KANDRA KSHEERAJ **19BCE0829**

CSE 1004 – Network and Communication

SLOT: L47+L48

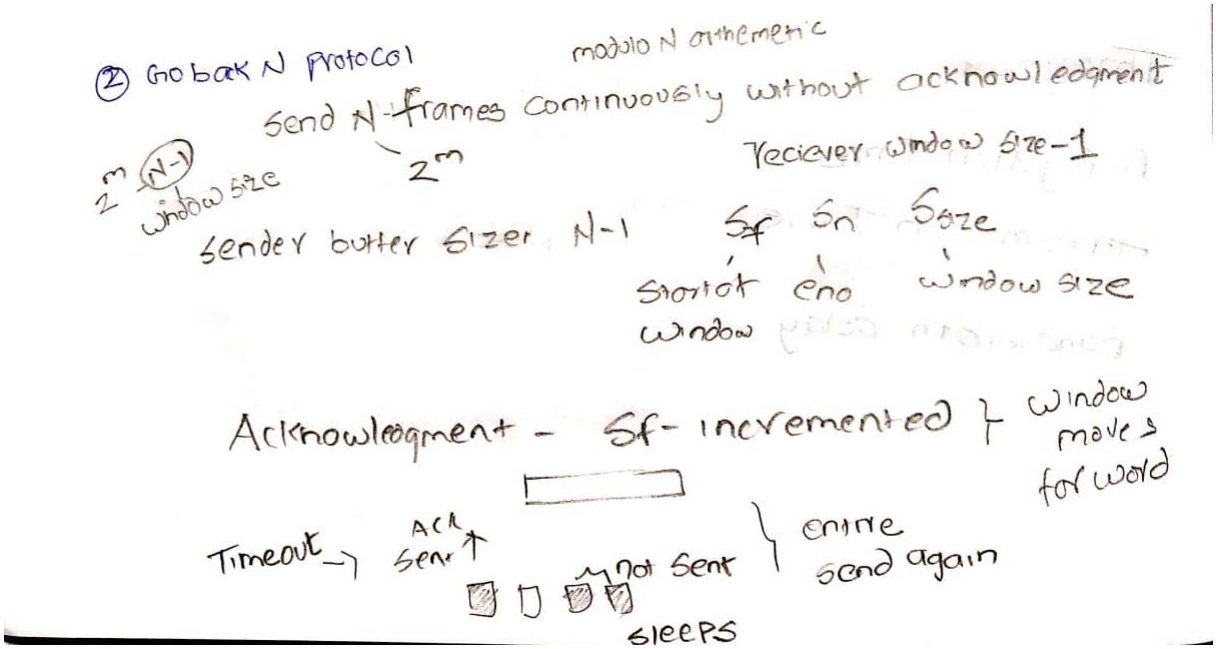
Faculty: SRIMATHI C mam

LAB Digital Assignment- 3

Lab Date: 16-03-2021

1. Go BACK N ARQ

The size of the sending window determines the sequence number of the outbound frames. If the sequence number of the frames is an n-bit field, then the range of sequence numbers that can be assigned is 0 to 2^n-1. Consequently, the size of the sending window is 2^n-1. Thus in order to accommodate a sending window size of 2^n-1, a n-bit sequence number is chosen. If the acknowledgment of a frame is not received within an agreed upon time period, all frames starting from that frame are retransmitted.



Inputs: 1. No of frames to be sent :8

2. N =4 or m=2(no of bits combination -)

Processing: 1.Assign the frameid(Modulo N) to each frame

0 1 2 3 0 1 2 3

2. Sender Window size(N-1 or 2^m -1) and receiver Window size(1)

Sender Events (sf, sn, and Ssize)

1. Frame sent : Sender window size(3) – 3 frames that can be sent without waiting for the acknowledgement, after each frame is sent, Increment the Sn value(Modulo N Arithemetic)
2. Acknowledgement Received : Increment Sf value (modulo N arithemetic)
3. Timer expires: Resent all the frames from sf to sn

Receiver Events (Rn)

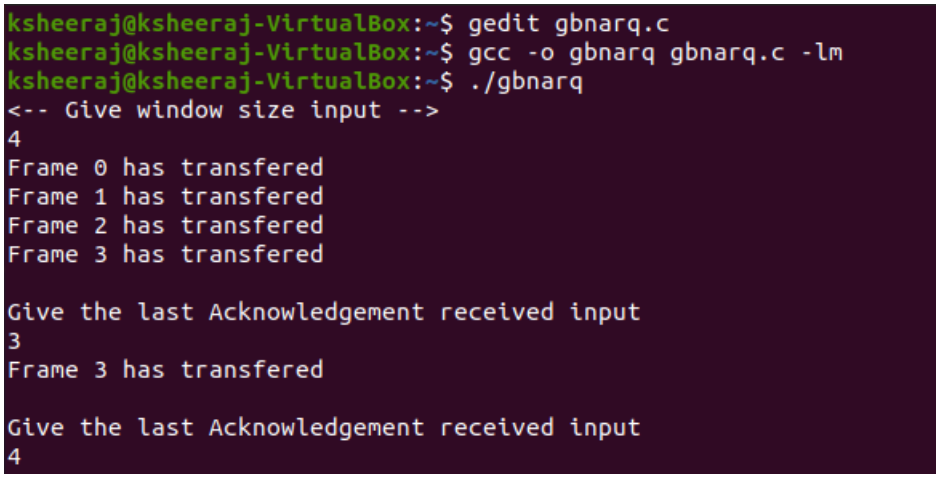
1. Frame received successfully- Increments Rn(Modulo N arithemetic) , sent the ACK(Rn)
2. Corrupted frame or Frame out of order: Rn remain the same , discard the frame

Code:

```
#include<stdio.h>
void main()
{
int ws;
int transmit=0;
int acknowled;
printf("<-- Give window size input -->\n");
scanf("%d",&ws);
while(1){
for(int i=0;i<ws;i++)
{
  printf("Frame %d has transfered\n",transmit);
  transmit++;
if(transmit == ws)
{
break;
}
}
  printf("\nGive the last Acknowledgement received
input\n");
```

```
  scanf("%d",&acknowled);
if(acknowled==ws)
{
    break;
}
else
transmit = acknowled;
}
}
```

Output Screenshot:



2. Selective Repeat ARQ

The receiver records the sequence number of the earliest incorrect or un-received frame. It then fills the receiving window with the subsequent frames that it has received. It sends the sequence number of the missing frame along with every acknowledgement frame.

The sender continues to send frames that are in its sending window. Once, it has sent all the frames in the window, it retransmits the frame whose sequence number is given by the acknowledgements. It then continues sending the other frames.

KANDRA KSHEERAJ
19BCE0829

③ Selective Repeat ARQ modulo N arithemetic

Se, Rev window size N/2

-ve Acknowledgement — when frame lost resend

When received - +ve

No order Accepts 0 1 2 3

When timer outs, resends only Particular frame

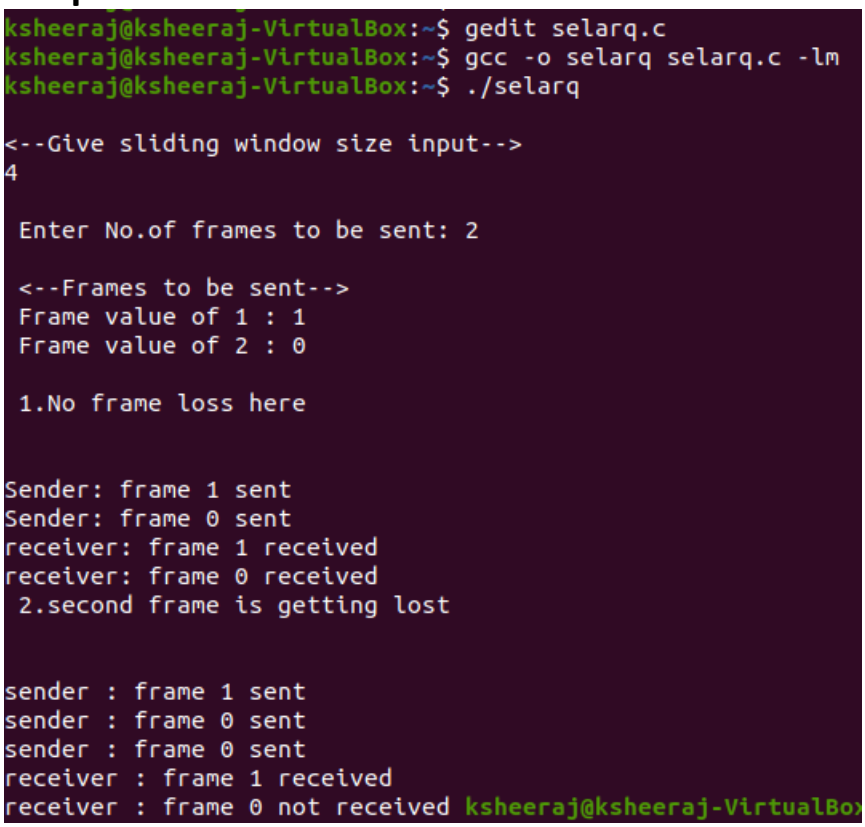
Code:

```
#include<stdio.h>
void main()
{
int m,j=0,d,sliding_window,frame_number;
printf("\n<--Give sliding window size input-->\n");
scanf("%d",&sliding_window);
printf("\n Enter No.of frames to be sent: ");
scanf("%d",&frame_number);
int Frames_arr[frame_number];
printf("\n <--Frames to be sent--> \n");
for(m=0;m<frame_number;m++)
{
   printf(" Frame value of %d : ",m+1);
   scanf("%d",&Frames_arr[m]);
}
   printf("\n 1.No frame loss here\n\n");
m=0;
while((j*sliding_window)<frame_number && m<frame_number)
{
d=(m-(j*sliding_window))/sliding_window;
if(d==0)
{
   printf("\nSender: frame %d sent ",Frames_arr[m]);
m++;
}
else {
   printf("\n\nwindow needs to be moved ");
j++;
}
}
m=0;
```

```
j=0;
while((j*sliding_window)<frame_number && m<frame_number)
{
d=(m-(j*sliding_window))/sliding_window;
if(d==0)
{
  printf("\nreceiver: frame %d received
",Frames_arr[m]);
m++;
}
else
j++;
}
m=0;
j=0;
  printf("\n 2.second frame is getting lost\n\n");
while((j*sliding_window)<frame_number && m<frame_number)
{
d=(m-(j*sliding_window))/sliding_window;
if(d==0)
{
  printf("\nsender : frame %d sent ",Frames_arr[m]);
if(m==1)
  printf("\nsender : frame %d sent ",Frames_arr[m]);
m++;
}
else
{
  printf("\n\n Window has to be moved ");
j++;
}
}
m=0,j=0;
while((j*sliding_window)<frame_number&&m<frame_number)
{
  d=(m-(j*sliding_window))/sliding_window;
if(d==0)
{
if(m!=1)
  printf("\nreceiver : frame %d received
",Frames_arr[m]);
if(m==1){
```

```
   printf("\nreceiver : frame %d not received
",Frames_arr[m]);
}
m++;
}
else
{
if(m==3)
   printf("\neceiver : frame %d
received\n",Frames_arr[1]);
j++;
}
}
}
```

Output Screenshot:



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

<--Give sliding window size input-->
4

 Enter No.of frames to be sent: 2

 <--Frames to be sent-->
 Frame value of 1 : 1
 Frame value of 2 : 0

 1.No frame loss here


Sender: frame 1 sent
Sender: frame 0 sent
receiver: frame 1 received
receiver: frame 0 received
 2.second frame is getting lost


sender : frame 1 sent
sender : frame 0 sent
sender : frame 0 sent
receiver : frame 1 received
receiver : frame 0 not received ksheeraj@ksheeraj-VirtualBox
```

3. Transmission Control Protocol (TCP)

Communications protocol using which the data is transmitted between systems over the network. In this, the data is transmitted into the form of packets. It includes error-checking, guarantees the delivery and preserves the order of the data packets.

Code:

KANDRA KSHEERAJ
19BCE0829

```
/****************** SERVER CODE ****************/

#include <stdio.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <string.h>

int main(){
  int welcomeSocket, newSocket;
  char buffer[1024];
  struct sockaddr_in serverAddr;
  struct sockaddr_storage serverStorage;
  socklen_t addr_size;

  /*---- Create the socket. The three arguments are: ---
-*/
  /* 1) Internet domain 2) Stream socket 3) Default
protocol (TCP in this case) */
  welcomeSocket = socket(PF_INET, SOCK_STREAM, 0);
/*---- Configure settings of the server address struct -
---*/
  /* Address family = Internet */
  serverAddr.sin_family = AF_INET;
  /* Set port number, using htons function to use proper
byte order */
  serverAddr.sin_port = htons(7891);
  /* Set IP address to localhost */
  serverAddr.sin_addr.s_addr = inet_addr("127.0.0.1");
  /* Set all bits of the padding field to 0 */
  memset(serverAddr.sin_zero, '\0', sizeof
serverAddr.sin_zero);

  /*---- Bind the address struct to the socket ----*/
  bind(welcomeSocket, (struct sockaddr *) &serverAddr,
sizeof(serverAddr));

  /*---- Listen on the socket, with 5 max connection
requests queued ----*/
  if(listen(welcomeSocket,5)==0)
    printf("Listening\n");
  else
  printf("Error\n");
```

```
  /*---- Accept call creates a new socket for the
incoming connection ----*/
  addr_size = sizeof serverStorage;
  newSocket = accept(welcomeSocket, (struct sockaddr *)
&serverStorage, &addr_size);

  /*---- Send message to the socket of the incoming
connection ----*/
  strcpy(buffer,"Hello World\n");
  send(newSocket,buffer,13,0);

  return 0;
}
```

/**************** CLIENT CODE ****************/**

```
#include <stdio.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <string.h>

int main(){
  int clientSocket;
  char buffer[1024];
  struct sockaddr_in serverAddr;
  socklen_t addr_size;

  /*---- Create the socket. The three arguments are: ---
-*/
  /* 1) Internet domain 2) Stream socket 3) Default
protocol (TCP in this case) */
  clientSocket = socket(PF_INET, SOCK_STREAM, 0);

  /*---- Configure settings of the server address struct
----*/
  /* Address family = Internet */
  serverAddr.sin_family = AF_INET;
/* Set port number, using htons function to use proper
byte order */
  serverAddr.sin_port = htons(7891);
  /* Set IP address to localhost */
```

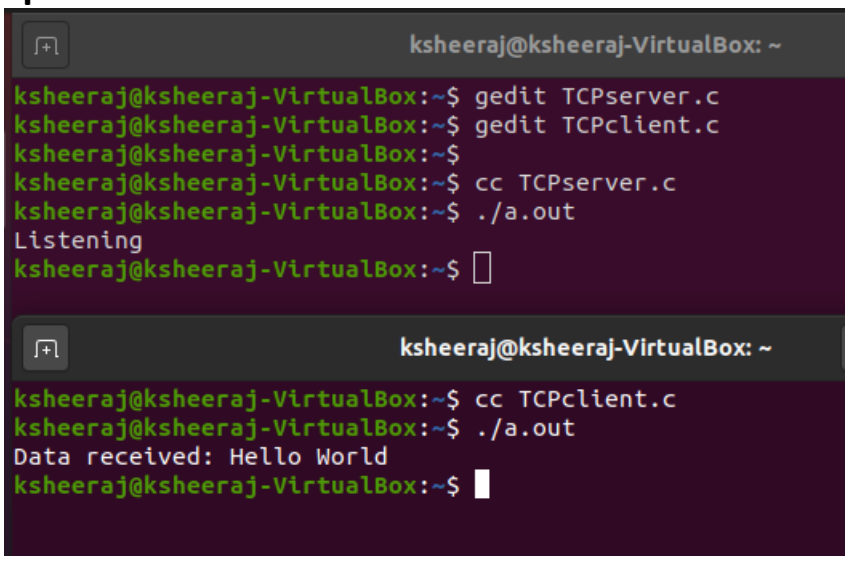
```
  serverAddr.sin_addr.s_addr = inet_addr("127.0.0.1");
  /* Set all bits of the padding field to 0 */
  memset(serverAddr.sin_zero, '\0', sizeof
serverAddr.sin_zero);

  /*---- Connect the socket to the server using the
address struct ----*/
  addr_size = sizeof serverAddr;
  connect(clientSocket, (struct sockaddr *) &serverAddr,
addr_size);

  /*---- Read the message from the server into the
buffer ----*/
  recv(clientSocket, buffer, 1024, 0);

  /*---- Print the received message ----*/
  printf("Data received: %s",buffer);
return 0;
}
```

Output Screenshot:



4. User Datagram Protocol (UDP)

It is same as the TCP protocol except this doesn't guarantee the error-checking and data recovery. If you use this protocol, the data will be sent continuously, irrespective of the issues in the receiving end.

Code:

KANDRA KSHEERAJ
19BCE0829

/**************** SERVER CODE ****************/**

```
#include <stdio.h>
#include <string.h>
#include <sys/socket.h>
#include <arpa/inet.h>
#include <ctype.h>

int main(void){
    int socket_desc;
    struct sockaddr_in server_addr, client_addr;
    char server_message[100], client_message[100];
    int client_struct_length = sizeof(client_addr);

    // Clean buffers:
    memset(server_message, '\0', sizeof(server_message));
    memset(client_message, '\0', sizeof(client_message));

    // Create UDP socket:
    socket_desc = socket(AF_INET, SOCK_DGRAM,
IPPROTO_UDP);

    if(socket_desc < 0){
        printf("Error while creating socket\n");
        return -1;
    }
    printf("Socket created successfully\n");

    server_addr.sin_family = AF_INET;
    server_addr.sin_port = htons(2000);
    server_addr.sin_addr.s_addr = inet_addr("127.0.0.1");

    // Bind to the set port and IP:
    if(bind(socket_desc, (struct sockaddr*)&server_addr,
sizeof(server_addr)) < 0){
        printf("Couldn't bind to the port\n");
        return -1;
    }
    printf("Done with binding\n");

    printf("Listening ..\n\n");
```

```
    // Receive client's message:
    if (recvfrom(socket_desc, client_message,
sizeof(client_message), 0,
         (struct sockaddr*)&client_addr,
&client_struct_length) < 0){
        printf("Couldn't receive\n");
        return -1;
    }
    printf("Received message from IP: %s and port: %i\n",
           inet_ntoa(client_addr.sin_addr),
ntohs(client_addr.sin_port));

    printf("Msg from client: %s\n", client_message);

    // Change to uppercase:
    for(int i = 0; client_message[i]; ++i)
        client_message[i] = toupper(client_message[i]);

    // Respond to client:
    strcpy(server_message, client_message);

    if (sendto(socket_desc, server_message,
strlen(server_message), 0,
         (struct sockaddr*)&client_addr,
client_struct_length) < 0){
        printf("Can't send\n");
        return -1;
    }
}
```

/**************** CLIENT CODE ****************/**

```
#include <stdio.h>
#include <string.h>
#include <sys/socket.h>
#include <arpa/inet.h>
#include <ctype.h>
int main(void){
    int socket_desc;
    struct sockaddr_in server_addr;
    char server_message[100], client_message[100];
    int server_struct_length = sizeof(server_addr);
```

```
    // Clean buffers:
    memset(server_message, '\0', sizeof(server_message));
    memset(client_message, '\0', sizeof(client_message));

    // Create socket:
    socket_desc = socket(AF_INET, SOCK_DGRAM,
IPPROTO_UDP);

    if(socket_desc < 0){
        printf("Error while creating socket\n");
        return -1;
    }
    printf("Socket created successfully\n");
    server_addr.sin_family = AF_INET;
    server_addr.sin_port = htons(2000);
    server_addr.sin_addr.s_addr = inet_addr("127.0.0.1");
    printf("Enter message: ");
    fgets(client_message, 20, stdin);
    // Send the message to server:
    if(sendto(socket_desc, client_message,
strlen(client_message), 0,
         (struct sockaddr*)&server_addr,
server_struct_length) < 0){
        printf("Unable to send message\n");
        return -1;
    }

    // Receive the server's response:
    if(recvfrom(socket_desc, server_message,
sizeof(server_message), 0,
         (struct sockaddr*)&server_addr,
&server_struct_length) < 0){
        printf("Error while receiving server's msg\n");
        return -1;
    }

    printf("Data received: %s\n", server_message);


    return 0;
}
```

Output Screenshot:

```
ksheeraj@ksheeraj-VirtualBox: ~
ksheeraj@ksheeraj-VirtualBox:~$ gedit udpserver.c
ksheeraj@ksheeraj-VirtualBox:~$ gedit udpclient.c
ksheeraj@ksheeraj-VirtualBox:~$
Terminal ksheeraj-VirtualBox:~$ cc udpserver.c
ksheeraj@ksheeraj-VirtualBox:~$ ./a.out
Socket created successfully
Done with binding
Listening ..
```

```
ksheeraj@ksheeraj-VirtualBox: ~
ksheeraj@ksheeraj-VirtualBox:~$ cc udpclient.c
ksheeraj@ksheeraj-VirtualBox:~$ ./a.out
Socket created successfully
Enter message: Hello World
Data received: HELLO WORLD

ksheeraj@ksheeraj-VirtualBox:~$
```