Sliding Window Protocol

A COURSE PROJECT REPORT

By

Anubhav Mishra (RA1911029010023)

Under the guidance of

Dr.Naresh R

In partial fulfilment for the Course

of

18CSC302J - COMPUTER NETWORKS

in

BTech CSE-CN

FACULTY OF ENGINEERING AND TECHNOLOGY

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

Kattankulathur, Chenpalpattu District

NOVEMBER 2021

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

(Under Section 3 of UGC Act, 1956)

BONAFIDE CERTIFICATE

Certified that this project report " **Sliding Window Protocol**" is the bonafide work of **Anubhav Mishra (RA1911029010023)** who carried out the project work under my supervision.

SIGNATURE	**SIGNATURE**
Subject Staff	**Dr.E. Sasikala,**
Designation	**Course Cordinator**
Department	**Associate Professor,**
SRM Institute of Science and Technology	**Data Science and Business Systems**
Potheri, SRM Nagar, Kattankulathur,	SRM Institute of Science and Technology
Tamil Nadu 603203	Potheri, SRM Nagar, Kattankulathur,
	Tamil Nadu 6032

ACKNOWLEDGEMENT

We express our heartfelt thanks to our honorable **Vice Chancellor Dr. C. MUTHAMIZHCHELVAN**, for being the beacon in all our endeavors.

We would like to express my warmth of gratitude to our **Registrar Dr. S. Ponnusamy,** for his encouragement

We express our profound gratitude to our **Dean (College of Engineering and Technology) Dr. T. V.Gopal,** for bringing out novelty in all executions.

We would like to express my heartfelt thanks to Chairperson, School of Computing **Dr. Revathi Venkataraman,** for imparting confidence to complete my course project

We wish to express my sincere thanks to **Course Audit Professor Dr.M.LAKSHMI, Professor and Head, Data Science and Business Systems** and **Course Cordinator Dr.E. Sasikala, Associate Professor**, **Data Science and Business Systems** for their constant encouragement and support.

We are highly thankful to our my Course project Internal guide **Subject handling staff name , Designation , Department,** for **his/her** assistance, timely suggestion and guidance throughout the duration of this course project.

We extend my gratitude to **Student HOD name Department** and my Departmental colleagues for their Support.

Finally, we thank our parents and friends near and dear ones who directly and indirectly contributed to the successful completion of our project. Above all, I thank the almighty for showering his blessings on me to complete my Course project

TABLE OF CONTENTS

Abstract

A sliding window protocol is a feature of packet-based data transmission protocols. Sliding window protocols are used where reliable in-order delivery of packets is required, such as in the Data Link Layer (OSI model) as well as in the Transmission Control Protocol (TCP). Conceptually, each portion of the transmission (packets in most data link layers, but bytes in TCP) is assigned a unique consecutive sequence number, and the receiver uses the numbers to place received packets in the correct order, discarding duplicate packets and identifying missing ones. The problem with this is that there is no limit of the size of the sequence numbers that can be required. By placing limits on the number of packets that can be transmitted or received at any given time, a sliding window protocol allows an unlimited number of packets to be communicated using fixed-size sequence numbers. The term "window" on transmitter side represents the logical boundary of the total number of packets yet to be acknowledged by the receiver. The receiver informs the transmitter in each acknowledgment packet the current maximum receiver buffer size (window boundary). The TCP header uses a 16 bit field to report the receive window size to the sender. Therefore, the largest window that can be used is 2^16=65k bytes. In slow-start mode, the transmitter starts with low packet count and increases the number of packets in each transmission after receiving acknowledgment packets from receiver. For every ack packet received, the window slides by one packet (logically) to transmit one new packet. When the window threshold is reached, the transmitter sends one packet for one ack packet received. If the window limit is 10 packets then in slow start mode the transmitter may start transmitting one packet followed by two packets (before transmitting two packets, one packet ack has to be received), followed by three packets and so on until 10 packets. But after reaching 10 packets, further transmissions are restricted to one packet transmitted for one ack packet received. In a simulation this appears as if the window is moving by one packet distance for every ack packet received. On the receiver side also the window moves one packet for every packet received. The sliding window method ensures that traffic congestion on the network is avoided. The application layer will still be offering data for transmission to TCP without worrying about the network traffic congestion issues as the TCP on sender and receiver side implement sliding windows of packet buffer. The window size may vary dynamically depending on network traffic.

Introduction

Sliding window protocols are data link layer protocols for reliable and sequential delivery of data frames. The sliding window is also used in Transmission Control Protocol. In this protocol, multiple frames can be sent by a sender at a time before receiving an acknowledgment from the receiver. The term sliding window refers to the imaginary boxes to hold frames. Sliding window method is also known as windowing.

Working Principle in these protocols is that, the sender has a buffer called the sending window and the receiver has buffer called the receiving window. 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 $2n-1$. Consequently, the size of the sending window is $2n-1$.

Thus in order to accommodate a sending window size of $2n-1$, a n-bit sequence number is chosen. The sequence numbers are numbered as modulo-n. For example, if the sending window size is 4, then the sequence numbers will be 0, 1, 2, 3, 0, 1, 2, 3, 0, 1, and so on. The number of bits in the sequence number is 2 to generate the binary sequence 00, 01, 10, 11.

The size of the receiving window is the maximum number of frames that the receiver can accept at a time. It determines the maximum number of frames that the sender can send before receiving acknowledgment.

The Stop and Wait ARQ offers error and flow control, but may cause big performance issues as sender always waits for acknowledgement even if it has next packet ready to send. Consider a situation where you have a high bandwidth connection and propagation delay is also high (you are connected to some server in some other country through a high-speed connection), you can’t use this full speed due to limitations of stop and wait.

Sliding Window protocol handles this efficiency issue by sending more than one packet at a time with a larger sequence number. The idea is same as pipelining in architecture.

In Stop and Wait protocol, only 1 packet is transmitted onto the link and then sender waits for acknowledgement from the receiver. The problem in this setup is that efficiency is very less as we are not filling the channel with more packets after 1st packet has been put onto the link.

Within the total cycle time of Tt + 2*Tp units, we will now calculate the maximum number of packets that sender can transmit on the link before getting an acknowledgement.All the packets in the current window will be given a sequence number. Number of bits required to represent the sender window = ceil(log2(1+2*a)).

But sometimes number of bits in the protocol headers is pre-defined. Size of sequence number field in header will also determine the maximum number of packets that we can send in total cycle time. If N is the size of sequence number field in the header in bits, then we can have 2N sequence numbers.

Window Size ws = min(1+2*a, 2N)

If you want to calculate minimum bits required to represent sequence numbers/sender window, it will be ceil(log2(ws)).

Sliding Window Protocol is actually a theoretical concept in which we have only talked about what should be the sender window size (1+2a) in order to increase the efficiency of stop and wait arq. Now we will talk about the practical implementations in which we take care of what should be the size of receiver window. Practically it is implemented in two protocols namely :

Go Back N (GBN)

Selective Repeat (SR)

Selective Repeat attempts to retransmit only those packets that are actually lost (due to errors)The go-back-n protocol works well if errors are less, but if the line is poor it wastes a lot of bandwidth on retransmitted frames. An alternative strategy, the selective repeat protocol, is to allow the receiver to accept and buffer the frames following a damaged or lost one.

To guarantee lossless delivery, a unique (increasing) sequence number (ID) is attached to every message. The combination of the message's sender address and its ID identifies every message uniquely. To overcome message loss, sender and receiver agree on some start ID, e.g.

TCP connection establishment involves the sender and receiver exchanging their start sequence numbers in the SYN/ACK packets. Now the IDs of messages received from a sender S always have to monotonically increase: if message 56 was received from S, the next message has to be 57. If a gap is detected, the receiver sends a retransmission request to the sender of the message.

Upon receiving a retransmission request, the sender resends the message with the requested ID. Being able to resend old messages requires the senders to store recently sent messages for some time, usually until it is know that the receiver(s) has/have received all messages below a certain ID.

In this case the messages below this ID may be deleted. When a receiver only sends retransmission requests when a gap is detected, the mechanism is called negative acknowledgment NAK. NAK is used in systems where message loss is infrequent, so NAKs would only rarely have to be used to retransmit lost messages.

Requirement Analysis

The sliding window protocol is an established protocol in the ISO-OSI protocol stack. Sliding window protocol assumes two-way communication. It uses two types of frames first data second acknowledgment.

The basic idea of sliding window protocol is that both sender and receiver keep a ―window‖ of acknowledgement. The sender keeps the value of expected acknowledgement, while the receiver keeps the value of expected receiving frame. When it receives an acknowledgement from the receiver, the sender advances the window. When it receives the expected frame, the receiver advances the window.

The sliding window protocol has been widely used in many popular communication protocols such as TCP, HDLC and SPX. The protocol can ensure a correct data transfer over poor quality communication channels where the packets may be duplicated, lost, or re-ordered.

Sender and Receiver

In a sliding protocol, there are two main components: the sender and the receiver. The sender obtains an infinite sequence of data from the sending host. We call indivisible blocks of data in this sequence ―frames‖, and the sequence itself the ―input sequence‖. The input sequence must be transmitted to the receiver via an unreliable network. After receiving a frame via the channel, the receiver may decide to accept the frame and eventually deliver it to the receiving host. The correctness condition for a sliding window protocol says that the receiver should deliver the frames to the receiving host in the same order in which they appear in the input sequence.

Messages and Channels

In order to transmit a frame, the sender puts it In order to transmit a frame; the sender puts it into a frame message together with some additional information, and sends it to the Analysis of Sliding Window Protocol for Connected Node Kapil Kumar Bansal, Rakesh Kumar Yadav Analysis of Sliding Window Protocol for Connected Node 293 frame channel. After the receiver eventually receives the frame message from this channel, it sends an acknowledgment message for the corresponding frame back to the sender.

This acknowledgment message is transmitted via the acknowledgment channel. After receiving an acknowledgment message, the sender knows that the corresponding frame has been received by the receiver. Thus the communication between the sender and the receiver is bi-directional; the sender transmits frames to the receiver via the frame channel, and the receiver transmits acknowledgments for these frames to the sender via the acknowledgment channel.

Sequence Numbers

The sender sends the frames in the same order in which they appear in its input sequence. However, the frame channel is unreliable, so the receiver may receive these frames in a very different order. Therefore it is clear that each frame message must contain some information about the order of the corresponding frame in the input sequence.

Such additional information is called sequence number. In the sliding window protocol, instead of the exact position of the frame in the input sequence, the sender sends the remainder of this position with respect to some fixed modulus.

Sending and Receiving Windows

At any instant of time, the sender maintains a sending window of consecutive sequence numbers corresponding to frames it is permitted to send. These sequence numbers within the sending window represent frames sent but not yet acknowledged. Similarly, the receiver maintains a receiving window, corresponding to the number of out of order frames it is permitted to accept. Any frames falling outside the receiving window are discarded without comment.

Communication Procedure

In the beginning of communication between the sender and receiver, the sending window and the receiving window are empty. The sender is starting to send data frames to the receiver via the frame channel. On the other side, the receiver is waiting for receiving data frames from the network.

Whenever a new packet arrives at the sender from the network layer, the next highest sequence number according to the sending window is given, and the upper edge of the sending window is advanced by one. The sender puts the packet and the sequence number with some other control information into a frame, and then sends the frame to the frame channel.

The sliding window (windowing) technique is used by Transmission Control Protocol (TCP) to manage the flow of packets between two computers or network hosts. TCP is a core component of the Internet Protocol suite and operates at the transport layer.

The data link layer in the Open Systems Interconnection model also supports a class of protocols that use the sliding window technique. Protocols that support this technique are often referred to sliding window protocols. Sliding window protocols ensure the reliable and sequential delivery of data packets between network devices.

ARCHITECTURE AND DESIGN

Sliding Window Protocol-

1)Sliding window protocol is a flow control protocol.

2)It allows the sender to send multiple frames before needing the acknowledgements.

3)Sender slides its window on receiving the acknowledgements for the sent frames.

4)This allows the sender to send more frames.

5)It is called so because it involves sliding of sender’s window.

Maximum number of frames that sender can send without acknowledgemen = Sender window size

Optimal Window Size-

In a sliding window protocol, optimal sender window size = 1 + 2a

Derivation-

$$\boxed{\text{Efficiency } (\eta) = \frac{T_t}{T_t + 2 \times T_p}}$$

To get 100% efficiency, we must have-

$\eta = 1$

$T_t / (T_t + 2T_p) = 1$

$T_t = T_t + 2T_p$

Thus,

To get 100% efficiency, transmission time must be Tt + 2Tp instead of Tt.

This means sender must send the frames in waiting time too.

Now, let us find the maximum number of frames that can be sent in time Tt + 2Tp.

We have-

In time Tt, sender sends one frame.

Thus, In time Tt + 2Tp, sender can send (Tt + 2Tp) / Tt frames i.e. 1+2a frames.

Thus, to achieve 100% efficiency, window size of the sender must be 1+2a.

Required Sequence Numbers-

Each sending frame has to be given a unique sequence number.

Maximum number of frames that can be sent in a window = 1+2a.

So, minimum number of sequence numbers required = 1+2a.

To have 1+2a sequence numbers,

Minimum number of bits required in sequence number field = ⌈log2(1+2a)⌉

NOTE-

When minimum number of bits is asked, we take the ceil.

When maximum number of bits is asked, we take the floor.

Choosing a Window Size-

The size of the sender’s window is bounded by-

1. Receiver’s Ability-

Receiver’s ability to process the data bounds the sender window size.

If receiver can not process the data fast, sender has to slow down and not transmit the frames too fast.

2. Sequence Number Field-

Number of bits available in the sequence number field also bounds the sender window size.

If sequence number field contains n bits, then 2n sequence numbers are possible.

Thus, maximum number of frames that can be sent in one window = 2n.

For n bits in sequence number field, Sender Window Size = min (1+2a , 2n)

Sliding window protocols are data link layer protocols for reliable and sequential delivery of data frames. The sliding window is also used in Transmission Control Protocol. In this protocol, multiple frames can be sent by a sender at a time before receiving an acknowledgment from the receiver. The term sliding window refers to the imaginary boxes to hold frames. Sliding window method is also known as windowing.

WORKING PRINCIPLE

In these protocols, the sender has a buffer called the sending window and the receiver has buffer called the receiving window. 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 $2n-1$.

Consequently, the size of the sending window is $2n-1$. Thus in order to accommodate a sending window size of $2n-1$, a n-bit sequence number is chosen. The sequence numbers are numbered as modulo-n.

For example, if the sending window size is 4, then the sequence numbers will be 0, 1, 2, 3, 0, 1, 2, 3, 0, 1, and so on. The number of bits in the sequence number is 2 to generate the binary sequence 00, 01, 10, 11.

The size of the receiving window is the maximum number of frames that the receiver can accept at a time. It determines the maximum number of frames that the sender can send before receiving acknowledgment.

IMPLEMENTATION

Suppose that we have sender window and receiver window each of size 4. So, the sequence numbering of both the windows will be 0,1,2,3,0,1,2 and so on. The following diagram shows the positions of the windows after sending the frames and receiving acknowledgments.

We implemented the Sliding Window Protocol using Selective Repeat Protocol that requires the receiver to receive out of order data. Because of this requirement, the receiver must be able to buffer the packets. This protocol retransmits the data under 2 conditions:

The receiver never received the packet, or the packet is received but the Acknowledgement (ACK) of that packet never reaches the sender. We assume that the packet or the ACK is lost in transmission, hence the sender retransmits the packet after a timeout.

The receiver received the packet, but the packet is invalid because it has a wrong checksum. Then the receiver sends a Negative Acknowledgement (NAK) to notify the sender to begin retransmission immediately.
Packets and Buffers

Data sent using the sliding window protocol is divided into packets (frames) that has a fixed maximum length. Each packet has a unique sequence number that is used reorder the packets into the complete data once the transmission is complete. Since the data being sent can be quite large, the data is first divided into buffers and the transmission is done in a per-buffer basis.

In each buffer, the packets are given a sequence number that starts from zero. Hence, to disambiguate between buffers at the reciever, the buffer size must be more than twice the window size.

Sender
The sender defined a window that is marked by the Last Acknowledgement Received (LAR) and Last Frame Sent (LFS) variables.

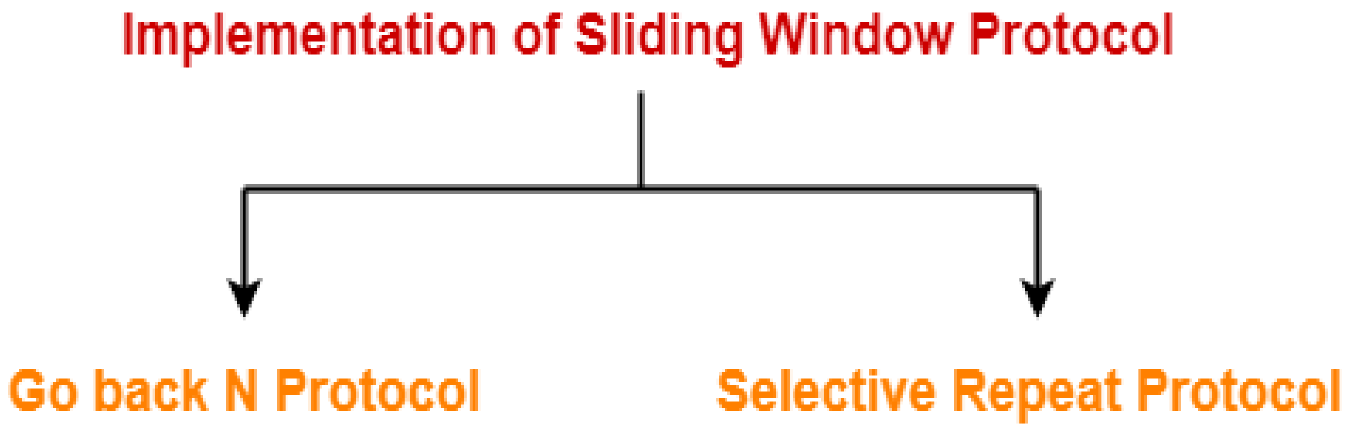
At first, the sender sends all the packets in the window and waits for each of their corresponsing ACKs. Everytime an ACK is recieved, the sender will shifts its window until the smallest un-ACK-ed sequence number is at LAR + 1. Notice that the window is not shifted everytime an ACK arrives, this is because ACKs can arrive in out of order. That is why the window is shifted only if it is possible.

Receiver

The receiver defined a window that is marked by the Last Frame Received (LFR) and Largest Acceptable Frame (LAF) variables.

When the receiver accepts a frame, it will sends and ACK of that frame if the checksum is valid, and else it will sends a NAK. Then, just like the sender, the receiver will shifts its window until the smallest unreceived sequence number is at LFR + 1.

For n bits in sequence number field, Sender Window Size = min (1+2a , 2n)
The two well known implementations of sliding window protocol are-
1 - Go back N Protocol
2 - Selective Repeat Protocol


Implementation of Sliding Window Protocol
Go back N Protocol
Selective Repeat Protocol

Efficiency-

Efficiency of any flow control protocol may be expressed as

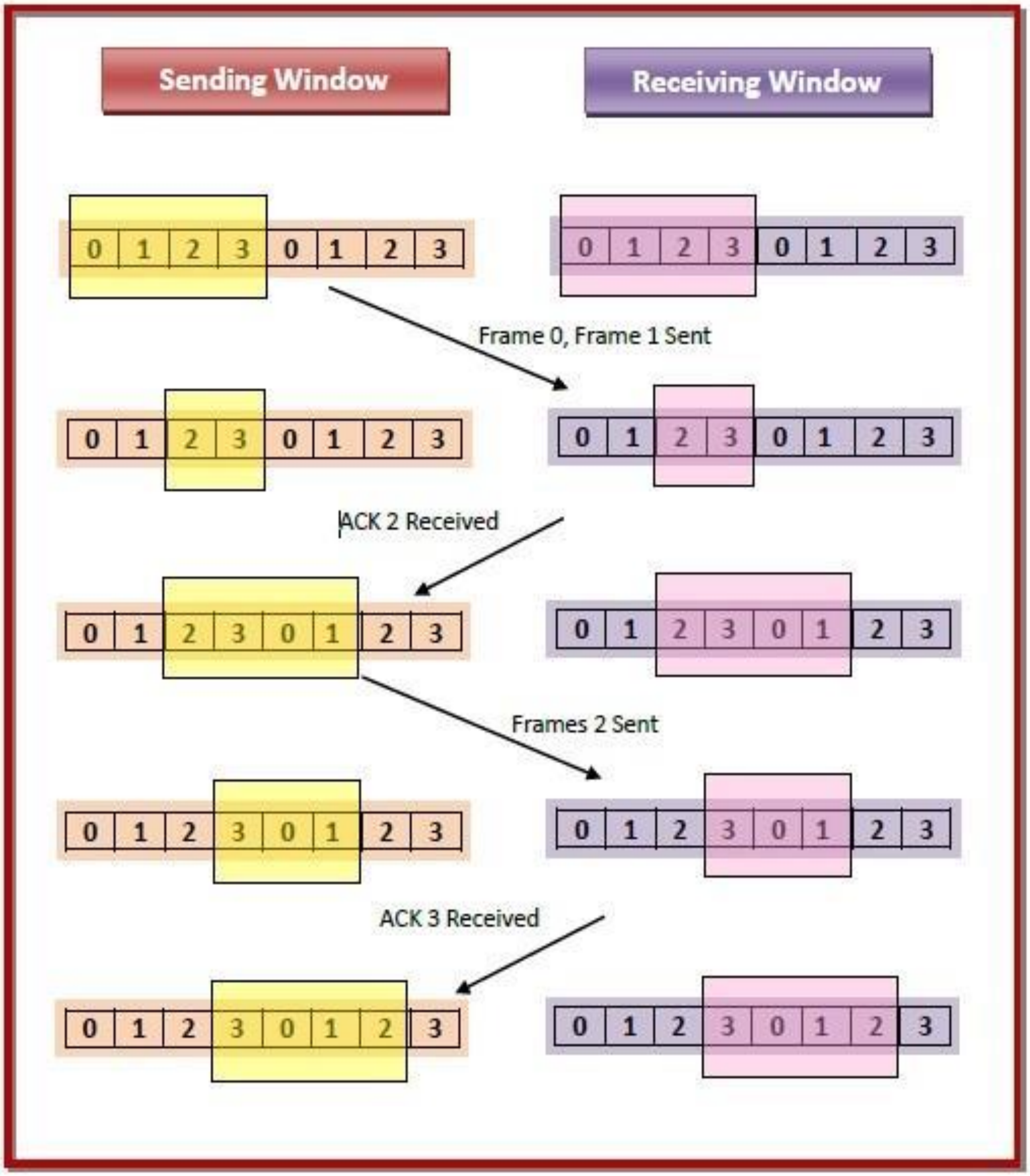
$$\boxed{\text{Efficiency } (\eta) = \frac{\text{Number of frames sent in one window}}{\text{Total number of frames that can be sent in one window}}}$$

OR

$$\boxed{\text{Efficiency } (\eta) = \frac{\text{Sender Window Size in the Protocol}}{\text{Optimal Sender Window Size}}}$$

OR

$$\boxed{\text{Efficiency } (\eta) = \frac{\text{Sender Window Size in the Protocol}}{1 + 2a}}$$


Sending Window
Receiving Window
0 1 2 3 0 1 2 3
Frame 0, Frame 1 Sent
ACK 2 Received
Frames 2 Sent
ACK 3 Received

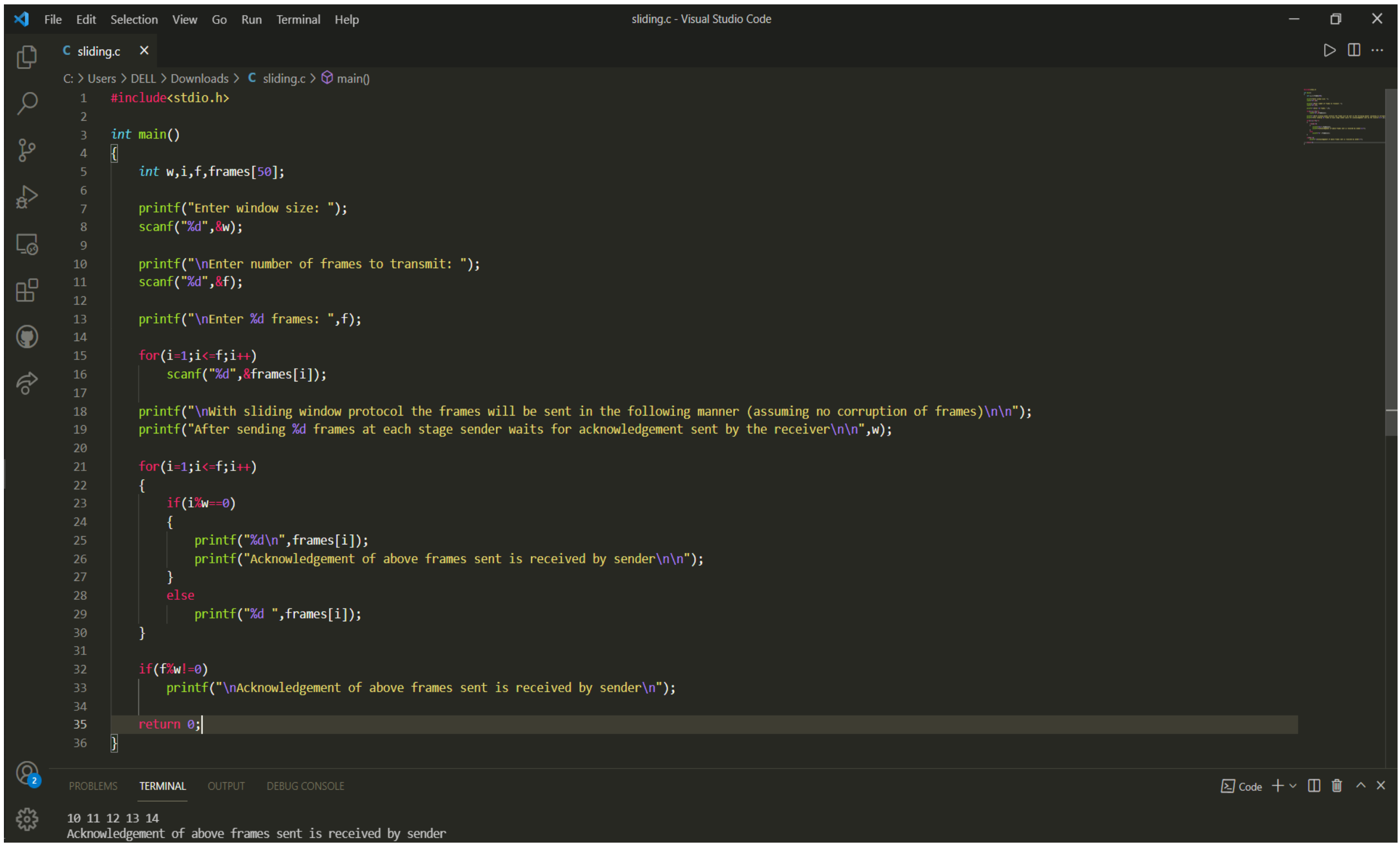
Code:

```
#include<stdio.h>

int main()
{
   int w,i,f,frames[50];

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

   printf("\nEnter number of frames to transmit: ");
   scanf("%d",&f);

   printf("\nEnter %d frames: ",f);

   for(i=1;i<=f;i++)
      scanf("%d",&frames[i]);

   printf("\nWith sliding window protocol the frames will be sent in the following
manner (assuming no corruption of frames)\n\n");
   printf("After sending %d frames at each stage sender waits for acknowledgement
sent by the receiver\n\n",w);

   for(i=1;i<=f;i++)
   {
      if(i%w==0)
      {
         printf("%d\n",frames[i]);
         printf("Acknowledgement of above frames sent is received by sender\n\n");
      }
      else
         printf("%d ",frames[i]);
   }

   if(f%w!=0)
      printf("\nAcknowledgement of above frames sent is received by sender\n");

   return 0;
}
```

EXPERIMENT RESULTS AND ANALIYSIS

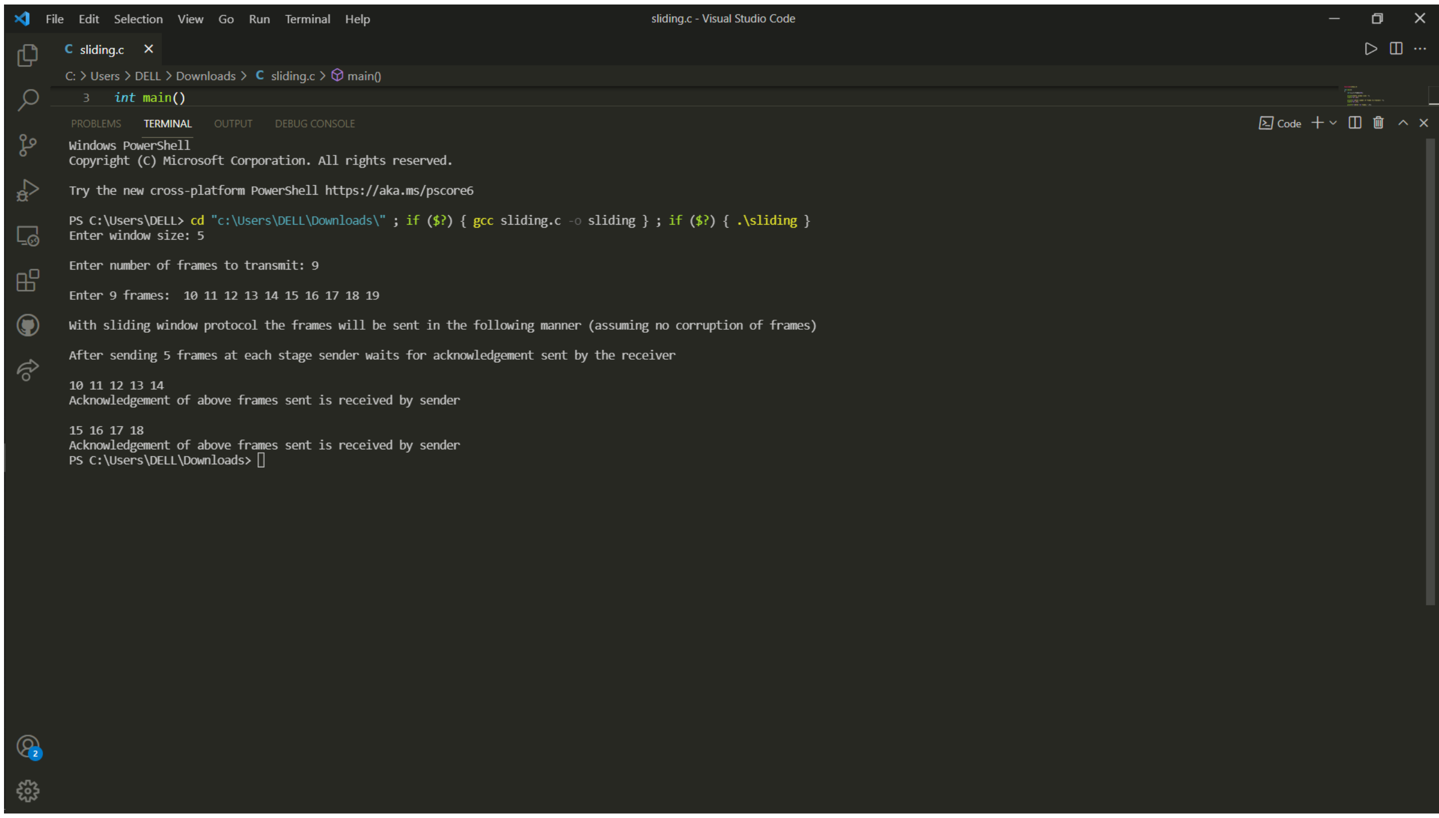
6.1. RESULTS

The Sliding Window Protocol is Successfully created and ran successfully.


File Edit Selection View Go Run Terminal Help
sliding.c - Visual Studio Code
sliding.c
C: > Users > DELL > Downloads > sliding.c > main()
PROBLEMS TERMINAL OUTPUT DEBUG CONSOLE
Code

```c
#include<stdio.h>

int main()
{
    int w,i,f,frames[50];

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

    printf("\nEnter number of frames to transmit: ");
    scanf("%d",&f);

    printf("\nEnter %d frames: ",f);

    for(i=1;i<=f;i++)
        scanf("%d",&frames[i]);

    printf("\nWith sliding window protocol the frames will be sent in the following manner (assuming no corruption of frames)\n\n");
    printf("After sending %d frames at each stage sender waits for acknowledgement sent by the receiver\n\n",w);

    for(i=1;i<=f;i++)
    {
        if(i%w==0)
        {
            printf("%d\n",frames[i]);
            printf("Acknowledgement of above frames sent is received by sender\n\n");
        }
        else
            printf("%d ",frames[i]);
    }

    if(f%w!=0)
        printf("\nAcknowledgement of above frames sent is received by sender\n");

    return 0;
}
```

```
10 11 12 13 14
Acknowledgement of above frames sent is received by sender
```



```
int main()
```

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\Users\DELL> cd "c:\Users\DELL\Downloads\" ; if ($?) { gcc sliding.c -o sliding } ; if ($?) { .\sliding }
Enter window size: 5

Enter number of frames to transmit: 9

Enter 9 frames:  10 11 12 13 14 15 16 17 18 19

With sliding window protocol the frames will be sent in the following manner (assuming no corruption of frames)

After sending 5 frames at each stage sender waits for acknowledgement sent by the receiver

10 11 12 13 14
Acknowledgement of above frames sent is received by sender

15 16 17 18
Acknowledgement of above frames sent is received by sender
PS C:\Users\DELL\Downloads> 
```

6.2 RESULT ANALYSIS

A number of parameters affect the performance of the sliding window protocol. Here we conclude some key-parameters are used in the simulation process.

A. Simulation Tolerance

Simulation tolerance defines when the simulation would end. If you want precise result, the simulation tolerance should be high, but it would take extremely long to simulate.

B. Frame Transmission Time

Frame transmission time is the time for a frame to be transmitted from source node to destination node.

C. Simulation Tolerance

The increment step tells how the simulation will proceed. This is similar to the steps in which we alter the parameters in any normal experiments to finally obtain plots.

D. Window Size

Window Size is the governing parameter of the sliding window protocol.

E. Bandwidth

It is the maximum amount of data passing the channel at a given time. Usually, it depends on the medium capacity of the communication channels itself.

Following simulation parameters are used in the simulation process.

Table 1.

Parameters	**Values**
No of Nodes	04
Simulation Tolerance	.8 sec
Final propagation Time	100 micro-sec
Frame Transmission Time	1 sec
Increment Step	1
Initial Propagation Time	1 micro-sec
Window Size	7

Table 2.

Parameters	**Values**
No of Nodes	07
Simulation Tolerance	.8 sec
Final propagation Time	100 micro-sec
Frame Transmission Time	1 sec
Increment Step	1
Initial Propagation Time	1 micro-sec
Window Size	4

Table 3.

Parameters	**Values**
No of Nodes	09
Simulation Tolerance	.8 sec
Final propagation Time	100 micro-sec
Frame Transmission Time	1 sec
Increment Step	1
Initial Propagation Time	1 micro-sec
Window Size	5

REFERENCES

- ➢ https://www.javatpoint.com/sliding-window-protocol
- ➢ https://www.gatevidyalay.com/sliding-window-protocol-flow-control/
- ➢ https://computing.dcu.ie/~humphrys/Notes/Networks/data.sliding.html
- ➢ https://ieeexplore.ieee.org/document/7754487
- ➢ https://www.seminarsonly.com/Engineering-Projects/Computer/Sliding_Window_Protocol.php