NAME OF THE EXPERIMENT: 2.a) Simplex protocol (stop and wait protocol)

OBJECTIVE: Implement the data link layer framing method.

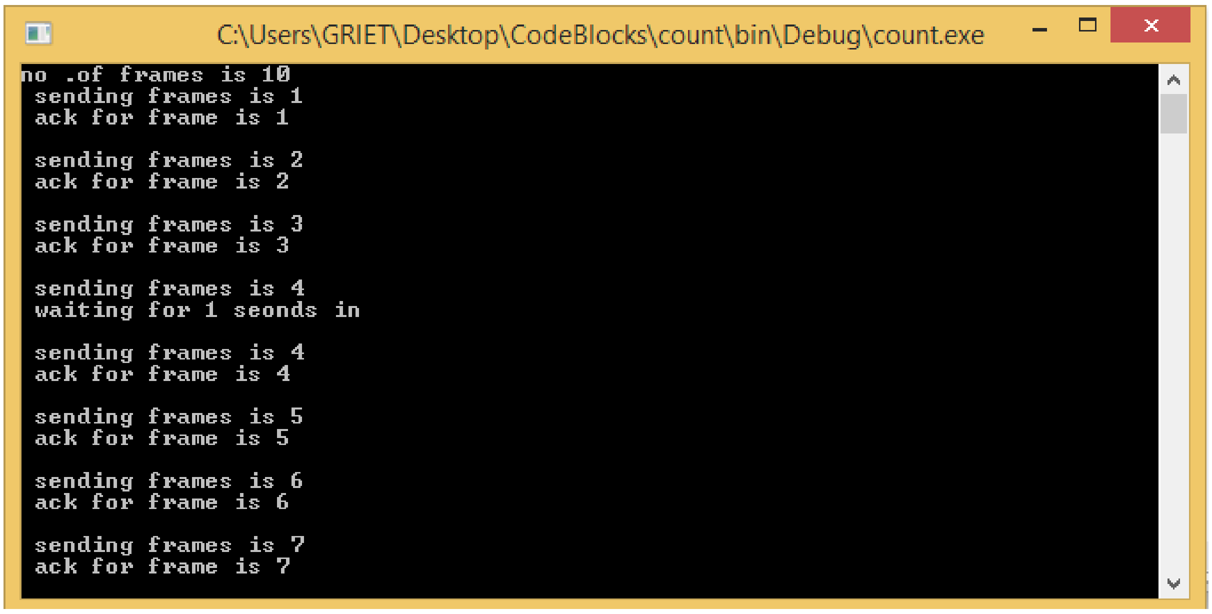
RESOURCE: Code blocks

PROGRAM LOGIC: Stop and wait protocol is a simple and reliable protocol for flow control. Stop and wait protocol is a data link layer protocol. In this protocol, the sender will not send the next packet to the receiver until the acknowledgment of the previous packet is received.

Program:

```
#include <stdio.h>
#include <stdlib.h>
void main()
{
    int i,j,n,x,x1=10,x2;
    n=10;
    i=1;j=1;
    printf("no .of frames is %d",n);
    getch();
    while(n>0)
    {
      printf("\n sending frames is %d",i);
      x=rand()%10;
      if(x%10==0)
      {
          for(x2=1;x2<2;x2++)
          {
              printf("\n waiting for %d seonds in \n ",x2);
              sleep(x2);
          }
          printf("\n sending frames is %d ",i);
          x=rand()%10;
      }
      printf("\n ack for frame is %d \n ",j);
      n=n-1;
      i++;
      j++;
    }
    printf("\n end of stop and wait protocol \n");
    getch();
}
```

Output:


C:\Users\GRIET\Desktop\CodeBlocks\count\bin\Debug\count.exe

```
no .of frames is 10
 sending frames is 1
 ack for frame is 1

 sending frames is 2
 ack for frame is 2

 sending frames is 3
 ack for frame is 3

 sending frames is 4
 waiting for 1 seonds in

 sending frames is 4
 ack for frame is 4

 sending frames is 5
 ack for frame is 5

 sending frames is 6
 ack for frame is 6

 sending frames is 7
 ack for frame is 7
```

NAME OF THE EXPERIMENT: 2.b) Sliding Window protocol(Go Back N)

OBJECTIVE: Implement the data link layer framing method.
RESOURCE: Code blocks

PROGRAM LOGIC: Go-Back-N is a data link layer protocol that uses a sliding window method. In this, if any frame is corrupted or lost, all subsequent frames have to be sent again.

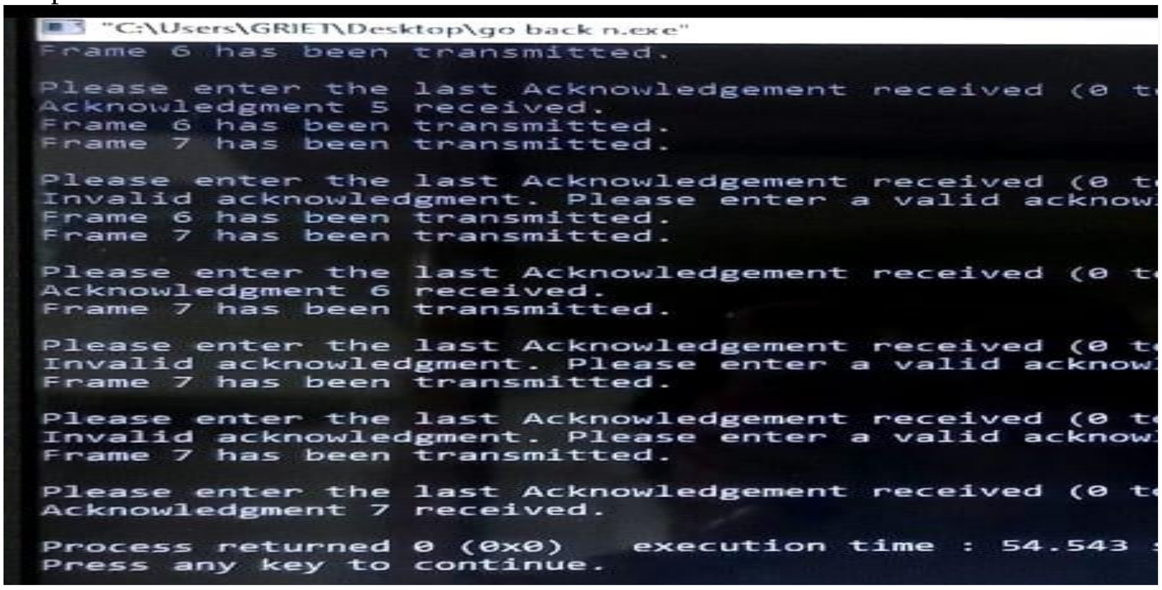
The size of the sender window is N in this protocol. For example, Go-Back-8, the size of the sender window, will be 8. The receiver window size is always 1.

If the receiver receives a corrupted frame, it cancels it. The receiver does not accept a corrupted frame. When the timer expires, the sender sends the correct frame again.

Program:

```
#include <stdio.h>

int main() {
   int total_frames, windowsize, sent = 0, ack = 0, i;
   int frame_start = 0; // Start of the window

   printf("Enter the total number of frames to be sent: ");
   scanf("%d", &total_frames);

   printf("Enter the window size: ");
   scanf("%d", &windowsize);

   if (windowsize <= 0 || total_frames <= 0 || windowsize > total_frames) {
      printf("Invalid window size or number of frames.\n");
      return 1;
   }

   while (sent < total_frames) {
      // Transmit frames within the window
      for (i = 0; i < windowsize && (frame_start + i) < total_frames; i++) {
            printf("Frame %d has been transmitted.\n", frame_start + i);
        }

        // Prompt for the last acknowledgment received
        printf("\nPlease enter the last Acknowledgement received (0 to %d): ", frame_start + i - 1);
        scanf("%d", &ack);

        if (ack < frame_start || ack > frame_start + i - 1) {
            printf("Invalid acknowledgment. Please enter a valid acknowledgment within the transmitted
window.\n");
        } else if (ack >= total_frames) {
            printf("Acknowledgment %d is beyond the total number of frames. Please enter a valid
acknowledgment.\n", ack);
        } else {
            printf("Acknowledgment %d received.\n", ack);

            // Move the window to start after the last acknowledged frame
            frame_start = ack + 1;

            // Update the `sent` variable to ensure frames are transmitted correctly
            sent = frame_start;

            // Adjust the end of the window
            int end_frame = frame_start + windowsize;
            if (end_frame > total_frames) end_frame = total_frames;

        }
    }

    printf("All frames have been transmitted and acknowledged.\n");

    return 0;
}
```

Output:



```
"C:\Users\GRIET\Desktop\go back n.exe"
Frame 6 has been transmitted.

Please enter the last Acknowledgement received (0 t
Acknowledgment 5 received.
Frame 6 has been transmitted.
Frame 7 has been transmitted.

Please enter the last Acknowledgement received (0 t
Invalid acknowledgment. Please enter a valid acknow
Frame 6 has been transmitted.
Frame 7 has been transmitted.

Please enter the last Acknowledgement received (0 t
Acknowledgment 6 received.
Frame 7 has been transmitted.

Please enter the last Acknowledgement received (0 t
Invalid acknowledgment. Please enter a valid acknow
Frame 7 has been transmitted.

Please enter the last Acknowledgement received (0 t
Invalid acknowledgment. Please enter a valid acknow
Frame 7 has been transmitted.

Please enter the last Acknowledgement received (0 t
Acknowledgment 7 received.

Process returned 0 (0x0)    execution time : 54.543 s
Press any key to continue.
```

NAME OF THE EXPERIMENT: 2.c) Sliding window protocol (selective repeat)

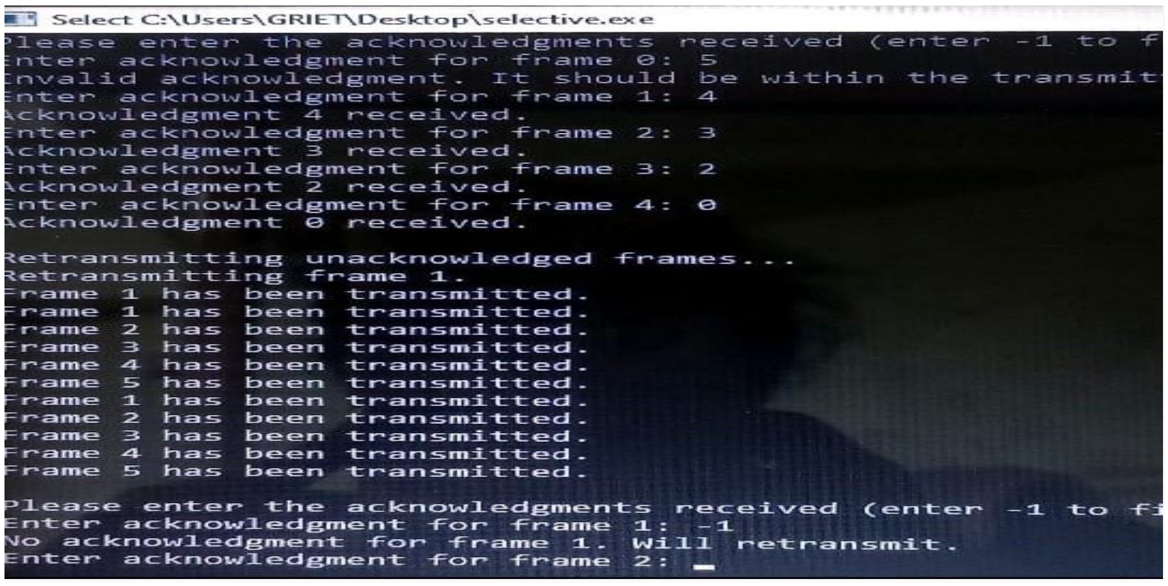
OBJECTIVE: Implement the data link layer framing method.

RESOURCE: Code blocks

PROGRAM LOGIC: Selective Repeat ARQ is also known as the Selective Repeat Automatic Repeat Request. It is a data link layer protocol that uses a sliding window method. The Go-back-N ARQ protocol works well if it has fewer errors. But if there is a lot of error in the frame, lots of bandwidth loss in sending the frames again. So, we use the Selective Repeat ARQ protocol. In this protocol, the size of the sender window is always equal to the size of the receiver window.

```
#include <stdio.h>
#include <stdbool.h>

#define MAX_FRAMES 10

void transmit_frames(int start, int end) {
for (int i = start; i < end; i++) {
printf("Frame %d has been transmitted.\n", i);
}
}

int main() {
int total_frames, window_size, i;
bool ack_received[MAX_FRAMES] = {false};
int sent_start = 0;
int ack;

// Input for total frames and window size
printf("Enter the total number of frames to be sent (max %d): ", MAX_FRAMES);
scanf("%d", &total_frames);

printf("Enter the window size: ");
scanf("%d", &window_size);

if (window_size <= 0 || total_frames <= 0 || window_size > total_frames) {
printf("Invalid window size or number of frames.\n");
return 1;
}

while (sent_start < total_frames) {
// Determine the end of the current window
int end = sent_start + window_size;
if (end > total_frames) end = total_frames;

// Transmit frames within the window
transmit_frames(sent_start, end);

// Prompt for acknowledgments
printf("\nPlease enter the acknowledgments received (enter -1 to finish):\n");
for (i = sent_start; i < end; i++) {
printf("Enter acknowledgment for frame %d: ", i);
scanf("%d", &ack);

if (ack == -1) {
printf("No acknowledgment for frame %d. Will retransmit.\n", i);
} else if (ack >= sent_start && ack < end) {
ack_received[ack] = true;
printf("Acknowledgment %d received.\n", ack);
} else {
printf("Invalid acknowledgment. It should be within the transmitted window.\n");
}
}

// Retransmit only the frames that need it
printf("\nRetransmitting unacknowledged frames...\n");
for (i = sent_start; i < end; i++) {
if (!ack_received[i]) {
printf("Retransmitting frame %d.\n", i);
transmit_frames(i, i + 1);
}
}

// Move the window forward
while (sent_start < total_frames && ack_received[sent_start]) {
sent_start++;
}

// Adjust window end to include new frames that come after the last acknowledged frame
int new_end = sent_start + window_size;
if (new_end > total_frames) new_end = total_frames;

// Transmit new frames if there are any
if (sent_start < total_frames && new_end > sent_start) {
transmit_frames(sent_start, new_end);
}
}

printf("All frames have been transmitted and acknowledged.\n");

return 0;
}
```

Output::



```
Select C:\Users\GRIET\Desktop\selective.exe
Please enter the acknowledgments received (enter -1 to f
Enter acknowledgment for frame 0: 5
Invalid acknowledgment. It should be within the transmit
Enter acknowledgment for frame 1: 4
Acknowledgment 4 received.
Enter acknowledgment for frame 2: 3
Acknowledgment 3 received.
Enter acknowledgment for frame 3: 2
Acknowledgment 2 received.
Enter acknowledgment for frame 4: 0
Acknowledgment 0 received.

Retransmitting unacknowledged frames...
Retransmitting frame 1.
Frame 1 has been transmitted.
Frame 1 has been transmitted.
Frame 2 has been transmitted.
Frame 3 has been transmitted.
Frame 4 has been transmitted.
Frame 5 has been transmitted.
Frame 1 has been transmitted.
Frame 2 has been transmitted.
Frame 3 has been transmitted.
Frame 4 has been transmitted.
Frame 5 has been transmitted.

Please enter the acknowledgments received (enter -1 to fi
Enter acknowledgment for frame 1: -1
No acknowledgment for frame 1. Will retransmit.
Enter acknowledgment for frame 2: _
```