**Computer Networks Laboratory**

**Assignment 4**

Name: Anirban Das Class: BCSE-III  Roll: 001910501077  Group: A3

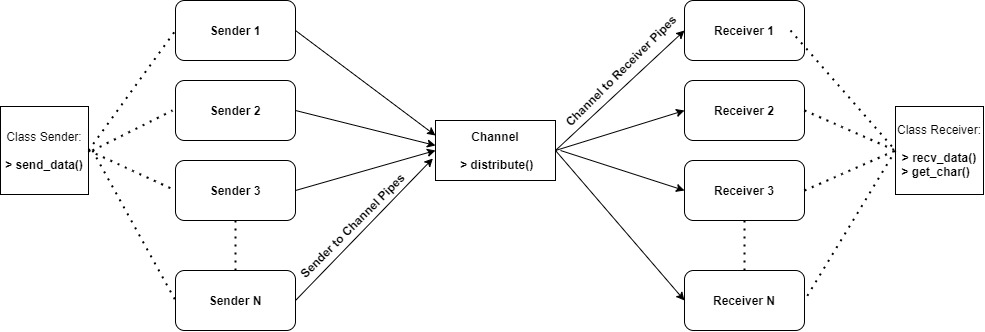
**Problem Statement:**

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.

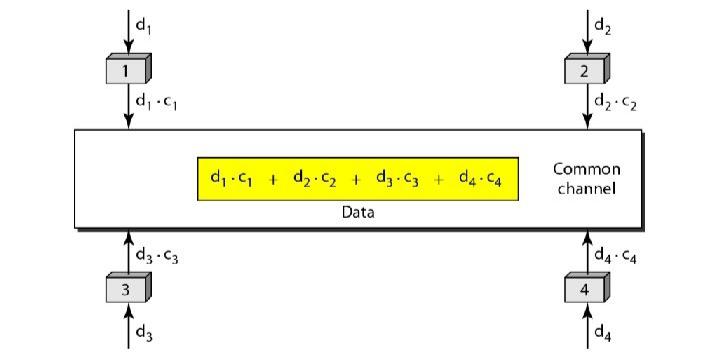
**Design:**

CDMA has been implemented in this assignment. In the main.py file, walsh table is built for encoding data. Pipes have been created for transfer between sender and channel and receiver. Data is transferred from senders to channel using one pipe. For transfer from channel to receiver, a pipe has been created for each receiver. In main, user has to give input of the number of bytes that have to be transferred from sender to receiver. In each sender, those many characters have been randomly generated and sent to the channel. The channel adds data received from each sender and then sends the same data to all receivers. The receiver receives the data from the channel and extracts data sent to it from the corresponding sender. When it has received the full byte it writes the byte to the corresponding output file. After the senders have completed sending all the bytes, the bytes sent, delay and throughput are written to a file. The operations occurring like sending data, receiving data are written to a file continuously. Below is a diagram showing the design.



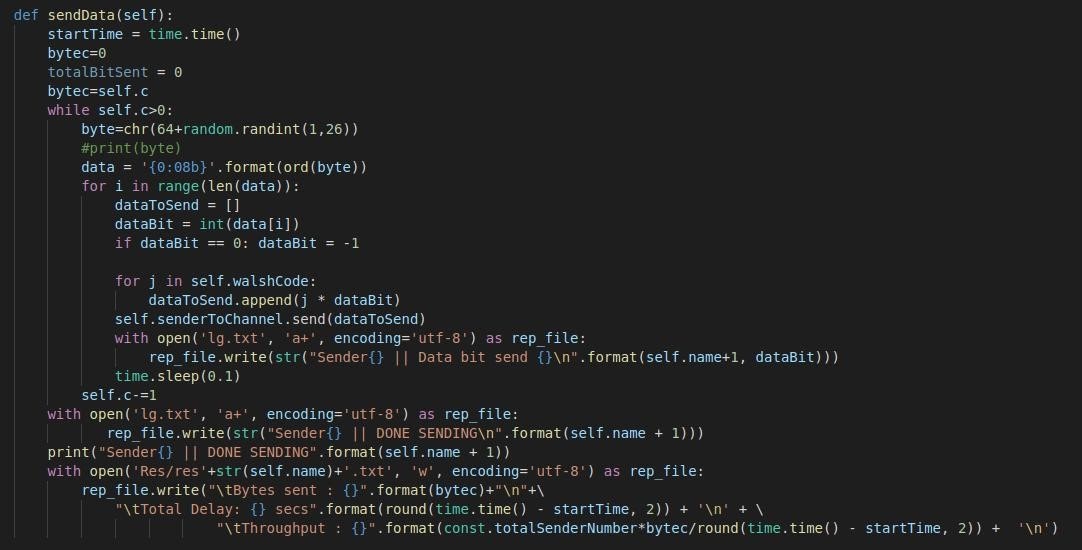
CDMA Design Implementation

CDMA stands for Code Division Multiple Access. It is basically a channel access method and is also an example of multiple access. There are multiple users which are provided or assigned variant CDMA codes and thus the users can access the entire band of frequencies or the whole bandwidth. This method does not limit the frequency range of the user. Hence, with the help of CDMA, multiple users can share a band of frequencies without any kind of undue interference between them. A full spectrum is used by all the channels in CDMA. There is no particular limit to the number of users in a CDMA system but with increase in the number of users the performance degrades.

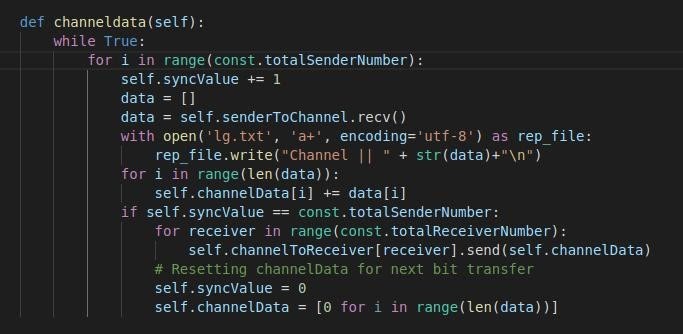


Sharing Channel in CSMA

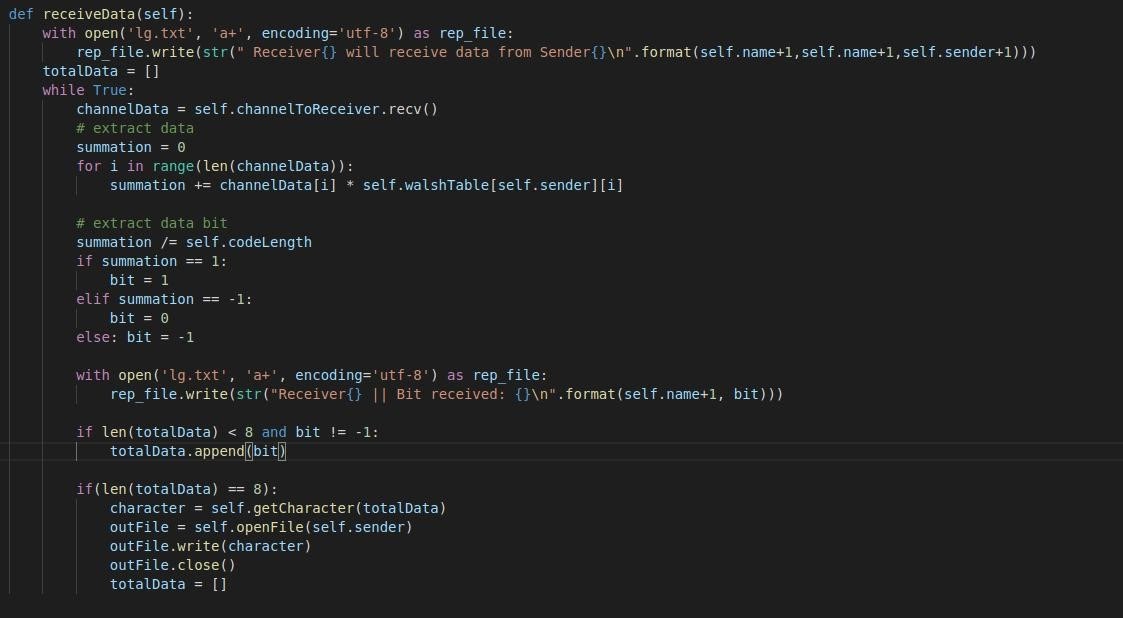
**Code Snippets:**



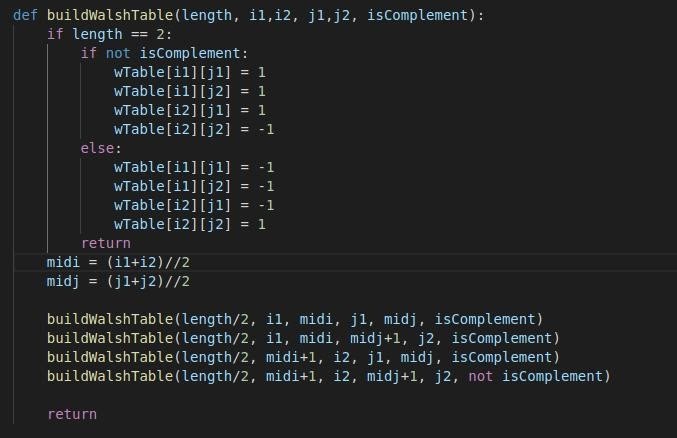
The sending of data takes place in sender.py using the following sendData function



The channeldata function in channel.py adds data from all senders and sends to receivers



The receiveData function in receiver.py accepts data from channel and extracts data bit and when it has receiver full byte it gets the character that was sent to it by sender and writes it to corresponding output file.



Function to build Walsh table in main.py

**Results:**

An observation table for throughputs of various “input size” and different “total senders” in the system.

TABLE 1

|  |  |  |  |
| --- | --- | --- | --- |
| **No. Of Senders** | **String Size** (in bytes) | **Delay** (in s) | **Throughput** (in bytes/s) |
| 2 | 5 | 5.030 | 1.992 |
| 10 | 10.310 | 1.932 |
| 20 | 20.535 | 1.957 |
| 30 | 30.955 | 1.938 |
| 50 | 51.715 | 1.934 |
| 4 | 5 | 4.960 | 4.032 |
| 10 | 10.285 | 3.890 |
| 20 | 20.372 | 3.927 |
| 30 | 30.782 | 4.092 |
| 50 | 50.045 | 4.027 |
| 8 | 5 | 4.765 | 8.394 |
| 10 | 9.625 | 8.311 |
| 20 | 19.201 | 8.339 |
| 30 | 29.797 | 8.054 |
| 50 | 47.843 | 8.536 |
| 16 | 5 | 4.514 | 16.778 |
| 10 | 9.545 | 16.763 |
| 20 | 18.786 | 17.035 |
| 30 | 27.910 | 17.198 |
| 50 | 46.592 | 17.171 |
| 32 | 5 | 4.855 | 32.961 |
| 10 | 9.599 | 33.339 |
| 20 | 31.308 | 20.443 |
| 30 | 54.552 | 17.599 |
| 50 | 110.285 | 14.508 |
| 64 | 5 | 5.659 | 56.786 |
| 10 | 34.639 | 18.478 |
| 20 | 90.964 | 14.072 |
| 30 | 130.447 | 14.719 |
| 50 | 236.521 | 13.53 |

TABLE 2

|  |  |
| --- | --- |
| **NO. OF SENDERS** | **AVERAGE THROUGHPUT** |
| 2 | 1.950 |
| 4 | 3.993 |
| 8 | 8.327 |
| 16 | 16.988 |
| **32** | **23.770** |
| **64** | **23.517** |

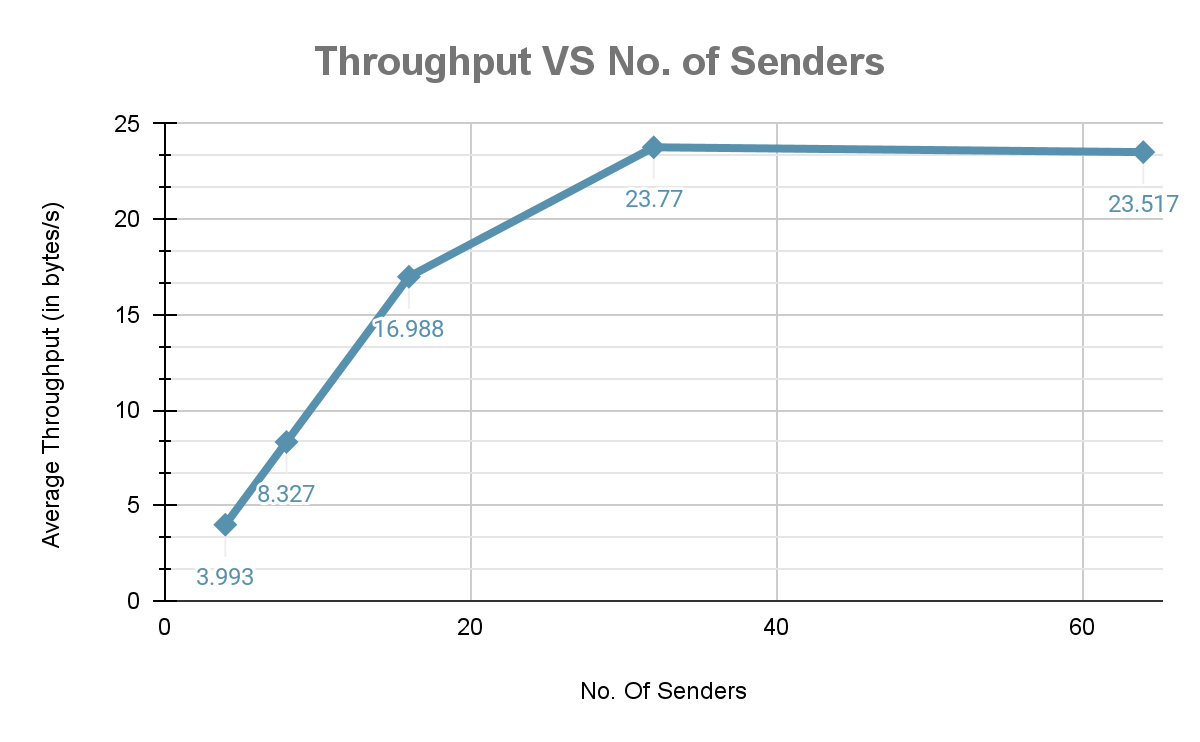
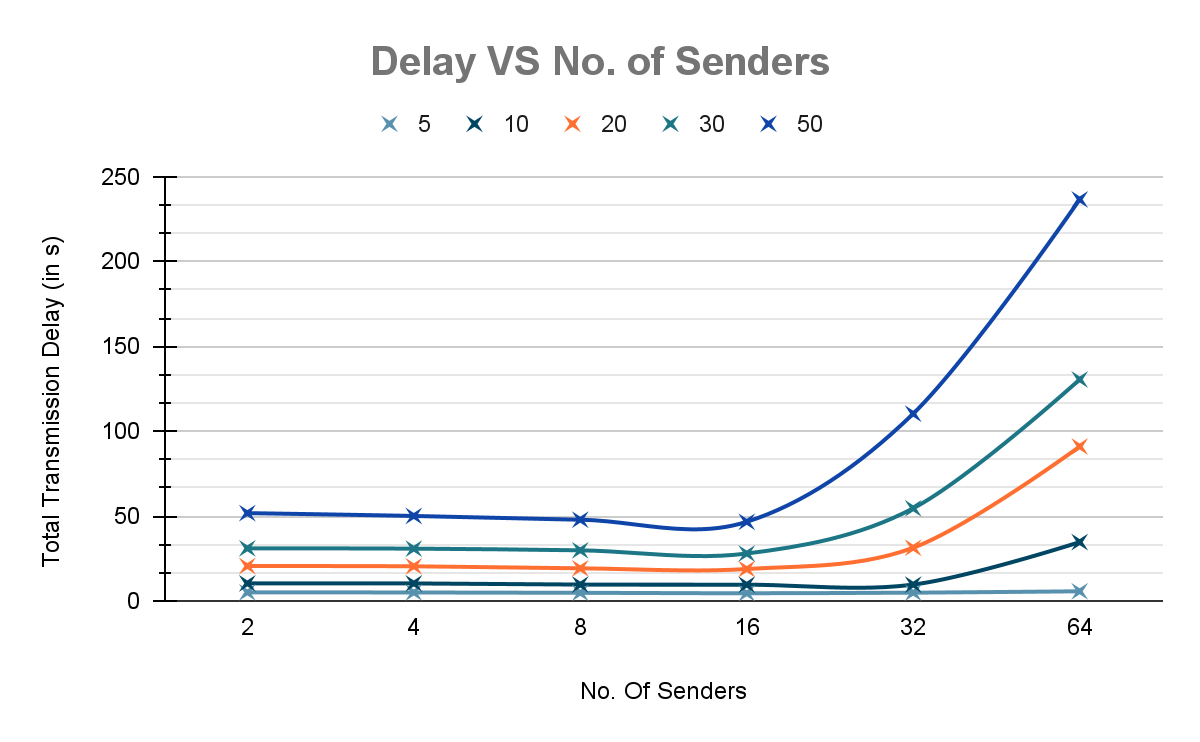


TABLE 3

|  |  |
| --- | --- |
| **NO. OF**  **SENDERS** | **RANGE (MAX-MIN) OF**  **THROUGHPUT** |
| 2 | 0.06 |
| 4 | 0.20 |
| 8 | 0.48 |
| 16 | 0.44 |
| **32** | **18.331** |
| **64** | **43.256** |

The unusual increase in the range value for 32 & 64 senders shows the delay produced while operating with walsh table of larger size.



This chart depicts the delay in transmission for different data-packet size and number of senders.

**Analysis:**

The throughput increases as we increase the number of senders as depicted in TABLE 2 and CHART 1. However, this linear increase is evident only for a fixed amount of senders, 16 in this case, after which there is a slight increase for 32 senders and a decrease for 64 senders.

The same phenomenon is observed for Total Delay VS Senders graph. The Delay is constant for all number of packets till 16 senders, after which there is a steep rise in the graph showing a huge time consumption for large number of senders (32 & 64).

This is mainly caused due to the increased size of Walsh Table for 32 and 64 senders. As a result it takes more time to traverse the entire table and operate on the data to obtain the throughput.

We have also compared the Range of Throughput values for different senders with varying input size. The increased range for 32 & 64 senders proves the above statement.

**Improvements:**

As I have not considered any flow control protocol, no errors have been injected in the data. Absence of errors in all packets deviates from practical scenarios.