KANDRA KSHEERAJ **19BCE0829**

CSE 1004 – Network and Communication

SLOT: L47+L48

Faculty: SRIMATHI C mam

LAB Digital Assignment- 2

Lab Date: 24-03-2021

1. Hamming code

Input- No of bits in the data word
Calculate the no of redundant bits
then Start calculating the redundant bits
Hamming code- ODD parity
Output - Codeword
Sender side Receiver side
flip any one bit and calculate the syndrome bits
do as 2 function as c as sender and receiver
for any input u should manual calculation in word document
data word should be input

KANDRA KSHEERAJ
19BCE0829

when $K = 7$ $r = 4$ $2^4 \geq 12$

1101001 – Data word Code word = 7+4

11	10	9	8	7	6	5	4	3	2	1
			2^3				2^2		2^1	2^0
			r_3				r_2		r_1	r_0

1101001 $m = 7$, $r = 4$, $n = 11$

1	1	0	r_3	1	0	0	r_2	1	r_1	r_0
11	10	9	8	7	6	5	4	3	2	1

$R_1 \rightarrow 1, 3, 5, 7, 9, 11$

$R_2 \rightarrow 2, 3, 6, 7, 10, 11$

$R_3 \rightarrow 4, 5, 6, 7$

$R_4 \rightarrow 8, 9, 10, 11$

Pos	$R_8 R_2 R_1 R_0$
0	0000
1	0001
2	0010
3	
11	

$R_{16} \rightarrow 16$

$r_0 = 1\ 3\ 5\ 7\ 9\ 11$
Parity 1 1 0 1 0 1
$\boxed{r_0 = 1}$

$r_1 = 2\ 3\ 6\ 7\ 10\ 11$
0 1 0 1 1 1
$\boxed{r_1 = 0}$

$r_3 = 4\ 5\ 6\ 7$
1 0 0 1
$\boxed{r_3 = 1}$

$r_4 = 8\ 9\ 10\ 11$
0 0 1 1
$\boxed{r_4 = 0}$

11001001101 – codeword
11 10 9 8 7 6 5 4 3 2 1

1 bit flipped on receiver side : 11001001101
4th bit 11001000101
11 10 9 8 7 6 5 4 3 2 1

Calculate Syndrome

$S_0 = 1, 3, 5, 7, 9, 11$
0 1 1 0 1 0 1

$S_1 = 2\ 3\ 6\ 7\ 10\ 11$
0 0 1 0 1 1 1

$S_2 = 4\ 5\ 6\ 7$
1 0 0 0 1

$S_3 = 8\ 9\ 10\ 11$
0 0 0 1 1

$S_3 S_2 S_1 S_0 = 0100$
4 - decimal equivalent

S_0, 4th position got flipped

Program Code:

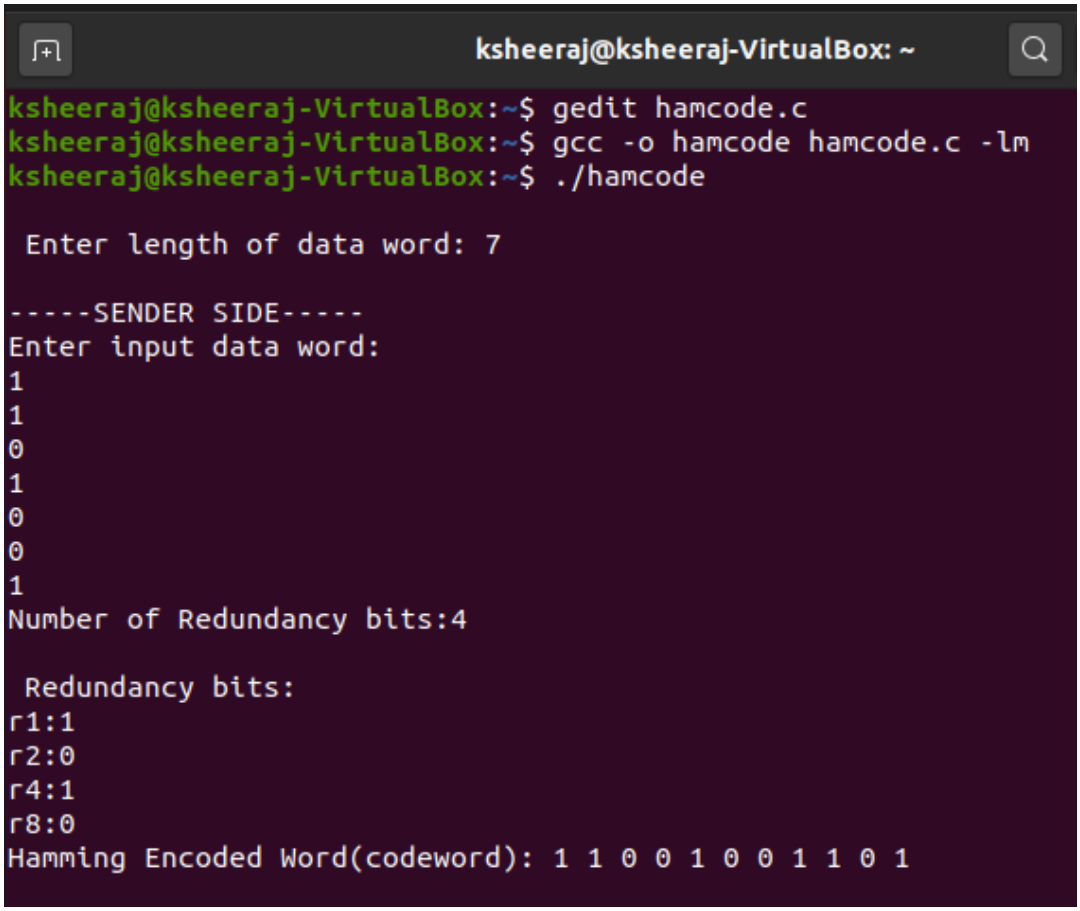
```
#include <stdio.h>
#include <math.h>
int ham_code(int position,int n,int codeword[])
{
     int count=0,i,j;
     i=position-1;
     while(i<n)
     {
     for(j=i;j<i+position;j++)
     {
          if(j<n)
          count=count+codeword[j];
     }
          i=i+2*position;
     }
     if(count%2 == 0)
```

```
    return 0;
    else
    return 1;
}
int sender(int a)
{
    int i,n,j,k;
    int data[a];
    printf("Enter input data word: \n");
    for(i=0;i<a;i++)
    {
        scanf("%d",&data[i]);
    }
    int l=0,r=0;
    while(a>(int)pow(2,l)-(l+1))
    {
        r++;l++;
 }
    printf("Number of Redundancy bits:%d\n",r );
    n = r + a;
    int codeword[n];
    j=k=0;
    for(i=0;i<n;i++)
    {
        if(i==((int)pow(2,k)-1))
    {
        codeword[i]=0;
        k++;
    }
    else
    {
        codeword[i]=data[a-1-j];
        j++;
    }
}
    printf("\n Redundancy bits:\n");
    for(i=0;i<r;i++)
    {
        int position = (int)pow(2,i);
        int value = ham_code(position,n,codeword);
        printf("r%d:%d\n",position,value );
        codeword[position-1]=value;
```

```
     }
     printf("Hamming Encoded Word(codeword): ");
     for(i=n-1;i>=0;i--)
          printf("%d ",codeword[i]);
     return r;
}
void receiver(int t,int r){
     int i;
     int n=t+r;
     printf("Enter Received Hamming Encoded
Word(codeword): \n");
     int received[n];
     for(i=0;i<n;i++)
          scanf("%d",&received[n-i-1]);
     int error_pos = 0;
     printf("Syndrome bits:\n");
     for(i=0;i<r;i++)
     {
     int position = (int)pow(2,i);
     int value = ham_code(position,n,received);
     printf("s%d :%d\n",position,value );
     if(value == 1)
     error_pos+=position;
}
     if(error_pos == 0)
          printf("Message is correct\n");
     else
     {
          printf("Error detected at bit position:
%d\n",error_pos);
          printf("Corrected Word: \n");
     for(i=0;i<n;i++)
     {
          if(n-i==error_pos)
          printf("%d ",received[n-i-1]^1 );
     else
          printf("%d ",received[n-i-1] );
     }
  }
     printf("\n");
}
int main()
```

```
{
     int w,re;
     printf("\n Enter length of data word: ");
     scanf("%d",&w);
     printf("\n-----SENDER SIDE-----\n");
     re=sender(w);
     printf("\n\n-----RECEIVER SIDE-----\n");
     receiver(w,re);
return 0;
}
```

Output Screenshot:



KANDRA KSHEERAJ
19BCE0829

No corruption in message

```
-----RECEIVER SIDE-----
Enter Received Hamming Encoded Word(codeword):
1
1
0
0
1
0
0
1
1
0
1
Syndrome bits:
s1 :0
s2 :0
s4 :0
s8 :0
Message is correct
```

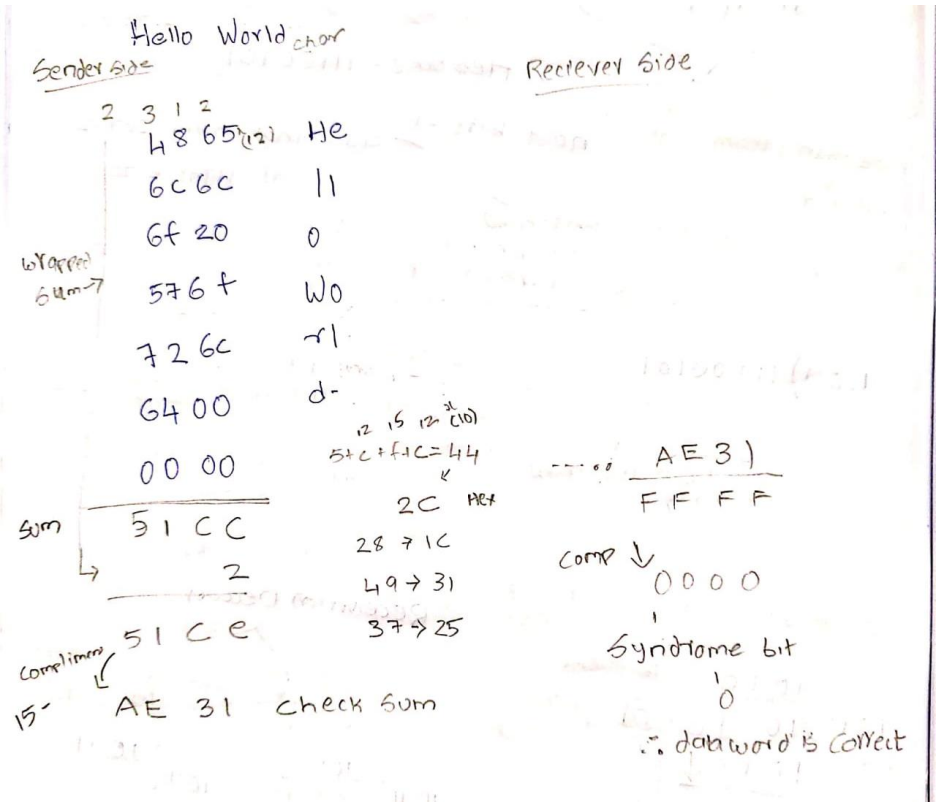
When 1 bit flipped on receiver side

```
-----RECEIVER SIDE-----
Enter Received Hamming Encoded Word(codeword):
1
1
0
0
1
0
0
0
1
0
1
Syndrome bits:
s1 :0
s2 :0
s4 :1
s8 :0
Error detected at bit position: 4
Corrected Word:
1 1 0 0 1 0 0 1 1 0 1
```

KANDRA KSHEERAJ
19BCE0829

2. Checksum

Sender side and Receiver side as 2 functions



Program Code:

```
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
struct array
{
     int arr[50];
};
     struct array convert(char a[])
  {
     int n=strlen(a),i,j=0;
     struct array y;
     printf("\n message converted to hexdecimal: \n");
     for(i=0;i<strlen(a);i++)
     {
          char hex[20];
          int k;
          if(i+1<=strlen(a)){
          sprintf(hex , "%02X%02X", a[i] & 0xff,a[i+1]
&0xff);
```

```
        y.arr[j]= (int)strtol(hex, NULL,16);
    }
    else
    {
        sprintf(hex , "%04X", a[i] & 0xff);
        y.arr[j]= (int)strtol(hex, NULL,16);
    }
        printf("%s \n",hex);
        i++;j++;
    }
    return y;
}
int checksum(int arr[],int n)
{
 int sum=0;
 for(int i=0;i<n;i++)
 {
    sum=sum+arr[i];
    if(sum>0XFFFF)
    {
    sum = sum/0X10000+sum%0X10000;
    }
 }
    sum=0XFFFF-sum;
 return sum;
}

int sender(char s[])
{
    struct array x=convert(s);
    int n = strlen(s);
    n=(n+1)/2+1;
    int cksum= 0;
    x.arr[n-1]=cksum;
    cksum=checksum(x.arr,n);
    x.arr[n-1]=cksum;
    printf("\n wrapped sum\n");
    for (int i=0;i<n;i++)
    {
        printf("%X \n",x.arr[i]);
    }
 return cksum;
```

```
}

int receiver(char s[],int cksm)
{
     struct array x=convert(s);
     int n = strlen(s);
     n=(n+1)/2+1;
     int cksum= 0;
     x.arr[n-1]=cksm;
     cksum=checksum(x.arr,n);
     printf("\n with checksum\n");
  for (int i=0;i<n;i++)
     {
          printf("%X \n",x.arr[i]);
     }
     if(cksum==0)
     {
     printf("message sucessfully recived at the recever
side");
     }
     else
     {
     printf(" message is corrupted");
  }
}
int main()
{
     char msg[50],recmsg[50];
     int checksumr;
     printf("\n ------SENDER SIDE------");
     printf("\n enter message : ");
     scanf("%s",msg);
     checksumr=sender(msg);
     printf("\n -------RECEIVER SIDE-------");
     printf("\n received message : ");
     scanf("%s",recmsg);
     receiver(recmsg,checksumr);
     return 0;
}
```

Output Screenshot:

```
ksheeraj@ksheeraj-VirtualBox:~$ gedit checksum.c
ksheeraj@ksheeraj-VirtualBox:~$ gcc -o checksum checksum.c -lm
ksheeraj@ksheeraj-VirtualBox:~$ ./checksum

 ------SENDER SIDE------
 enter message : Router

 message converted to hexdecimal:
526F
7574
6572

 wrapped sum
526F
7574
6572
D2A9
```

Data word is correct on receiver side

```
 -------RECEIVER SIDE-------
 received message : Router

 message converted to hexdecimal:
526F
7574
6572

 with checksum
526F
7574
6572
D2A9
message correctly recived at the recever side
```

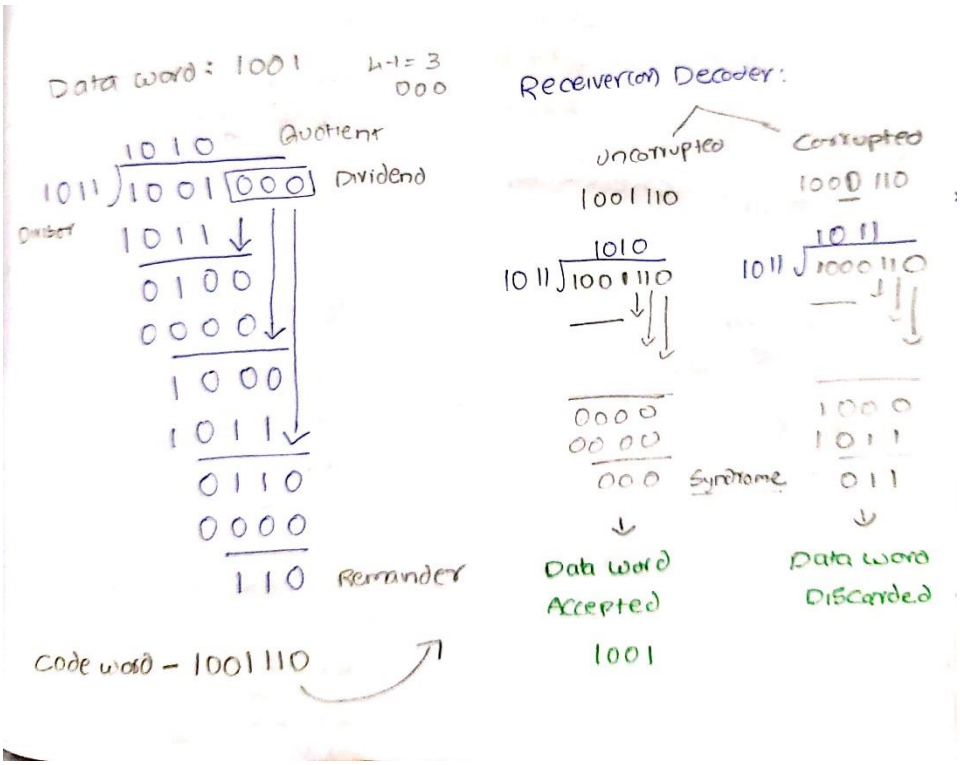
Corrupted dataword on receiver side

```
 -------RECEIVER SIDE-------
 received message : router

 message converted to hexdecimal:
726F
7574
6572

 with checksum
726F
7574
6572
D2A9
 message is corruptedksheeraj@ksheeraj-VirtualBox:~$
```

KANDRA KSHEERAJ
19BCE0829

3. CRC

Sender side and Receiver side as 2 functions



Program Code:

```
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
char* xor(char *a, char *b){
int i, length=strlen(a);
char *result;
result = (char *)malloc(sizeof(char)*length);
for (i=0;i<length;i++){
     if(a[i]==b[i])
          {result[i]='0';}
     else
          {result[i]='1';}
     }
     return(result);
}
     char* mod2div(char *dividend,char *divisor){
     int dividend_length = strlen(dividend);
     int divisor_length = strlen(divisor);
     int i;
```

```
    char *tmp;
    tmp = (char *)malloc((divisor_length-
1)*sizeof(char));
    for(i=0;i<divisor_length;i++){
        tmp[i] = dividend[i];
    }
    int index=divisor_length;
    char* zero;
    while (index < dividend_length)
    {
    if (tmp[0] == '1')
    {
    tmp = xor(divisor, tmp);
    for(i=0;i<divisor_length-1;i++)
        tmp[i]=tmp[i+1];
        tmp[divisor_length-1]=dividend[index];
    }
    else
    {
    zero = (char *)malloc(sizeof(char)*(divisor_length));
    for(i=0;i<divisor_length;i++)
        {zero[i]='0';}
        tmp = xor(zero, tmp);
    for(i=0;i<divisor_length-1;i++)
        tmp[i]=tmp[i+1];
        tmp[divisor_length-1]=dividend[index];
    }
    index += 1;
    }
    if (tmp[0] == '1')
    {
    tmp = xor(divisor, tmp);
    }
    else
    {
        zero = (char
*)malloc(sizeof(char)*(divisor_length));
    for(i=0;i<divisor_length;i++)
        {zero[i]='0';}
    tmp = xor(zero, tmp);
    }
    char *crc;
```

```
    divisor_length--;
    crc = (char *)malloc((divisor_length)*sizeof(char));
    for(i=0;i<divisor_length;i++)
        crc[i]=tmp[i+1];
        crc[divisor_length]='\0';
        printf("remainder: %s\n", crc);
    return(crc);
}
char* append_data(char *message,char *append_with,int
zeros){
 int i;
 int message_length = strlen(message);
 int append_with_length = strlen(append_with);
 char *result;
 append_with_length--;
 result = (char
*)malloc((message_length+append_with_length)*sizeof(char))
;
 for(i=0;i<strlen(message);i++)
    result[i]=message[i];
 if(zeros==1){
 for(i=0;i<append_with_length;i++)
    result[message_length+i]='0';
 }
 else{
    append_with_length++;
 for(i=0;i<append_with_length;i++)
    result[message_length+i]=append_with[i];
 }
 result[message_length+append_with_length]='\0';
return(result);
}
void sender(char dataword[],char divisor[])
{
char *appended_data=append_data(dataword,divisor,1);
printf("Appended data: %s\n", appended_data);
char *crc = mod2div(appended_data,divisor);
 char *codeword = append_data(dataword,crc,0);
 printf("codeword: %s\n",codeword);
}
void receiver(char received_codeword[],char divisor[])
{
```

```
 char *received_crc = mod2div(received_codeword,divisor);
 int i;
 for(i=0;i<strlen(divisor)-1;i++){
 if(received_crc[i]!='0'){
 printf("Message is corrupted\n");
 exit(0);
 }
 }
 printf("Message is correct \n");
}
int main(){
 int i;
 int dataword_length,divisor_length;
     printf("Enter dataword Length: ");
 scanf("%d",&dataword_length);
 char dataword[dataword_length];
     printf("Enter dataword: ");
 scanf("%s",dataword);
     printf("Enter length of divisor: ");
 scanf("%d",&divisor_length);
 char divisor[divisor_length];
     printf("Enter divisor(generator polynomial): ");
 scanf("%s",divisor);
     printf("\n-----SENDER SIDE-----\n");
 sender(dataword,divisor);
     printf("\n-----RECEIVER SIDE-----\n");
 char received_codeword[dataword_length+divisor_length-1];
     printf("Enter received dataword: ");
 scanf("%s",received_codeword);
 receiver(received_codeword,divisor);
 return 0;
}
```

Output Screenshot:

Dataword accepted on receiver side

```
ksheeraj@ksheeraj-VirtualBox:~$ gcc -o crc crc.c -lm
ksheeraj@ksheeraj-VirtualBox:~$ ./crc
Enter dataword Length: 4
Enter dataword: 1001
Enter length of divisor: 4
Enter divisor(generator polynomial): 1011

-----SENDER SIDE-----
Appended data: 1001000
remainder: 110
codeword: 1001110

-----RECEIVER SIDE-----
Enter received dataword: 1001110
remainder: 000
Message is correct
```

Dataword Discarded on receiver side

```
-----SENDER SIDE-----
Appended data: 1001000
remainder: 110
codeword: 1001110

-----RECEIVER SIDE-----
Enter received dataword: 1000110
remainder: 011
Message is corrupted
```

4. Stop and wait ARQ

SENDER SIDE (function)

1. Enter the No of frames to be sent : N
   Let N =15
2. Frame Numbering
   0 1 0 1 0 1 0 1 0 1 0 1 0 1 0

3 Menu

   1. Frame t o be send(sent the next frame) - Sleep (2000)

2 .Frame Resent (send the previous frame)- Timer expired , ACK lost ,Frame lost

Receiver side (function)

1. Menu (i) Receive the frame successfully (ii) Frame lost

GO BACK N ARQ

SENDER SIDE

1. Enter the No of frames to be sent : n (15) and
2. N – Modulo arithmetic(4)
3. Calculate the sender Window Size and Receiver window size
4. Frame Numbering (0 1 2 3 0 1 2 3 0 1 2 3 0 1 2)
5. Sender Window(Sf, Sn and S size)
6. Menu 1 . Frames to be sent (max window size)
   2. Ack Received
   3. Timer out

)

Program Code:

```
#include <stdio.h>
int B[30];
int receiveFrame(int p){
 if((p-1)%2==B[p-1]){
 printf(" Received Frame & ID = %d\nAcknowledgement for
the frame number %d \n", B[p-1],p);
 return 1;
 }
 else{
     printf("Frame lost\n");
 return 0;
 }
}
void sendFrame(int m){
 for(int i=0; i<m; i++){
     if(i%2==0)
     B[i] = 0;
     else
```

```
    B[i] = 1;
 }
 for(int i=1; i<=m; i++){
    printf("Sending Frame Number%d & ID = %d\n",i,B[i-
1]);
    sleep(1);
    if(receiveFrame(i))
    printf("Acknowledgement Received for the frame %d &
ID = %d\n\n",i,B[i-1]);
    else
    printf("Time Out, Frame Lost. Frame ID = %d\nSending
Frame %d again\n",B[i-1],i);
    }
 }
void main()
{
 int num;
 printf(" Number of frames ");
 scanf("%d",&num);
 sendFrame(num);
}
```

Output Screenshot:

```
ksheeraj@ksheeraj-VirtualBox:~$ gcc -o sawarq sawarq.c -lm
ksheeraj@ksheeraj-VirtualBox:~$ ./sawarq
 Number of frames 5
Sending Frame Number1 & ID = 0
 Received Frame & ID = 0
Acknowledgement for the frame number 1
Acknowledgement Received for the frame 1 & ID = 0

Sending Frame Number2 & ID = 1
 Received Frame & ID = 1
Acknowledgement for the frame number 2
Acknowledgement Received for the frame 2 & ID = 1

Sending Frame Number3 & ID = 0
 Received Frame & ID = 0
Acknowledgement for the frame number 3
Acknowledgement Received for the frame 3 & ID = 0

Sending Frame Number4 & ID = 1
 Received Frame & ID = 1
Acknowledgement for the frame number 4
Acknowledgement Received for the frame 4 & ID = 1

Sending Frame Number5 & ID = 0
 Received Frame & ID = 0
Acknowledgement for the frame number 5
Acknowledgement Received for the frame 5 & ID = 0
```