.
- After sending the frame, it sets a timer.
- The station then waits for acknowledgement from the receiver. If it receives the acknowledgement before the expiry of the timer, it marks a successful transmission.
- Otherwise, it waits for a back-off time period and restarts the algorithm.

Code:

```
import  socket
import time

reciever_Port = 4040
reciever_IP = "127.0.0.1"

station_Port = 4050
station_IP = "127.0.0.1"

recieverSocket = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
stationSocket = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
stationSocket.bind((station_IP, station_Port))

def main():
    c = 0
    frame_to_send = "hello, Bob!".encode()

    is_transmission_successful = False

    IFG_time = 2

    time_bound = 2
```

```
    max_c = 3

    while not is_transmission_successful:

        print(f'Try number:   {c + 1}')
        # wait for IFG_time
        time.sleep(IFG_time)

        cur_time = time.time()

        stationSocket.sendto(frame_to_send, (reciever_IP, reciever_Port))

        # start waiting for the reciever
        data = stationSocket.recvfrom(1024)

        turn_around_time = time.time() - cur_time

        is_transmission_successful = turn_around_time <= time_bound

        if not is_transmission_successful:
            print(f"Didn't recieve acknowledgement from reciever in the time
 bound.. trying again")

            c += 1

            if c > max_c:
                print(f"Couldn't send data to the reciever.. backed off {max
_c} times")
                break
    if is_transmission_successful:
        print("Successfully transmitted data to the reciever in the time bou
nd")


main()
```

Receiver Code:

```
import random
import  socket
import time
reciever_Port = 4040
reciever_IP = "127.0.0.1"
```

```
station_Port = 4050
station_IP = "127.0.0.1"


stationSocket = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)

recieverSocket = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
recieverSocket.bind((reciever_IP, reciever_Port))

while True:
    data, add = recieverSocket.recvfrom(1024)

    print(f"Frame recieved from client:    {data.decode()}")
    print("Sending acknowledgement")

    random_time = random.randint(0, 3)
    print(random_time)
    time.sleep(random_time)

    stationSocket.sendto("hi".encode(), (station_IP, station_Port))
```

Output:

MACA :

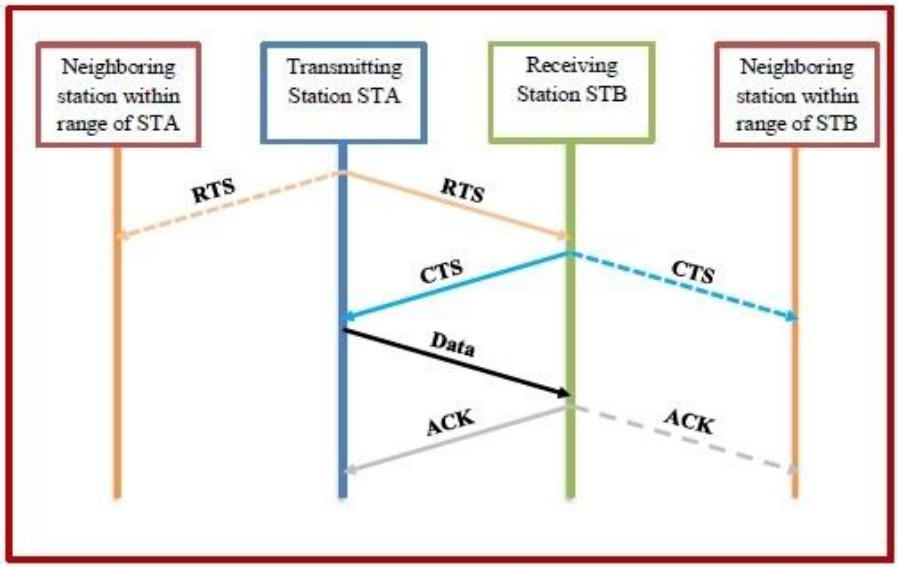
Multiple Access with Collision Avoidance (MACA) is a medium access control (MAC) layer protocol used in wireless networks, with a view to solving the hidden terminal problem. It also provides the solution to the exposed terminal problem. The MAC layer protocol IEEE 802.11 RTS/CTS has been adopted from MACA.

The MACA protocol works with the condition that the stations are synchronized and frame sizes and data speed are the same. It involves the transmission of two frames called RTS and CTS prior to data transmission. RTS stands for Request to Send and CTS stands for Clear to Send.

Let us consider that a transmitting station STA has a data frame to send to a receiving station STB. The operation works as follows:

- Station STA sends an RTS frame to the receiving station.
- On receiving the RTS, station STB replies by sending a CTS frame.
- On receipt of the CTS frame, station STA begins transmitting its data frame.
- After successful receipt of the data frame, station STB sends an ACK frame (acknowledgement frame).

The sequence is illustrated as follows:



Any station that can hear RTS is close to the transmitting station and remains silent long enough for the CTS, or waits for a certain time period. If the RTS is not followed by a CTS, the maximum waiting time is the RTS propagation time.

Any station that can hear the CTS is close to the receiving station and remains silent during the data transmission. It attempts for transmission after hearing the ACK.

MACA is a non-persistent slotted protocol. This implies that if the medium is detected as busy, a station waits for a random time period after the beginning of a time slot and then it sends an RTS. This assures fair access to the medium.

Code:

1. **Representation of a Terminal in python:**

```
import  math


class Terminal:
    algorithm = 'maca'

    def __init__(self, name, center_x, center_y, radius):
        self.name = name
        self.center_x = center_x
        self.center_y = center_y
        self.radius = radius
        self.nodes_in_coverage_range = []

    def in_coverage_range(self, other_terminal):
```

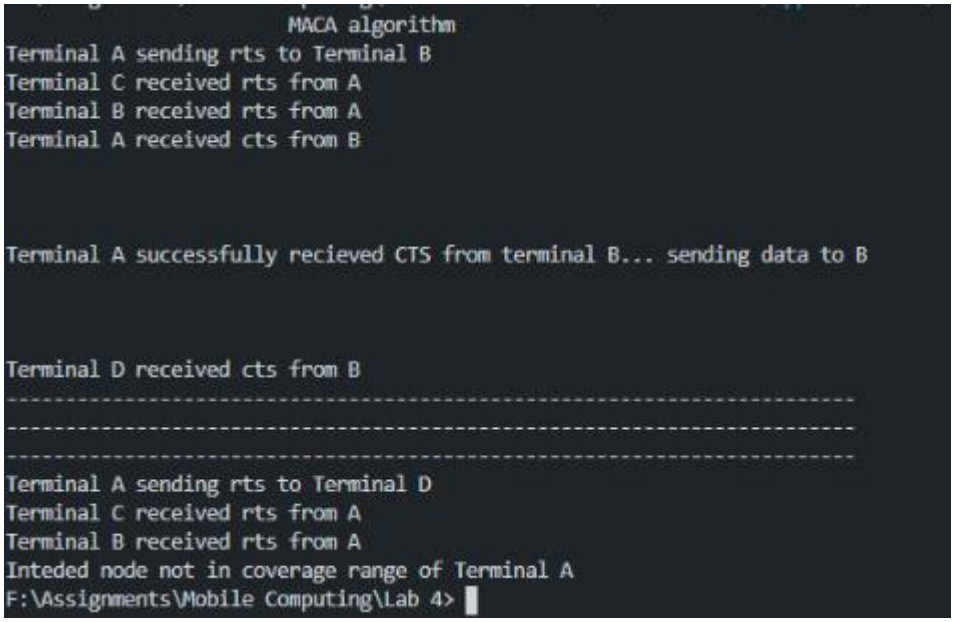
```
        dist_between_centers = math.sqrt((self.center_x -
 other_terminal.center_x) ** 2 + (self.center_y -
 other_terminal.center_y)**2)

        return dist_between_centers <= self.radius

    def send_rts(self, intedended_terminal):
        successfully_sent_rts = False
        print(f'Terminal {self.name} sending rts to Terminal {intedended_ter
minal}')

        for node in self.nodes_in_coverage_range:

            node.recieve_rts(self.name, intedended_terminal)
            if node.name == intedended_terminal:
                successfully_sent_rts = True


        if not successfully_sent_rts:
            print(f'Inteded node not in coverage range of Terminal {self.nam
e}')


    def recieve_rts(self, sender_name, intedended_terminal):
        print(f'Terminal {self.name} received rts from {sender_name}')

        if(self.name == intedended_terminal):
            self.send_cts(self.name, sender_name)


    def send_cts(self, source_terminal, dest_terminal):
        successfully_send_cts = False

        for node in self.nodes_in_coverage_range:
            node.recieve_cts(source_terminal, dest_terminal)
            if dest_terminal == node.name:
                successfully_send_cts = True


        if not successfully_send_cts:
            print(f'Terminal {dest_terminal} is not in coverage range of ter
minal {source_terminal}... couldnt send cts')


    def recieve_cts(self, source_terminal, dest_terminal):
        print(f'Terminal {self.name} received cts from {source_terminal}')
```

```
        if(dest_terminal == self.name):
            print('\n\n')
            print(f'Terminal {dest_terminal} successfully recieved CTS from
terminal {source_terminal}... sending data to {source_terminal}')
            print('\n\n')
            if Terminal.algorithm == 'macaw':
                self.send_ds(source_terminal)
            return

    def send_ds(self, dest_terminal):
        print(f'Terminal {self.name} is sending data send request to Termina
l {dest_terminal}')

        data_to_send = "Hello, Bob!"
        print(f'Data to send : {data_to_send}')
        self.send_data(data_to_send, dest_terminal)

    def send_data(self, data, dest_terminal):
        # find the terminal... call recieve data method on that terminal
        for terminal in self.nodes_in_coverage_range:
            if terminal.name == dest_terminal:
                terminal.recieve_data(data, self.name)
                break

    def recieve_data(self, data, source_terminal):
        print(f'Terminal {self.name} recieved data from Terminal {source_ter
minal}..\n Sending acknowledgement to Terminal {source_terminal}')
        self.send_ack(source_terminal)

    def send_ack(self, dest_terminal):
        # send ack to the terminal
        for terminal in self.nodes_in_coverage_range:
            if terminal.name == dest_terminal:
                print('\n\n')
                print(f'Terminal {terminal.name} recieved acknowledgement fr
om Terminal {self.name}')
                print('Data transfer successfully completed')
                print('\n\n')



    def __str__(self):
        return f'Terminal {self.name} : [Center : ({self.center_x}, {self.ce
nter_y})\tRadius : {self.radius}]'
```

Maca algorithm:

```
import Terminal

# send rts
def main():
    a = Terminal.Terminal("A", 2, 0, 2)
    b = Terminal.Terminal("B", 3, 0, 2)
    c = Terminal.Terminal("C", 0, 0, 2)
    d = Terminal.Terminal("D", 5, 0, 2)

    terminals = [c, a, b, d]

    print(f'\t\t\t{Terminal.Terminal.algorithm.upper()} algorithm')


    # print terminals in coverage range of a
    for terminal in terminals:
        for t in terminals:
            if terminal != t:
                if terminal.in_coverage_range(t):
                    terminal.nodes_in_coverage_range.append(t)



    a.send_rts("B")
    print('-----------------------------------------------------------------------
-------')
    print('-----------------------------------------------------------------------
-------')
    print('-----------------------------------------------------------------------
-------')

    a.send_rts("D")



main()
```

Output:

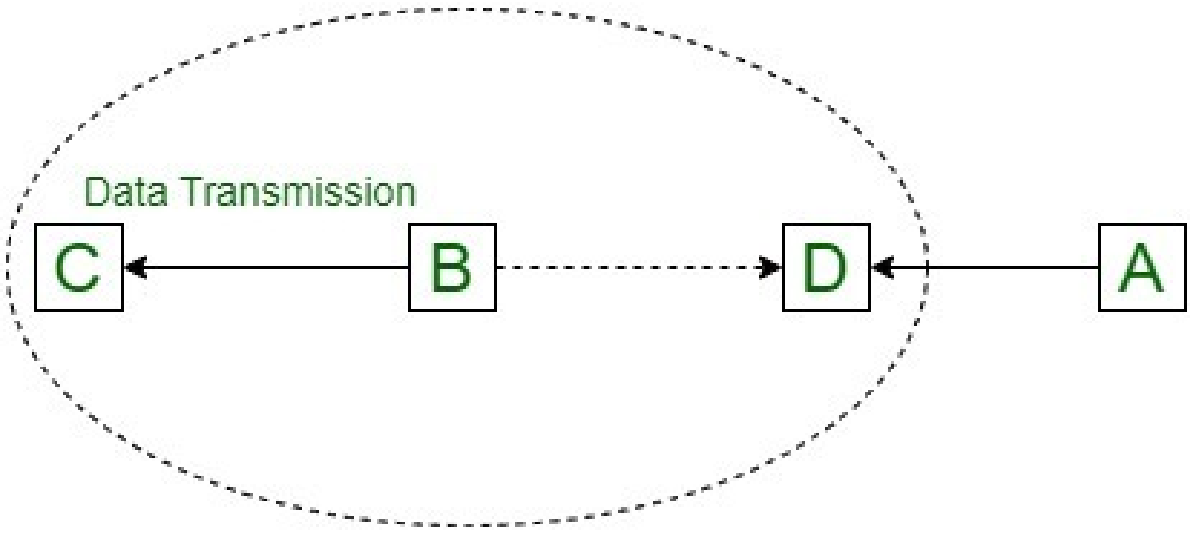


MACAW Algorithm

Multiple Access with Collision Avoidance for Wireless (MACAW) is a medium access control (MAC) protocol broadly utilized in ad hoc network systems. Besides, it is the establishment of numerous other MAC protocols utilized in wireless sensor systems (WSN).
The IEEE 802.11 RTS/CTS system is received from this protocol. It utilizes RTS-CTS-DS-DATA-ACK frame succession for moving information Although protocol dependent on MACAW, are S-MAC, MACAW doesn't utilize carrier sense.

Features :

- The problem in MACA that if there are two senders and two receivers A, B, C and D respectively.
- If B has sent RTS to C and D at the same time and but only send data upon receiving CTS from C.
- Now A wants to send data to D but will not able to send because it will sense that D is currently busy and will increase the backoff counter (for how much time A will wait before re-transmitting) value by twice because of which it will get stuck in a loop until the D gets free.
- Blockage data trade between pairwise stations, prompting better clog control and backoff approaches



Advantages over MACA :

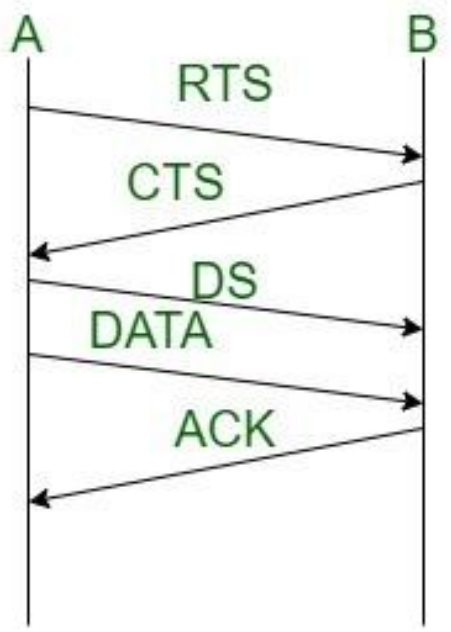
- The sender detects the bearer to see and transmits an RTS (Request To Send) if no close-by station transmits an RTS.
- The fairness of MACAW is much better than MACA.
- It handles hidden and exposed node problem better than MACA.
- ACK signal is sent to the MAC layer, after every data frame.
- It also incorporates carrier detecting to additionally diminish collision
- Irregular pause and re-attempt transmission at each message level, rather than at each node level.

Example :

A successful transmission in the case of MACAW will look like this:

- RTS from A to B
- CTS from B to A
- DS from A to B
- DATA frame from A to B
- ACK from B to A.

Example:



Code:

```
import Terminal

# send rts
def main():
    a = Terminal.Terminal("A", 2, 0, 2)
    b = Terminal.Terminal("B", 3, 0, 2)
    c = Terminal.Terminal("C", 0, 0, 2)
    d = Terminal.Terminal("D", 5, 0, 2)

    terminals = [c, a, b, d]
    Terminal.Terminal.algorithm = 'macaw'
    print(f'\t\t\t{Terminal.Terminal.algorithm.upper()} algorithm')


    # print terminals in coverage range of a
    for terminal in terminals:
        for t in terminals:
            if terminal != t:
                if terminal.in_coverage_range(t):
                    terminal.nodes_in_coverage_range.append(t)
```

```
    a.send_rts("B")
    print('------------------------------------------------------------------
------')
    print('------------------------------------------------------------------
------')
    print('------------------------------------------------------------------
------')


main()
```

Output:

```
                    MACAW algorithm
Terminal A sending rts to Terminal B
Terminal C received rts from A
Terminal B received rts from A
Terminal A received cts from B



Terminal A successfully recieved CTS from terminal B... sending data to B



Terminal A is sending data send request to Terminal B
Data to send : Hello, Bob!
Terminal B recieved data from Terminal A..
 Sending acknowledgement to Terminal A



Terminal A recieved acknowledgement from Terminal B
Data transfer successfully completed



Terminal D received cts from B
------------------------------------------------------------------
------------------------------------------------------------------
------------------------------------------------------------------
```