**SSN College of Engineering, Kalavakkam**

**Department of Computer Science and Engineering**

**UCS1511 NETWORKS LAB**

# Exercise 7 - Hamming Code Error Detection and Correction

**Name :** Kshitij Sharma **Roll No. :** 185001080 **Date :** 13/10/2020

**Aim:**

To implement hamming code error detection and correction using C socket program.

# Algorithm

## Server (Sender)

* 1. Create a socket descriptor with **socket()** system call with AF INET (IPV4 domain), SOCK STREAM, default protocol and store as sockfd.
  2. If sockfd is a negative number, socket creation failed, end program.
  3. Create sockaddr in object to assign IP address and Port number for socket. Set family to AF INET, IP address to INADDR ANY to accept connections from any client and port number required.
  4. Bind newly created socket to addresss given in sockaddr in.
  5. If bind is non zero, bind failed, print error message and terminate.
  6. Listen on the socked defined for as many clients as required. If **listen()** returns non zero value, print error message and terminate.
  7. Read data to send from user into buffer, let len of buffer be ‘n’ bits.
  8. Find number of redundant bits ‘r’ such that 2*r > n* + *r* + 1
  9. Create a new buffer of size ‘n + r’ bits.
  10. LOOP i: 1 *→* (n+r)
      + IF i is a power of 2, continue
      + Else buffer[i] = data[i]
  11. LOOP i: 0 *→* r
      + pos = 2*i*
      + count = 0
      + LOOP j: 1 *→* (n+r)
        - IF j == pos, continue
        - ELSE IF binary(pos) in binary(j), count += buffer[j]
      + arr[pos] = 1 if count is odd, else 0
  12. Introduce error at random bit of data buffer.
  13. Accept connections of client into new fd using **accept()** system call.
  14. Send data buffer to client using **send()** system call.
  15. Close connections on socket using **close()** and terminate program.

## Client (Receiver)

* 1. Create a socket descriptor with **socket()** system call with AF INET (IPV4 domain), SOCK STREAM, default protocol and store as sockfd.
  2. If sockfd is a negative number, socket creation failed, end program.
  3. Create sockaddr in object to assign IP address and Port number for socket. Set family to AF INET, IP address to localhost(127.0.0.1) to connect to server and port number required.
  4. Connect the client to server at address given in socket descriptor using **connect()** system call.
  5. If connect() returns -1, connection failed; Print error message and terminate the program.
  6. Read data sent by server into buffer using **read()** system call.
  7. Find the length ‘n’ of buffer.
  8. Number of redundant bits *r* = log2 *n*
  9. Initialise arr of length ‘r’ with 0.
  10. LOOP i: 0 *→* r
      + pos = 2*i*
      + count = 0
      + LOOP j: 1 *→* (n+r)
        - IF binary(pos) in binary(j), count += buffer[j]
      + arr[i] = 1 if count is odd, else 0
  11. Convert arr to decimal
  12. IF it is non zero, the decimal value represents the location of error and toggle bit at the location in the received buffer
  13. ELSE IF it is zero, no error occurred during transmission or multiple bits were incorrect.
  14. Print the buffer to user.
  15. Close the connections on socket using **close()** and terminate program.

# Program

## Server Side

*#include <stdio.h> #include <stdlib.h> #include <time.h> #include <sys/types.h> #include <sys/socket.h> #include <netinet/in.h> #include <stdlib.h> #include <string.h> #include <unistd.h> #include <arpa/inet.h> #include <fcntl.h> #include <stdbool.h> #include <math.h> #define MAXLINE 1024*

*#include "Hamming.h"*

int main(int argc, char \*\* argv)

{

srand(time(NULL));

if (argc < 2){

fprintf(stderr, "Enter port number as argument!\n"); exit(EXIT\_FAILURE);

}

int PORT = atoi(argv[1]);

int sockfd, newfd, n, arr[30], count = 0, bin; char buff[MAXLINE], buffer[MAXLINE], data\_t[40]; int i, j, r, total, nob, rem, dig, pos;

long data;

struct sockaddr\_in servaddr,cliaddr;

if((sockfd = socket(AF\_INET, SOCK\_STREAM, 0)) < 0)

{

perror("Socket creation failed!"); exit(1);

}

bzero(&servaddr,sizeof(servaddr)); servaddr.sin\_family = AF\_INET; *// IPv4* servaddr.sin\_addr.s\_addr = INADDR\_ANY; servaddr.sin\_port = htons(PORT);

if(bind(sockfd, (const struct sockaddr \*)&servaddr,sizeof(servaddr)) < 0)

{

perror("Bind failed!"); exit(1);

}

int len, m; listen(sockfd, 2);

printf("Enter the data to send: "); scanf("lu", &data);

n = countbits(data); r = log2(n);

r = floor(r);

*// Finding number of redundant bits*

while(pow(2, r) < n + r + 1) r += 1;

printf("\nNo.of redundant bits required: d\n", r); total = n + r;

nob = floor(log2(total)); for(i = 1; i <= total; i++)

{

dig = data  10; if(isapower2(i) == 0)

{

arr[total - i] = dig;

data /= 10;

}

else

arr[total-i]=0;

}

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

{

for(j = 1; j <= total; j++)

{

if((int)(pow(2, i)) != j)

{

bin = binary(j); if(ispresent(bin, i + 1))

count += arr[total - j];

}

}

if(count  2 == 0)

arr[total - (int)(pow(2, i))] = 0; else

arr[total - (int)(pow(2, i))] = 1; count = 0;

}

printf("\nData with redundant bits: "); for(i = 0; i < total; i++)

printf(" d", arr[i]);

*// printf("\nEnter error position: ");*

*// scanf(" d", &pos);*

pos = rand()  total + 1;

printf("\nIntroducing error automatically at bit: d\n", pos);

if(arr[total - pos] == 0) arr[total - pos] = 1;

else

arr[total - pos] = 0;

int k = 0; long num = 0;

for(i = total - 1; i >= 0; i--)

{

num += pow(10, k) \* arr[i]; k++;

}

sprintf(data\_t, " lu", num);

printf("Data transmitted is s\n", data\_t);

len = sizeof(cliaddr);

newfd = accept(sockfd, (struct sockaddr\*)&cliaddr, &len); m = write(newfd, data\_t, sizeof(data\_t));

}

## Client Side

*#include <stdio.h> #include <sys/types.h> #include <sys/socket.h> #include <netinet/in.h> #include <string.h> #include <unistd.h> #include <arpa/inet.h> #include <stdlib.h> #include <math.h> #define MAXLINE 1024*

*#include "Hamming.h"*

int main(int argc, char \*\*argv)

{

if (argc < 2){

fprintf(stderr, "Please enter port number as second argument!\n"); exit(EXIT\_FAILURE);

}

int PORT = atoi(argv[1]); long num;

int sockfd, total, i, rem, arr[20], count = 0, r = 0, result = 0, bin, j,

*‹→* newarr[20], finalarr[20];

char buffer1[40];

struct sockaddr\_in servaddr;

if ((sockfd = socket(AF\_INET, SOCK\_STREAM, 0)) < 0) { perror("Socket creation failed!");

exit(1);

}

bzero(&servaddr,sizeof(servaddr)); servaddr.sin\_family = AF\_INET; servaddr.sin\_port = htons(PORT);

servaddr.sin\_addr.s\_addr = inet\_addr("127.0.0.1"); int n, len;

connect(sockfd, (struct sockaddr\*)&servaddr, sizeof(servaddr)); n = read(sockfd, buffer1, sizeof(buffer1));

num = atol(buffer1); total = countbits(num);

printf("Received data: lu\n", num); i = 1;

while(num > 0)

{

rem = num  10; arr[total - i] = rem; num /= 10;

i++;

}

for(i = 1; i <= total; i++)

if(ceil(log2(i)) == floor(log2(i))) r++;

int k = 0;

for(i = 0; i < 4; i++)

{

for(j = 1; j <= total; j++)

{

bin = binary(j); if(ispresent(bin, i + 1))

count += arr[total - j];

}

if(count  2 == 0)

result += pow(10, k) \* 0; else

result += pow(10, k) \* 1; k++;

count=0;

}

int error = decimal(result);

printf("\nError bit in binary: d\n", result); printf("\nError in bit d\n", error); if(arr[total - error] == 0)

arr[total - error] = 1;

else

arr[total - error] = 0;

k = 0;

printf("\nData after error correction: "); for(i = total - 1; i >= 0; i--)

{

newarr[k] = arr[i]; k++;

}

int x = 0;

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

{

if(ceil(log2(i + 1)) != floor(log2(i + 1)))

{

finalarr[x] = newarr[i]; x++;

}

}

for(i = x - 1; i >= 0; i--)

printf(" d", finalarr[i]); printf("\n");

return 0;

}

## Hamming Code speciftc funtions

int countbits(long num)

{

int r, count = 0; while(num > 0)

{

num = num / 10;

count++;

}

return count;

}

int binary(int num)

{

int bin = 0, r; int i = 0; while(num > 0)

{

r = num  2;

bin += r \* pow(10, i); num /= 2;

i++;

}

return bin;

}

int ispresent(int num,int pos)

{

int rem;

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

{

rem = num  10; num = num / 10;

}

if(rem == 1)

return 1;

else

}

return 0;

int isapower2(int n)

{

if(ceil(log2(n)) == floor(log2(n))) return 1;

else

return 0;

}

int decimal(int num)

{

int rem, i = 0, result; while(num > 0)

{

rem = num  10;

result += pow(2, i) \* rem; num /= 10;

i++;

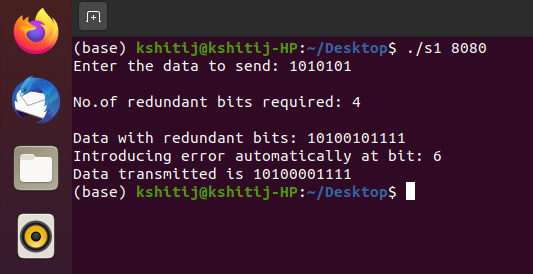
}

return result;

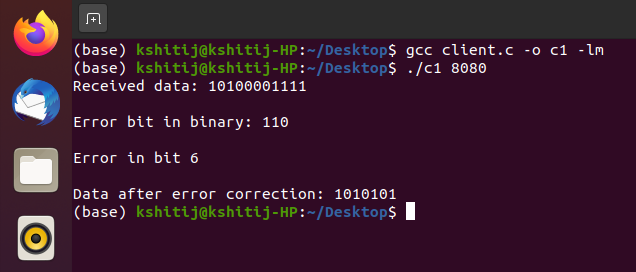
}

# Output

**Server:**



**Client:**



# Learning Outcomes:

* + We learn how to create a simple TCP client server connection.
  + We learn how to appropriate system calls to set up Server and Client Programs.
  + We learn the structure of ARP packet.
  + We handle ARP request and send appropriate response to simulate ARP.