CN LAB

Experiment – 6

IMPLEMENTATION AND STUDY OF SLIDING WINDOW PROTOCOLS (GO-BACK-N AND SELECTIVE REPEAT)

Submitted To

Prof. Mangal Singh

Computer Networks

SIT, Pune

Submitted By

Roshan Kumar Yadav

21070126130

AIML B3

AIM

To understand and simulate Go-Back-N and Selective Repeat protocols.

THEORY

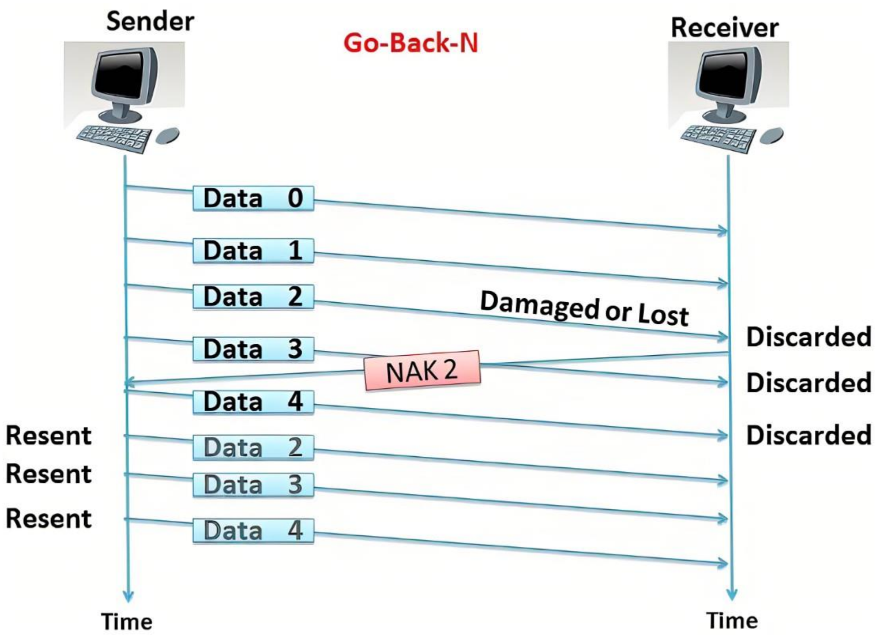
Go-Back-N Protocol

1. It's an Automatic Repeat reQuest (ARQ) protocol for retransmitting lost or corrupted frames.
2. Go-Back-N uses a sliding window approach with sender and receiver windows.
3. Sender's Perspective:
    - Sends multiple frames in sequence with unique sequence numbers.
    - Frames are sent within an allowable sequence number window.
4. Receiver's Perspective:
    - Maintains an acceptable sequence number window, discards out-of-sequence frames.
    - Sends acknowledgments (ACKs) for correctly received frames with the highest in-order frame number.
5. Sender:
    - Tracks sent frames and maintains timers for unacknowledged frames.
    - Buffers frames until acknowledgment and sends within the window size.
6. On receiving a NACK or a timeout for a specific frame, it retransmits all unacknowledged frames from that point (Go-Back-N).
7. Retransmits all frames from the point of failure onward, causing inefficiencies when multiple frames are lost.
8. Includes basic flow control to prevent receiver congestion.
9. Efficient in low-error environments, but less so with frequent errors due to unnecessary retransmissions.
10. Advantages:

- ✓ Simple to implement.
- ✓ Suitable for reliable communication with occasional errors.

11. Disadvantages:

- ➢ Inefficient in high-error environments, as it retransmits multiple frames for one loss.
- ➢ Limited in concurrent frames in transit.



Implementation

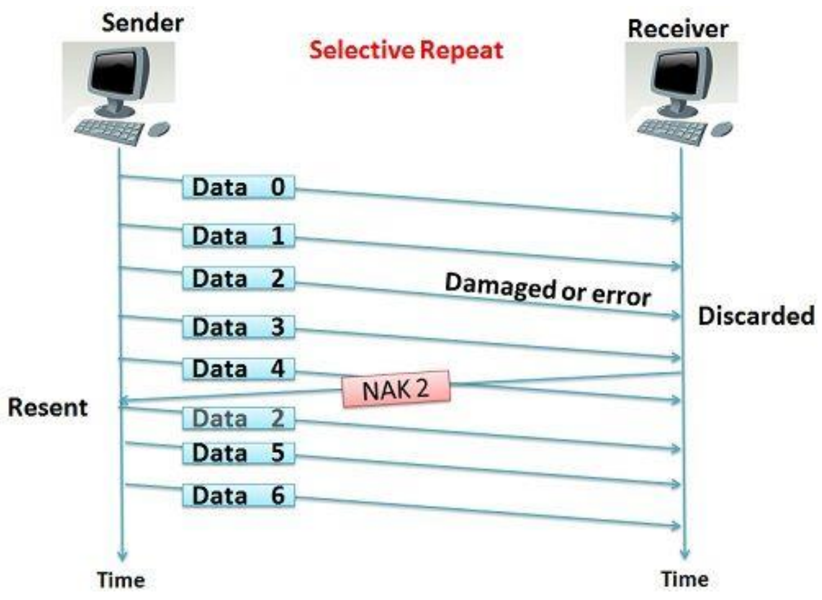
In C++

```
#include<bits/stdc++.h>
#include<ctime>
#define ll long long int
using namespace std;
void transmission(ll & i, ll & N, ll & tf, ll & tt) {
  while (i <= tf) {
    int z = 0;
    for (int k = i; k < i + N && k <= tf; k++) {
      cout << "Sending Frame " << k << "..." << endl;
      tt++;
    }
    for (int k = i; k < i + N && k <= tf; k++) {
      int f = rand() % 2;
      if (!f) {
        cout << "Acknowledgment for Frame " << k << "..." << endl;
        z++;
      } else {
        cout << "Timeout!! Frame Number : " << k << " Not Received" << endl;
        cout << "Retransmitting Window..." << endl;
        break;
      }
    }
    cout << "\n";
    i = i + z;
  }
}

int main() {
  ll tf, N, tt = 0;
  srand(time(NULL));
  cout << "Enter the Total number of frames : ";
  cin >> tf;
  cout << "Enter the Window Size : ";
  cin >> N;
  ll i = 1;
  transmission(i, N, tf, tt);
  cout << "Total number of frames which were sent and resent are : " << tt <<
    endl;
  return 0;
}
```

<u>Selective Repeat Protocol</u>

- ✓ It's a sliding window protocol with sender and receiver windows.
- ✓ Sender sends multiple frames with unique sequence numbers and maintains per-frame timers.
- ✓ Receiver discards out-of-sequence frames and sends individual acknowledgments.
- ✓ It selectively retransmits only lost or corrupted frames, improving efficiency.
- ✓ More efficient than Go-Back-N in the presence of errors.
- ✓ Implementation is more complex due to individual acknowledgment tracking.
- ✓ Provides basic flow control.
- ✓ Suited for high-error environments and variable error rates.



Implementation

In C

```
#include <stdio.h>
#include <stdlib.h>
#define WS 5                    //window size
#define N  21                   //number of frames to be send, frames are 0,1,2...N-1

int sendToReceiver(int data, int correctAck)
{
//        printf("Reciever: received data: %d, ack: %d\n", data, correctAck);
          if(rand()%3 == 0)
          {
                    return -1;                                        //negative
          }
          else
          {
                    return correctAck;                      //positive
          }
}
int main()
{
          //sender code
          int window[WS];
          int windowStart=1;

          while(windowStart<=N)
          {
                    printf("sender: window start at: %d\n", windowStart);

                    int i;

                    //initialising the window
                    //window[i] correspond to data (windowStart+i)
                    //window[i]!=0, means corresponding frames not acknowledged
                    for(i=0; i<WS; ++i)          window[i]=windowStart+i;

                    do
                    {
                      int ackn[WS];
                              for(i=0; i<WS && windowStart+i<=N; ++i)
                              {
                                        if(window[i]!=0)
                                        {
                                                  printf("Data Send: %d\n", windowStart+i);
                                                  int ack= sendToReceiver(windowStart+i, i);
                                                  ackn[i] = ack;
                                        //        printf("Sender: Acknowledgement Received: %d\n", ack+1);

                                                  if(ack!=-1)
                                                  {
                                                            window[ack]=0;
                                                  }
                                        }
                              }

                              for(i=0; i<WS; ++i){
                                if(ackn[i] == -1){
                                  printf("Reciever: sending negative acknowledgemnt\n");
                                }else{
                                  printf("Reciver: sending positive Acknowledgement\n");
                                }
                              }
```

```
                                        /*trying to find the first unacknowledged frame*/
                                        for(i=0; i<WS; ++i)
                                                if(window[i]!=0)     break;
                        }while(i!=WS && windowStart+i!=N+1);

                        /*next window starts from the first unacknowledged frame*/
                        windowStart+=WS;
        }
        return 0;
}
```

OSERVATION

Output of Go-Back-N Code Implementation

```
Enter the Total number of frames : 8
Enter the Window Size : 4
Sending Frame 1...
Sending Frame 2...
Sending Frame 3...
Sending Frame 4...
Acknowledgment for Frame 1...
Timeout!! Frame Number : 2 Not Received
Retransmitting Window...

Sending Frame 2...
Sending Frame 3...
Sending Frame 4...
Sending Frame 5...
Acknowledgment for Frame 2...
Acknowledgment for Frame 3...
Timeout!! Frame Number : 4 Not Received
Retransmitting Window...

Sending Frame 4...
Sending Frame 5...
Sending Frame 6...
Sending Frame 7...
Acknowledgment for Frame 4...
Acknowledgment for Frame 5...
Acknowledgment for Frame 6...
Timeout!! Frame Number : 7 Not Received
Retransmitting Window...

Sending Frame 7...
Sending Frame 8...
Timeout!! Frame Number : 7 Not Received
Retransmitting Window...

Sending Frame 7...
Sending Frame 8...
Acknowledgment for Frame 7...
Timeout!! Frame Number : 8 Not Received
Retransmitting Window...

Sending Frame 8...
Timeout!! Frame Number : 8 Not Received
Retransmitting Window...

Sending Frame 8...
Acknowledgment for Frame 8...

Total number of frames which were sent and resent are : 18
```

Output of Selective Repeat Code Implementation

```
sender: window start at: 1
Data Send: 1
Data Send: 2
Data Send: 3
Data Send: 4
Data Send: 5
Receiver: sending positive Acknowledgement 1
Receiver: sending positive Acknowledgement 2
Receiver: Negative Acknowledgement 3
Receiver: sending positive Acknowledgement 4
Receiver: sending positive Acknowledgement 5
Data Send: 3
Receiver: sending positive Acknowledgement 1
Receiver: sending positive Acknowledgement 2
Receiver: sending positive Acknowledgement 3
Receiver: sending positive Acknowledgement 4
Receiver: sending positive Acknowledgement 5
sender: window start at: 6
Data Send: 6
Data Send: 7
Data Send: 8
Data Send: 9
Data Send: 10
Receiver: sending positive Acknowledgement 6
Receiver: Negative Acknowledgement 7
Receiver: Negative Acknowledgement 8
Receiver: sending positive Acknowledgement 9
Receiver: sending positive Acknowledgement 10
Data Send: 7
Data Send: 8
Receiver: sending positive Acknowledgement 6
Receiver: sending positive Acknowledgement 7
Receiver: sending positive Acknowledgement 8
Receiver: sending positive Acknowledgement 9
Receiver: sending positive Acknowledgement 10
sender: window start at: 11
Data Send: 11
Data Send: 12
Data Send: 13
Data Send: 14
Data Send: 15
Receiver: sending positive Acknowledgement 11
Receiver: sending positive Acknowledgement 12
Receiver: sending positive Acknowledgement 13
Receiver: Negative Acknowledgement 14
Receiver: Negative Acknowledgement 15
Data Send: 14
Data Send: 15
Receiver: sending positive Acknowledgement 11
Receiver: sending positive Acknowledgement 12
Receiver: sending positive Acknowledgement 13
Receiver: sending positive Acknowledgement 14
Receiver: sending positive Acknowledgement 15
sender: window start at: 16
Data Send: 16
Data Send: 17
Data Send: 18
Data Send: 19
Data Send: 20
Receiver: sending positive Acknowledgement 16
Receiver: sending positive Acknowledgement 17
Receiver: Negative Acknowledgement 18
Receiver: Negative Acknowledgement 19
Receiver: sending positive Acknowledgement 20
```

```
Data Send: 18
Data Send: 19
Receiver: sending positive Acknowledgement 16
Receiver: sending positive Acknowledgement 17
Receiver: sending positive Acknowledgement 18
Receiver: sending positive Acknowledgement 19
Receiver: sending positive Acknowledgement 20
sender: window start at: 21
Data Send: 21
Receiver: sending positive Acknowledgement 21
Receiver: sending positive Acknowledgement 22
Receiver: sending positive Acknowledgement 23
Receiver: sending positive Acknowledgement 24
Receiver: sending positive Acknowledgement 25
```

SELF-ASSESSMENT

1. **Frame an algorithm for the Selective Repeat protocol as done in the above-mentioned manner for the Go-Back-N protocol.**

```
1. Initialize constants and data structures.
   - Set WS (Window Size) and N (Total number of frames).
   - Create an array 'window' to represent the sender's window.
   - Initialize 'windowStart' to 1, the start of the current window.

2. Start the sender's simulation loop:
   - While 'windowStart' is less than or equal to 'N':
     - Print "Sender: Window start at: windowStart".

3. Inside the sender's loop:
   - Initialize the 'window' array with the frames in the current window.
     - 'window[i]' corresponds to data (windowStart + i).
     - 'window[i] != 0' means that the corresponding frame is not yet acknowledged.

4. Start the sender's frame transmission loop:
   - Do the following until all frames in the current window are acknowledged:
     - Create an array 'ackn' to store acknowledgments for the frames.
     - For each frame in the current window (up to 'WS' frames or until 'N'):
       - If 'window[i]' is not 0:
         - Print "Data Send: windowStart + i".
         - Simulate sending the frame to the receiver by calling 'sendToReceiver'.
         - Update 'ackn[i]' with the returned acknowledgment (positive or negative).

5. After sending all frames in the current window, process acknowledgments:
   - For each frame in the current window:
     - If 'ackn[i]' is -1, print "Receiver: Not received windowStart + i".
     - Otherwise, print "Receiver: Sending positive Acknowledgement windowStart + i".
     - Mark the corresponding frame as acknowledged by setting 'window[ack]' to 0.

6. Find the first unacknowledged frame in the current window:
   - Iterate through the 'window' array to find the first frame with a non-zero value.
   - Continue to the next window if all frames are acknowledged, or 'windowStart + i' reaches 'N + 1'.

7. Update 'windowStart' to start the next window:
   - Increment 'windowStart' by 'WS' to begin the next window.

8. Repeat steps 2-7 until all frames have been acknowledged.

9. End of the sender's simulation.

10. Exit the program.
```

2. **Imagine a scenario where a Go-Back-N sender with a window size of 3 is communicating with a Selective Repeat receiver with a window size of 4. If the sender has sent frames 1 to 6, and the receiver has acknowledged frames 1 to 3, what actions will both the sender and receiver take?**

   Sender's Actions:

   ✓ The sender maintains a window of size 3. It has sent frames 1, 2, and 3, and they have been acknowledged by the receiver. Therefore, the sender can mark frames 1, 2, and 3 as successfully transmitted.

   ✓ The sender's window is now open for frames 4, 5, and 6. Since the receiver's window size is 4, the sender can send up to frame 6 without waiting for more acknowledgments.

   ✓ The sender will send frames 4, 5, and 6 if they are not already in transit, and it will start individual timers for each frame.

   ✓ If a frame times out (i.e., no acknowledgment is received within the timeout period), the sender will retransmit that specific frame and any unacknowledged frames that follow it.

   Receiver's Actions:

   ✓ The receiver has acknowledged frames 1, 2, and 3, and it has a window size of 4. Therefore, it is ready to accept frames 4, 5, and 6 if they arrive in order.

   ✓ The receiver will buffer frames 4, 5, and 6 as they arrive. If frame 4 arrives next, it will be stored. If frame 5 arrives next, it will also be stored. If frame 6 arrives next, it will be stored as well.

   ✓ The receiver will send individual acknowledgments for correctly received frames. If frames 4, 5, and 6 are received in order, the acknowledgments for these frames will be sent.

   ✓ If any frame arrives out of order (e.g., frame 6 arrives before frame 4), the receiver will buffer it and wait for the missing frame to arrive. Once the missing frame is received, the receiver can then send acknowledgments for all frames in order.

   ✓ If any frame is missing, the receiver will not send acknowledgments for frames beyond the missing frame until the missing frame is received.