Computer Networks Lab Report – Assignment 4

TITLE

Name – Sourav Dutta

Roll – 001610501076

Class – BCSE 3rd year

Group – A3

Assignment Number – 4

Problem Statement – Implement CDMA with Walsh code.

In this assignment you have to implement CDMA for multiple access of a common channel by n stations. Each sender uses a unique code word, given by the Walsh set, to encode its data, send it across the channel, and then perfectly reconstruct the data at n stations.

Evaluation date – 18/03/2019
Submission date – 25/03/2019

DESIGN

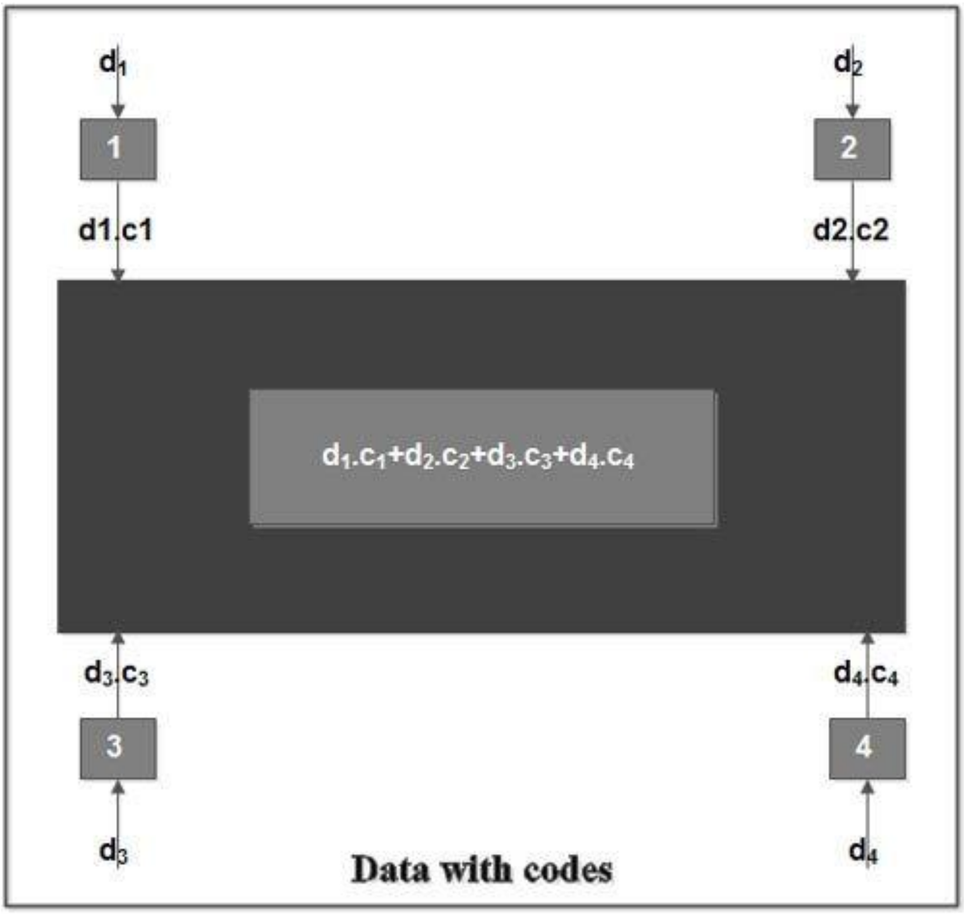
I have implemented the error detection module in three program files.

- **channel.py** (Program for channel)
- **station.py** (Program for a station process)

The individual files fulfils different assignment purposes, following which have been explained in details :

1. **channel.py** – The following are the tasks performed in this program :
    a. Asks for number of stations and initiates all stations.
    b. Creates a Walsh Table for the given number of stations (highest power of 2).
    c. Receives data bits from each station.
    d. Multiplies data bits with corresponding Walsh Sequence of each station, and summing them to get the final data.
    e. Asks for sender and receiver station number, from which sender to which receiver we want to send the data bit.
    f. Calculates the data bit by multiplying the final data with walsh sequence of sender station, summing it up; and then dividing the result by the number of stations.

2. **station.py** – The following are the tasks performed in this program :
    a. Sends the stream of data bits to the channel process.
    b. Let's say the maximum length of data bits sent by a station is X. If a station sends a stream having length less than X, then the rest of the bits are assumed to be silent.
    c. Receives a data bit from channel.



IMPLEMENTATION

Code Snippet of channel.py:

```
import socket
import time
import subprocess
import random
import os

def multiplyTuples(t1, t2):
	tup = []
	for i in range(len(t1)):
		tup.append(t1[i] * t2[i])
	return tup

def multiplyScalar(t1, x):
	tup = []
	for i in range(len(t1)):
		tup.append(t1[i] * x)
	return tup

def addTuples(t1, t2):
	tup = []
	for i in range(len(t1)):
		tup.append(t1[i]+t2[i])
```

```
        return tup

class Channel():
    def __init__(self, totalstations):
        self.totalstation = totalstations
        self.stationhost = '127.0.0.1'
        self.stationport = 8080
        self.stationconn = []
        self.walshtable = [ [] ]

    def initStations(self):
        stationSocket = socket.socket()
        stationSocket.bind((self.stationhost, self.stationport))
        stationSocket.listen(self.totalstation)
        for i in range(1, self.totalstation+1):
            conn = stationSocket.accept()
            self.stationconn.append(conn)
        print('Initiated all station connections')

    def closeStations(self):
        for conn in self.stationconn:
            conn[0].close()
        print('Closed all station connections')

    def generateWalshTable(self):
        print('Generating Walsh table ...')
        n = self.totalstation
        p = 1
        prevtable = [ [1] ]
        while p < n:
            p *= 2
            curtable = []
            for i in range(p):
                tup = []
                for j in range(p):
                    tup.append(0)
                curtable.append(tup)
            for i in range(0,p//2):
                for j in range(0,p//2):
                    curtable[i][j] = prevtable[i][j]
                    curtable[i+p//2][j] = prevtable[i][j]
                    curtable[i][j+p//2] = prevtable[i][j]
                    curtable[i+p//2][j+p//2] = -1*prevtable[i][j]
            prevtable = curtable

        self.walshtable = prevtable
        print('Printing Walsh table :')
        for i in range(p):
            for j in range(p):
                if self.walshtable[i][j] == 1:
                    print(end=' ')
                print(self.walshtable[i][j],end=' ')
            print()
```

```
def process(self):
        '''data = []
        data.append('1001')
        data.append('011')
        data.append('11011')
        data.append('0000')
        for i in range(self.totalstation):
                print('Station',i+1,'will send data:',end=' ')
                print(data[i])
        '''
        data = []
        for i in range(self.totalstation):
                conn = self.stationconn[i]
                d = conn[0].recv(1024).decode()
                data.append(d)

        for i in range(self.totalstation):
                print('Station',i+1,'will send data:',end=' ')
                print(data[i])
        maxlen = 0
        for i in data:
                maxlen = max(maxlen,len(i))
        datavalue = []
        for d in data:
                tup = []
                for j in range(maxlen):
                        if j < len(d):
                                if d[j] == '0':
                                        tup.append(-1)
                                elif d[j] == '1':
                                        tup.append(1)
                        else:
                                tup.append(0)
                datavalue.append(tup)

        for i in range(maxlen):
                print('---------------------------------------------')
                print('Sending bit',i+1,'of each station\'s data')
                finaldata = []
                d = []
                c = []
                n = len(self.walshtable)
                for j in range(n):
                        if j < self.totalstation:
                                d.append(datavalue[j][i])
                        else:
                                d.append(1)
                        c.append(self.walshtable[j])
                        finaldata.append(0)

                for j in range(n):
                        temp = multiplyScalar(c[j], d[j])
```

```
                    finaldata = addTuples(finaldata,temp)
            print('Bit',i+1,'of each station is:',end=' ')
            print(d)
            print()
            print('After multiplying data bit with code bits of corresponding stations, and adding
them all,')
            print('Final data is:',end=' ')
            print(finaldata)
            print()

            choice = input('Does any station want to receive data ? (y/n) ')
            while choice == 'y':
                    stnum,renum = input('Enter the sender and receiver station number:
').split()
                    stnum = int(stnum)
                    renum = int(renum)
                    if stnum > self.totalstation or stnum <= 0 or renum > self.totalstation or
renum <= 0:
                            print('Invalid station number')
                    else:
                            temp = multiplyTuples(finaldata, c[stnum-1])
                            print('Multiplying final data with Code bits of sender station',stnum)
                            print('The result is :',end=' ')
                            print(temp)
                            summ = sum(temp)
                            print('the sum of result is:',summ)
                            databit = str(summ//n)
                            print('THE DATA BIT OF STATION',stnum,'is:',databit)
                            conn = self.stationconn[renum-1]
                            conn[0].sendto(databit.encode(), conn[1])
                            print('Data bit sent to receiver',renum,'successfully!')
                            print()
                            choice = input('Does any station want to receive data ? (y/n) ')


if __name__ == '__main__':
        totalstations = int(input('Enter number of stations: '))

        ch = Channel(totalstations)
        ch.generateWalshTable()
        ch.initStations()
        ch.process()
        ch.closeStations()
```

Code Snippet of station.py:

```
import socket
import sys
import time
import random

def Main(senderno):
        print('Initiating Station #',senderno)
```

```
    host = '127.0.0.1'
    port = 8080

    mySocket = socket.socket()
    mySocket.connect((host, port))

    data = input('Enter $ ')
    mySocket.send(data.encode())

    while True:
        data = mySocket.recv(1024).decode()
        if not data:
            break
        print('Received bit value from channel:',str(data))
        data = int(data)
        if data == -1:
            val = 0
        elif data == 1:
            val = 1
        else:
            val = "silent"
        print('VALUE OF RECEIVED BIT IS :',str(val))

    mySocket.close()

if __name__ == '__main__':
    if len(sys.argv) > 1:
        senderno = int(sys.argv[1])
    else:
        senderno = 1
    Main(senderno)
```

TEST CASES

Channel.py :

```
C:\Users\SOURAV\Desktop\comp-networks-lab\ass4>python channel.py
Enter number of stations: 4
Generating Walsh table ...
Printing Walsh table :
 1  1  1  1
 1 -1  1 -1
 1  1 -1 -1
 1 -1 -1  1
Initiated all station connections
Station 1 will send data: 01
Station 2 will send data: 1
Station 3 will send data: 10
Station 4 will send data: 1
---------------------------------------------
Sending bit 1 of each station's data
Bit 1 of each station is: [-1, 1, 1, 1]

After multiplying data bit with code bits of corresponding stations, and adding them all,
Final data is: [2, -2, -2, -2]
```

Does any station want to receive data ? (y/n) y
Enter the sender and receiver station number: 4 2
Multiplying final data with Code bits of sender station 4
The result is : [2, 2, 2, -2]
the sum of result is: 4
THE DATA BIT OF STATION 4 is: 1
Data bit sent to receiver 2 successfully!

Does any station want to receive data ? (y/n) y
Enter the sender and receiver station number: 3 1
Multiplying final data with Code bits of sender station 3
The result is : [2, -2, 2, 2]
the sum of result is: 4
THE DATA BIT OF STATION 3 is: 1
Data bit sent to receiver 1 successfully!

Does any station want to receive data ? (y/n) y
Enter the sender and receiver station number: 2 3
Multiplying final data with Code bits of sender station 2
The result is : [2, 2, -2, 2]
the sum of result is: 4
THE DATA BIT OF STATION 2 is: 1
Data bit sent to receiver 3 successfully!

Does any station want to receive data ? (y/n) y
Enter the sender and receiver station number: 1 4
Multiplying final data with Code bits of sender station 1
The result is : [2, -2, -2, -2]
the sum of result is: -4
THE DATA BIT OF STATION 1 is: -1
Data bit sent to receiver 4 successfully!

Does any station want to receive data ? (y/n) n

Sending bit 2 of each station's data
Bit 2 of each station is: [1, 0, -1, 0]

After multiplying data bit with code bits of corresponding stations, and adding them all,
Final data is: [0, 0, 2, 2]

Does any station want to receive data ? (y/n) y
Enter the sender and receiver station number: 2 3
Multiplying final data with Code bits of sender station 2
The result is : [0, 0, 2, -2]
the sum of result is: 0
THE DATA BIT OF STATION 2 is: 0
Data bit sent to receiver 3 successfully!

Does any station want to receive data ? (y/n) n
Closed all station connections

Station.py (1st station):

C:\Users\SOURAV\Desktop\comp-networks-lab\ass4>python station.py 1
Initiating Station # 1
Enter $ 01
Received bit value from channel: 1
VALUE OF RECEIVED BIT IS : 1

Station.py (2nd station):

C:\Users\SOURAV\Desktop\comp-networks-lab\ass4>python station.py 2
Initiating Station # 2
Enter $ 1
Received bit value from channel: 1
VALUE OF RECEIVED BIT IS : 1

Station.py (3rd station):

C:\Users\SOURAV\Desktop\comp-networks-lab\ass4>python station.py 3
Initiating Station # 3
Enter $ 10
Received bit value from channel: 1
VALUE OF RECEIVED BIT IS : 1
Received bit value from channel: 0
VALUE OF RECEIVED BIT IS : silent

Station.py (4th station):

C:\Users\SOURAV\Desktop\comp-networks-lab\ass4>python station.py 4
Initiating Station # 4
Enter $ 1
Received bit value from channel: -1
VALUE OF RECEIVED BIT IS : 0

RESULTS & ANALYSIS

• Unlike TDMA, in CDMA all stations can transmit data simultaneously, there is no timesharing.
• CDMA allows each station to transmit over the entire frequency spectrum all the time.
• Multiple simultaneous transmissions are separated using coding theory.
• In CDMA each user is given a unique code sequence.
• The basic idea of CDMA is explained below:

1. Let us assume that we have four stations 1, 2, 3 and 4 that are connected to same channel. The data from station 1 are dl, from station 2 are d2 and so on.
2. The code assigned to first station is C_1, to the second is C_2 and so on.

3. These assigned codes have two properties:

(a) If we multiply each code by another, we get O.

(b) If we multiply each code by itself, we get 4. (No. of stations).

4. When these four stations are sending data on the same channel, station 1 multiplies its data by its code *i.e.* $d_l.c_1$}, station 2 multiplies its data by its code *i.e.* d_2 .C_2 and so on.

5. The data that go on channel are the sum of all these terms as shown in Fig.

6. Any station that wants to receive data from one of the other three stations multiplies the data on channel by the code of the sender. For example, suppose station 1 and 2 are talking to each other. Station 2 wants to hear what station 1 is saying. It multiples the data on the channel by CI (the code of station 1).

7. Because (C_1. C_1) is 4, but (C_2. C_1), (C_3. C_1), and (C_4. C_1) are all zeroes, station 2 divides the result by 4 to get the data from station 1.

$$\mathbf{data = (d_l . C_1 + d_2 . C_2 + d_3 . C_3 + d_4 . C_4) \bullet C_1}$$

$$\mathbf{= d_l . C_1 . C_1 + d_2 . C_2 . C_1 + d_3 . C_3 . C_1 + d_4 . C_4 . C_1 = 4 \times d_l}$$

• The code assigned to each station is a sequence of numbers called chips. These chips are called orthogonal sequences. This sequence has following properties:

1. Each sequence is made of N elements, where N is the number of stations as shown in fig.

2. If we multiple a sequence by a number, every element in the sequence is multiplied by that element. This is called multiplication of a sequence by a scalar.

For example:

[+1 +1-1 -1] = [+2 +2 -2 -2]

3. If we multiply two equal sequences, element by element and add the results, we get N, where N is the number of elements in each sequence. This is called inner product of two equal sequences. For example:

[+1 +1-1 -1] . [+1 +1-1-1] = 1+ 1+ 1+ 1 = 4

4. If we multiply two different sequences, element by element and add the results, we get 0. This is called inner product of two different sequences. For example:

[+1 +1-1-1]. [+1 +1 +1 +1] = 1+1-1 -1= 0

5. Adding two sequences means adding the corresponding elements. The result is another sequence. For example:

[+1 +1-1 -1] + [+1 +1+1 +1] = [+2 +2 0 0]

• The data representation and encoding is done by different stations in following manner:

1. If a station needs to send a 0 bit, it encodes it as -1.

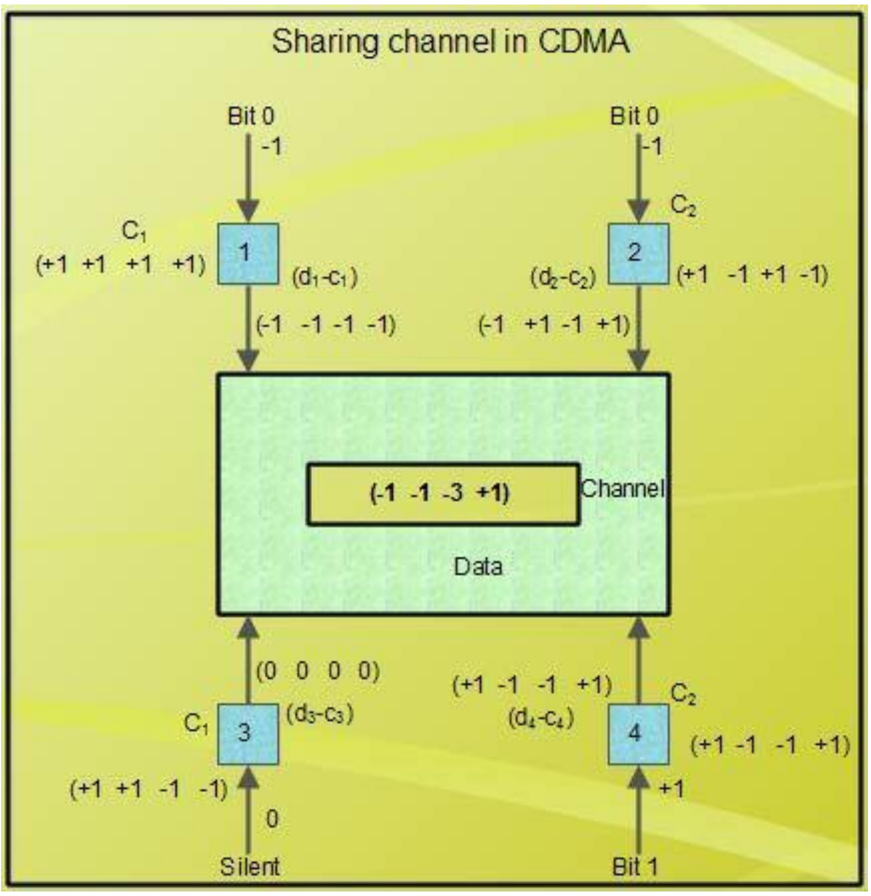
2. If it needs to send a 1 bit, it encodes it as + 1.

3. When station is idle, it sends no signal, which is interpreted as a 0.

• For example, If station 1 and station 2 are sending a 0 bit, station 3 is silent and station 4 is sending a 1 bit; the data at sender site are represented as -1, - 1,0 and +1 respectively.

• Each station multiplies the corresponding number by its chip, which is unique for each station.

• Each station send this sequence to the channel ; The sequence of channel is the sum of all four sequence as shown in fig.



If station 3, which was silent, is listening to station 2. Station 3 multiplies the total data on the channel by the code for station 2, which is [+ 1 -1 +1 -1], to get

[-1 -1 -3+1] . [+1 -1 +1 -1]= -4/4 = -1 --> bit 0

COMMENTS

This assignment has helped me to understand the how Walsh Table is built for a given number of stations, and how CDMA channelization protocol encodes and decodes the data bits sent by all stations simultaneously.