Name :- Rushikesh Karbhari Palve
Roll NO. 31258

| : Date : | Page No : |
|---|---|
| / /20 | |

# Assignment No. 4

DOS :- 31-10-2021

**Title :-** Study of Go back N and selective Repeat Modes of sliding window protocol.

## Problem statement :-

Write a program to simulate Go back N and selective Repeat Modes of sliding window protocol in Peer-to-peer mode.

## Objectives :-

① To learn Go Back N and selective Repeat Modes of sliding window protocol in Peer-to-peer mode.

## Outcomes :-

After completion of Assignment students will be able to -
① Understand the concept and write a program for simulating Go back N and selective Repeat Modes of sliding window protocol.

## THEORY :-

## 1.] GO-Back-N ARQ :-

Go-Back-N ARQ is a specific instance of the automatic repeat request (ARQ) protocol, in which the sending process continues to send a number of frames specified by a window size even without receiving an acknowledgement (ACK) packet from the receiver. It is a special case of the general sliding window protocol with the transmit window size of N and receive window size of 1. It can transmit N frames to the peer before requiring an ACK.

### Operation :-

The receiver process keeps track of the sequence number of the next frame it expects to receive and sends that number with every ACK it sends. The receiver will discard any frame that does not have the exact sequence number ~~that~~ it expects (either a duplicate frame it already acknowledged, or an out-of-order frame it expects to receive later) and will resend an ACK for the last correct in-order frame.

Once the sender has sent all of the frames in its window, it will detect that all of the frames since the first lost frame are outstanding, and will go back to the sequence number of the last ACK it received from the receiver process and ~~wit~~ fill its window starting with that frame and continue the process over again.

## Advantage:-

Go-Back-N ARQ is a more efficient use of a connection than stop-and-wait ARQ, since unlike waiting for an acknowledgement for each packet, the connection is still being utilized as packets are being sent. In other words, during the time that would otherwise be spent waiting, more packets are being sent.

## Disadvantage:-

In other words, ~~b~~ during the time that would otherwise be spent waiting, more packets are being sent. However, this method also results in sending frames multiple times - if any frame was lost or damaged, or the ACK acknowledging them was lost or damaged, then that frame and all following frames in the window (even if they were received without error) will be re-sent. To avoid this, Selective Repeat can be used.

## choosing a window size (N):-

There are a few things to keep in mind when choosing a value ~~of~~ for N:

1) the sender must not transmit too fast. N should be bounded by the receiver's ability to process packets.

2) N must be smaller than the number of sequence numbers (if they are numbered from zero to N) to verify transmission in cases of any packet (any data or Ack packet) being dropped. Given the bounds presented in ① and ②, choose N to be the largest number possible.

**ALGORITHM :-**

N = window size
$S_n$ = sequence Number
$S_b$ = sequence base
$S_m$ = sequence max
ack = ack number
nack = ~~first~~ non acknowledged

Receiver :
Do the following forever :
Randomly accept or reject packet
If the packet received and the packet is error free
Accept packet
Send a positive ack for packet
Else
Refuse packet
Send a negative ack for packet

Sender :
$S_b = 0$
$S_m = N-1$
ack = 0
Repeat the following steps forever :

```
send packet with ack
If positively ack is received :
ack++
Transmit a packet where $S_b$ <= ack <= $S_m$.
        packet are transmitted in order
Else
Enqueue the nack into the queue
// check if last packet in the window is sent
if (ack == $S_m$)
if (queue is not empty)
// start from the first nack packet
nack = queue.front();
empty the queue
ack = nack
$S_m$ = $S_m$ + (ack - $S_b$)
$S_b$ = ack .
```

## 2.] Selective Repeat :-

Selective Repeat is part of the automatic repeat-request (ARQ). With selective repeat, the sender sends a number of frames specified by a window size even without the need to wait for individual ACK from the receiver as in GO-Back-N ARQ. The receiver may selectively reject a single frame, which may be retransmitted alone; this contrasts with other forms of ARQ, which must send everyframe from that point again. The receiver accepts out-of-order frames and buffers them. The sender individually retransmits frames that have timed out.

## Advantage over Go-Back-N :-

Fewer Retransmissions.

## Disadvantages :-

More complexity at sender and receiver. Each frame must be acknowledged individually (no cumulative acknowledgements). Receiver may receive frames out of sequence.

## ALGORITHM :-

| | |
|---|---|
| N | = window size |
| $S_n$ | = Sequence Number |
| $S_b$ | = sequence base |
| $S_m$ | = Sequence max |
| ack | = ack number |
| nack | = First non acknowledged |

### Receiver :

```
Do the following forever :
Randomly accept or reject packet
If the packet received and the packet is error free
Accept packet
Send a positive ack for packet
Else
Refuse packet
send a negative ack for packet
```

### Sender :

$S_b = 0$

```
$S_m = N-1$
ack = 0
Repeat the following steps forever:
If the packet was not already positively
        acknowledged by receiver
Send packet with ack
If positively ack is received:
Transmit a packet where $S_b <= ark <= S_m$.
packets are transmitted in order
Else
Enqueue the nack into the queue
ack++
// check if last packet in the window is sent
if (ack == $S_m$)
if (queue is not empty)
// start from the first nack packet
nack = queue.front();
empty the queue
ack = nack
$S_m = S_m + (ack - S_b)$
$S_b$ = ack.
```

## CONCLUSION :-

Thus, we have studied and implemented the Go-Back-N and selective Repeat Modes of sliding window protocol in peer-to-peer mode.

**CODE :-**

```cpp
/*
 * Problem Statement :-
 *      Write a program to simulate Go back N and Selective Repeat Modes of Sliding
 *      Window Protocol in Peer-to-Peer mode.
*/

#include<bits/stdc++.h>
using namespace std;

class Frame
{
    friend class SlidingWindow;
    private:
        int data;
        bool ack;
    public:
        Frame()
        {
            data = 0;
            ack = true;
        }
};

class SlidingWindow
{
    private:
        Frame* frames;
    public:
        void sender(int);
        void recAck(int);
        void resendGb(int, int);
        void resendSr(int);
        void goBack(int, int);
        void selective(int, int);
};

void SlidingWindow:: sender(int n)
{
    frames = new Frame[n];

    for(int i=0; i<n; i++)
    {
        cout<<"\n\t Enter data for frame "<<i+1<<" : ";
        cin>>frames[i].data;
    }
}

void SlidingWindow::recAck(int r)
{
    frames[r].ack = false;
    cout<<"\n\t The Frame Number "<<r+1<<" is Not Received...!!"<<endl;

}
```

```cpp
void SlidingWindow::resendGb(int n, int r)
{
    cout<<"\n\t Resending frame from "<<r+1<<"..."<<endl;
    for(int i=r; i<n; i++)
    {
        frames[i].ack = true;
        cout<<"\n\t Received Data of frame "<<i+1<<" , "<<frames[i].data<<endl;
    }
}

void SlidingWindow::resendSr(int r)
{
    cout<<"\n\t Resending Frame Number "<<r+1<<"..."<<endl;
    frames[r].ack = true;
    cout<<"\n\t Received Data from frame "<<r+1<<" , "<<frames[r].data<<endl;
}

void SlidingWindow::goBack(int n, int r)
{
    sender(n);
    recAck(r);
    resendGb(n, r);
    cout<<"\n\t All Frames Sent Successfully...!!"<<endl;
}

void SlidingWindow::selective(int n, int r)
{
    sender(n);
    recAck(r);
    resendSr(r);
    cout<<"\n\t All Frames Sent Successfully...!!"<<endl;
}

int main()
{
    int n, r, choice;
    SlidingWindow sw;

    while(true)
    {
        cout<<"\n\t === MainMenu === \n\t\t 1. Go Back n ARQ \n\t\t 2. Selective Repea
t ARQ \n\t\t 3. Exit";
        cout<<"\n\n\t Enter Choice : ";
        cin>>choice;

        if(choice == 1)
        {
            cout<<"\n\t Enter Number of Frames : ";
            cin>>n;
            r = rand()%n;
            sw.goBack(n, r);
        }
        else if(choice == 2)
        {
            cout<<"\n\t Enter Number of Frames : ";
            cin>>n;
            r = rand()%n;
            sw.selective(n, r);
        }
        else if(choice == 3)
        {
            cout<<"\n\n\t\t\t ___ Thank You...!! ___";
            exit(0);
        }
        else
        {
            cout<<"\n\t Invalid choice...!!"<<endl;
        }
    }
}
```

**OUTPUT :-**

```
    === Main-Menu ===
       1. Go Back n ARQ
       2. Selective Repeat ARQ
       3. Exit

    Enter Choice : 1

    Enter Number of Frames : 5

    Enter data for frame 1 : 10

    Enter data for frame 2 : 20

    Enter data for frame 3 : 30

    Enter data for frame 4 : 40

    Enter data for frame 5 : 50

    The Frame Number 2 is Not Received...!!

    Resending frame from 2...

    Received Data of frame 2 , 20

    Received Data of frame 3 , 30

    Received Data of frame 4 , 40

    Received Data of frame 5 , 50

    All Frames Sent Successfully...!!
```

```
=== Main-Menu ===
   1. Go Back n ARQ
   2. Selective Repeat ARQ
   3. Exit

Enter Choice : 2

Enter Number of Frames : 5

Enter data for frame 1 : 10

Enter data for frame 2 : 20

Enter data for frame 3 : 30

Enter data for frame 4 : 40

Enter data for frame 5 : 50

The Frame Number 3 is Not Received...!!

Resending Frame Number 3...

Received Data from frame 3 , 30

All Frames Sent Successfully...!!

=== Main-Menu ===
   1. Go Back n ARQ
   2. Selective Repeat ARQ
   3. Exit

Enter Choice : 3


     ___ Thank You...!! ___
```