Computer Networks Lab Report - Assignment 4

Name: Bikash Sah

Class: BCSE-3

Group: A1

Assignment Number: 4

Problem Statement:

Assignment 4: Implement CDMA with Walsh code.

Submission due: 19th-23th September 2022

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.

Submission Date: **21 November, 2022**

Deadline: 23rd Sept, 2022

Introduction

There are n number of servers. Each server has a data bit to be sent to the reciever. The data bits are sent through a channel. The channel is lossy. The channel can drop the data bits. The channel can also add noise to the data bits. The reciever has to decode the data bits from the servers. The reciever has to detect the dropped data bits and the noise added to the data bits. The reciever has to correct the data bits. The reciever has to display the data bits from the servers.

Design

```
The program should have the following classes:
1. sender
2. Channel
3. receiver

The sender class should have the following functions:

1. sender(int n, vector<vector<int> > arr) - constructor to initialize the number of s
ervers and the walsh table for the servers.

2. vector<int> getData() - to get the data bits from the servers. The data bits are ra
ndomly generated.

3. vector<int> encodeData() - to encode the data bits from the servers using the walsh
table. The encoded data bits are sent to the channel.

The channel class should have the following functions:
1. Channel(sender *s) - constructor to initialize the sender object.

2. vector<int> getFromSender() - to get the encoded data bits from the sender.

3. vector<int> sendToreciver() - to send the encoded data bits to the receiver.

The receiver class should have the following functions:
1. receiver(int n, vector<vector<int> > arr, Channel *ch) - constructor to initialize
 the number of servers, the walsh table for the servers and the channel object.

2. vector<int> reciveData() - to recieve the encoded data bits from the channel.

3. vector<int> decodeData() - to decode the encoded data bits using the walsh table. T
he decoded data bits are sent to the display function.

 The main function should have the following:
 1. The main function should create the sender, channel and receiver objects.
 2. The main function should call the decodeData() function of the receiver object.
 3. The main function should display the data bits from the servers.
```

Implementation

```
#include <bits/stdc++.h>
using namespace std;

class sender
{
    int senders;
    int size;
    int startIndex;
    vector<vector<int> > walshTable;
    vector<vector<int> > data;

public:
    sender(int n, vector<vector<int> > arr)
    {
        this->startIndex = 0;
        this->senders = n;
        this->walshTable = arr;
        this->size = rand() % 20;

        //GENERATING RANDOM DATA FOR EVERY SENDER
        cout<<endl;
        cout<<"____Data table_______"<<endl;
        for (int i = 0; i < senders; i++)
        {
            cout<<"Data for sender "<<i<<":   ";
            vector<int> temp;
            for (int j = 0; j < size; j++)
            {
                int bit = rand() % 3;
                temp.push_back(bit);
                if(bit==2)
                {
                    cout << "*" <<"  ";
                }
                else
                {cout << bit <<"  ";}
            }
            cout << endl;
            data.push_back(temp);
        }
        cout<<"_________________"<<endl;
    }

    vector<int> getData()
    {
        if (startIndex == size)
        {
            cout << "no more data";
        }
        vector<int> res;
        for (int i = 0; i < senders; i++)
        {
            res.push_back(data[i][startIndex]);
        }
        startIndex++;
        cout << endl
```

```
             << endl
             << endl;
        cout<<"_____sender side______"<<endl;
        for (int i = 0; i < res.size(); i++)
        {
            cout <<"Data bit from Sender "<<i<<":  ";
            if(res[i]==2)
            {
                cout<<"*"<<endl;
            }
            else
             {cout<<res[i]<<endl;}
        }
        cout<<"_________________"<<endl;
        return res;
    }
    vector<int> encodeData()
    {
        vector<int> res(walshTable.size(), 0);
        vector<int> rawData = getData();
        for (int i = 0; i < rawData.size(); i++)
        {
            if (rawData[i] == 0)
                for (int k = 0; k < res.size(); k++)
                {

                    res[k] += (-1 * walshTable[i][k]);
                }
            else if (rawData[i] == 1)
                for (int k = 0; k < res.size(); k++)
                {

                    res[k] += (1 * walshTable[i][k]);
                }
            else
                for (int k = 0; k < res.size(); k++)
                {

                    res[k] += (0 * walshTable[i][k]);
                }
        }
        cout << endl;
        cout<<"____DATA SENT______"<<endl;
        for (int i = 0; i < res.size(); i++)
        {
            cout << res[i] << "    ";
        }
        cout<<endl;
        cout<<"_____________"<<endl;
        cout<<endl;
        return res;
    }
};

class Channel
{
public:
    sender *st;
```

```
    Channel(sender *s)
    {
        st = s;
    }
    vector<int> getFromSender()
    {
        return st->encodeData();
    }
    vector<int> sendToreciver()
    {
        return this->getFromSender();
    }
};

class reciver
{
    int stations;
    vector<vector<int> > walshTable;
    Channel *ch;

public:
    reciver(int s, vector<vector<int> > arr, Channel *ch)
    {
        this->stations = s;
        this->walshTable = arr;
        this->ch = ch;
    }
    vector<int> reciveData()
    {
        return ch->sendToreciver();
    }
    vector<int> decodeData()
    {
        vector<int> data = reciveData();
        return data;
    }
};
vector<vector<int> > fun(int n)
{
    if (n == 1)
    {
        vector<vector<int> > a(1, vector<int>(1, 1));
        return a;
    }

    vector<vector<int> > arr(n, vector<int>(n));
    vector<vector<int> > temp = fun(n / 2);
    for (int i = 0; i < n / 2; i++)
    {
        for (int j = 0; j < n / 2; j++)
        {
            arr[i][j] = temp[i][j];
            arr[i + (n / 2)][j] = temp[i][j];
            arr[i][j + (n / 2)] = temp[i][j];
            arr[i + n / 2][j + (n / 2)] = -temp[i][j];
        }
    }
    return arr;
```

```
}

int main()
{
    srand(time(0));

    int n;
    cout<<"enter the number of servers: ";
    cin >> n;
    int ser = n;
    n = pow(2, ceil(log2(n)));//maximum size of walash table required

    vector<vector<int> > arr = fun(n);
    cout<<"_____WALASH TABLE_______"<<endl;
    for (int i = 0; i < n; i++)
    {
        for (int j = 0; j < n; j++)
        {
            if(arr[i][j]==1)
            {
                cout<<" "<<arr[i][j]<<"   ";
            }
            else{
            cout << arr[i][j] << "   ";}
        }
        cout << endl;
    }
    cout<<"___________________"<<endl;

    sender st(ser, arr);
    Channel ch(&st);
    reciver rec(ser, arr, &ch);
    int flag=1;
    while (flag == 1)
    {
        vector<int> temp = rec.decodeData();
        cout<<"_____RECIVER SIDE__________"<<endl;
        for(int i=0;i<ser;i++)
        {
            int sum=0;
            for(int k=0;k<arr.size();k++)
            {
                sum+=(temp[k]*arr[i][k]);
            }
            int bit = sum / n;
            if (bit == 0)
            {
                   cout << "Data bit from server " << i << ": "<< " *"<<endl;
            }
            else if (bit == 1)
            {
                {
                    cout << "Data bit from server " << i << ": "<< " 1"<<endl;
                }
            }
            else
            {
                {
```

```
                    cout << "Data bit from server " << i << " "<< "0"<<endl;
                }
            }
        }
        cout<<"_____________________"<<endl;
        cout << "enter y to continue" << endl;
        cin >> flag;
    }
}
```

Test Cases and Results

```
thebikashsah@BIKASHs-MacBook-Air Network_4 % ./a.out
enter the number of servers: 2
_____WALASH TABLE_______
 1    1
 1   -1
_________________

____Data table_______
Data for sender 0:   0  1  *  *  0  0  1  *  1  1  *  0  *  1  0  *  *  *
Data for sender 1:   0  0  1  1  0  *  0  0  *  0  0  1  *  1  1  *  0  0
_________________



_____sender side______
Data bit from Sender 0:  0
Data bit from Sender 1:  0
_______________

____DATA SENT______
-2    0
_____________

_____RECIVER SIDE__________
Data bit from server 0 0
Data bit from server 1 0
____________________
enter y to continue
```

Discussion

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.