**Computer Networks Lab Report – Assignment 3**

**TITLE**

**Name – Aditya Ganguly**

**Roll – 002010501019**

**Class – BCSE 3rd year**

**Group – A1**

**Number – 4**

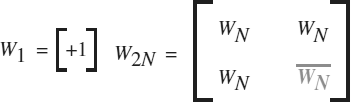
**Problem Statement: In this assignment you have to implement CDMA for multiple access of a common channel by n stations. Each sender uses a unique code word, given by the Walsh set, to encode its data, send it across the channel, and then perfectly reconstruct the data at n stations.**

**DESIGN**

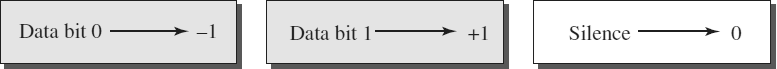
CDMA is based on coding theory. Each station is assigned a code, which is a sequence of numbers called chips. They are called orthogonal sequences and have the following properties:

• Product of any two different chips is 0

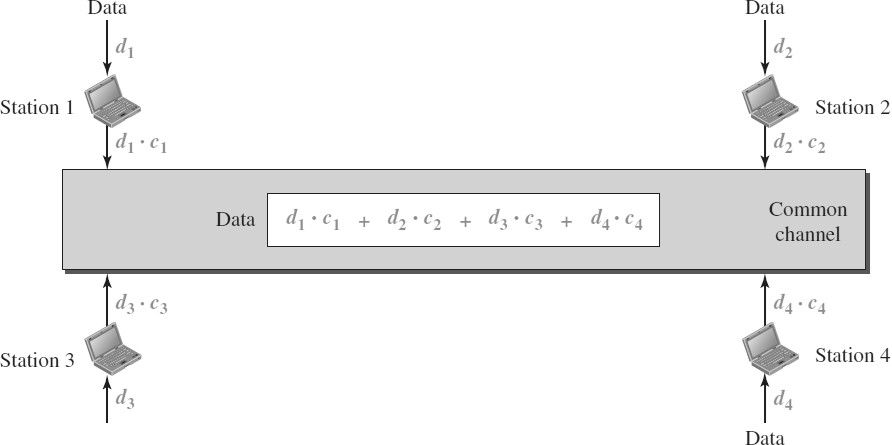
• A chip multiplied with itself gives the result N (= total number of stations) The chips are generated using Walsh Table.



The number of sequences in a Walsh Table needs to be a power of 2. The encoding of bits are as follows:



The data on the channel is present in the form:



To listen to any particular station, the station multiplies the data with the chip sequence of that station.

### IMPLEMENTATION

Generation of the Walsh Table

We first get the next greater power of 2, which will be equal to the dimension of the Walsh Table. Then, we build the Walsh Table recursively using the recursive formula given above.

printf("Enter number of senders and receivers\n");

int n;

scanf("%d", &n);

char \*NAME = (char \*)malloc(100 \* sizeof(int));

for (int i = 0; i < n; i++)

{

itoa(i, NAME, 10);

sem\_unlink(NAME);

}

int \*\*\*table;

table = (int \*\*\*)malloc(MAXPOW \* sizeof(int \*\*));

for (int i = 0; i < MAXPOW; i++)

{

table[i] = (int \*\*)malloc((1 << i) \* sizeof(int \*));

for (int j = 0; j < (1 << i); j++)

table[i][j] = (int \*)malloc((1 << i) \* sizeof(int));

}

table[0][0][0] = 1;

for (int i = 1; i < MAXPOW; i++)

{

for (int j = 0; j < (1 << i); j++)

{

for (int k = 0; k < (1 << i); k++)

{

int pow = (1 << (i - 1));

if (j >= pow && k >= pow)

table[i][j][k] = -table[i - 1][j - pow][k - pow];

else if (j >= pow)

table[i][j][k] = table[i - 1][j - pow][k];

else if (k >= pow)

table[i][j][k] = table[i - 1][j][k - pow];

else

table[i][j][k] = table[i - 1][j][k];

}

}

}

Encoding and Decoding is done in a multiprocessing environment where there is a Sender process, a Receiver Process and a Channel Process. Synchronization is done using Semaphores.

sem\_t \*semaphores[n];

for (int i = 0; i < n; i++)

{

itoa(i, NAME, 10);

if (i != 0)

semaphores[i] = sem\_open(NAME, O\_CREAT | O\_EXCL, 0666, 0);

else

semaphores[i] = sem\_open(NAME, O\_CREAT | O\_EXCL, 0666, 1);

}

int node\_to\_channel[n][2];

int channel\_to\_node[n][2];

for (int i = 0; i < n; i++)

{

pipe(node\_to\_channel[i]);

pipe(channel\_to\_node[i]);

}

int i = 0;

for (i = 0; i < n; i++)

{

int id = fork();

if (id == 0)

break;

else

{

printf("Initializing sender %d\n", i + 1);

}

}

if (i == n)

{

sleep(1);

i = 0;

for (i = 0; i < n; i++)

{

int id = fork();

if (id == 0)

break;

else

{

printf("Initializing receiver %d\n", i + 1);

}

}

if (i == n)

{

char \*temp = (char \*)malloc(sizeof(char) \* FRAMESIZE);

int \*data = (int \*)malloc(sizeof(int) \* FRAMESIZE \* MAXSEND);

int cnt = 0;

memset(data, 0, sizeof(int) \* FRAMESIZE \* MAXSEND);

for (int j = 0; j < n; j++)

{

close(node\_to\_channel[j][1]);

close(channel\_to\_node[j][0]);

}

while (1)

{

readmsg(node\_to\_channel[cnt], temp);

for (int j = 0; j < MAXSEND \* FRAMESIZE; j++)

{

data[j] += (temp[j / MAXSEND] \* table[4][cnt][j % MAXSEND]);

}

cnt = (cnt + 1) % n;

if (cnt == 0)

{

printf("\nTransmitting\n");

sleep(2);

for (int j = 0; j < n; j++)

{

write(channel\_to\_node[j][1], data, sizeof(int) \* FRAMESIZE \* MAXSEND);

}

memset(data, 0, sizeof(int) \* FRAMESIZE \* MAXSEND);

}

}

for (int j = 0; j < n; j++)

{

close(node\_to\_channel[j][0]);

close(channel\_to\_node[j][1]);

}

}

else

{ // Receiver here

close(channel\_to\_node[i][1]);

int \*data = (int \*)malloc(sizeof(int) \* FRAMESIZE \* MAXSEND);

char \*msg = (char \*)malloc(FRAMESIZE \* sizeof(char));

int \*frame = (int \*)malloc(sizeof(int) \* FRAMESIZE);

while (1)

{

memset(frame, 0, sizeof(int) \* FRAMESIZE);

read(channel\_to\_node[i][0], data, sizeof(int) \* FRAMESIZE \* MAXSEND);

for (int j = 0; j < FRAMESIZE; j++)

{

for (int k = 0; k < MAXSEND; k++)

frame[j] += (data[MAXSEND \* j + k] \* table[4][i][k]);

}

for (int j = 0; j < FRAMESIZE; j++)

msg[j] = frame[j] / MAXSEND;

unpad(msg);

printf("Receiver %d received message %s\n", i + 1, msg);

}

close(channel\_to\_node[i][0]);

}

}

else

{

close(node\_to\_channel[i][0]);

char \*buffer = (char \*)malloc(FRAMESIZE \* sizeof(char));

while (1)

{

sleep(3);

sem\_wait(semaphores[i]);

printf("\nFor node %d enter message\n", i + 1);

scanf("%s", buffer);

pad(buffer);

writemsg(node\_to\_channel[i], buffer);

// sem\_post(channel);

sem\_post(semaphores[(i + 1) % n]);

}

}

return 0;

for (int i = 0; i < n; i++)

{

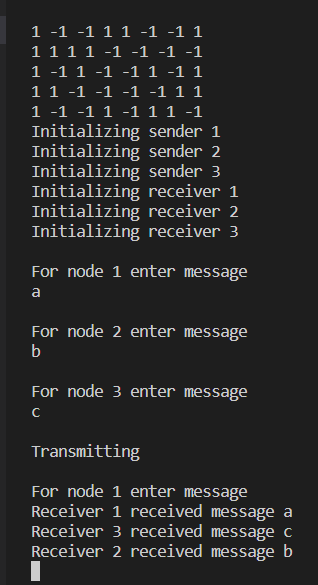
itoa(i, NAME, 10);

sem\_unlink(NAME);

}

sem\_unlink("/channel");

}



**DISCUSSION**

I could learn more about the CDMA protocol and its implementation with the help of this assignment.