


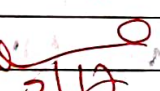
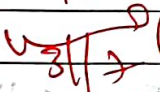
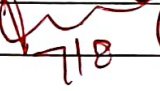


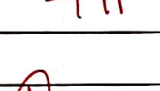
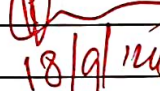
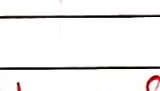


I N D E X

Harish Rayavendar.S.

220701087

Computer
Network Observation

NAME: _____ STD.: _____ SEC.: _____ ROLL NO.: _____ SUB.: _____

S. No.	Date	Title	Page No.	Teacher's Sign / Remarks
1.	13/7/2024	Basic Networking Commands.	1-8	 24/7/24
2.	24/7/2024	Study of different types of network cables.	9-12	 3/8/24
3.	31/7/2024	Study of Packet tracer tool & its user interface	13-14	 3/8/24 (8)
4.	05/8/2024	Setup and configuration of LAN network using cable (crossover).	15-17	 7/8/24 (8)
5.	05/8/2024	Packet tracer tool: Wireshark	18	 7/8/24 (9)
6.	21.8.2024	Implement error detection and correction using Hamming code - Code Comp	19-24	 21/8/24 (9)
7.	02.09.2024	Sliding Window Protocol.	25-28	 18/9/24 (9)
8a.	9.10.2024	Virtual LAN		 9/10/24
8b.	16.10.2024	Wireless LAN		
9.	18.10.2024	Subnetting in Cisco Packet Tracer		 9/11/24
10.	30.10.2024	Internetworking with router in Cisco		 9/11/24
11.	2.11.2024	Simulate static Routing.		 9/11/24

Exp No: 07

Sliding Window Protocol

Date :

Aim :

To write a program to achieve sliding window protocol for achieving packet transmission without packet loss.

Algorithm:

Step 1: Initialize variables to set # of sent packet to 0, total packets to total # of packets to be sent, set window-size (according to window size needed).

Step 2: loop until all the packets are sent,
while (sent-packets \leq total-packets).

Step 3: Wait for acknowledgement from the receiver function and if acknowledgement is received, send reset packet

Step 4: Else resend the packet.

Step 5: Do the above steps for all packets, till all packets are sent

Step 6: End.

Program:

import random

import time

```
def string-to-bits(message):  
    return "".join (message[i] for i in range(len(  
        message)))
```

```
def create-packets(bit-string, packet-size=1):  
    list = []  
    for i in range(0, len(bit-string), packet-size):  
        list.append(bit-string[i:i+packet-size])  
    return list
```

```
def sender(window-size, packets):  
    sent-packets = 0  
    total-packets = len(packets)  
    while sent-packets < total-packets:  
        print(f"sender: Sending packet {  
            sent-packets + 1} : {packets[sent-packets]}")  
        ack-received = receiver(sent-packets,  
            packets[sent-packets])  
        if ack-received == -1:  
            print("sender: Timeout, Resending  
                packet")
```

```
else  
    print (f "Sender: Acknowledgement received")  
    for packet in sent_packets + 1:  
        sent_packets += 1
```

```
def receiver (packet_num, packet_data):  
    time.sleep(0.5)  
    if random.random() < 0.4:  
        print(f "Receiver: Packet {packet_num} + 1 lost!")  
        return -1  
    else:  
        print(f "Receives: Packet {packet_num + 1} received: {packet_data}")  
        return packet_num.
```

```
message = input("Enter the message")  
bit_string = string_to_bits(message)  
packets = create_packets(bit_string)
```

```
print(f "Original Message: {message}")  
print(f "Bit string: {bit_string}")  
print(f "Packets: {packets}")  
window_size = input("Enter the window size")  
sender(window_size, packets)
```

Output:

Enter the (message: Hello.

Enter the window size: 2

Packets: ['H', 'e', 'l', 'l', 'o']

Sender: sending packets: 'H' (1st sent)

Sender: sending packets: 'e' (2nd sent)

Receiver: 1 packet received

Sender: Acknowledgement received for packet 1.

Receiver: 2 packet received

Sender: Acknowledgement received for packet 2.

Sender: sending packets: 'l' (3rd sent)

Sender: sending packets: 'l' (4th sent)

Receiver: 3 packet received.

Receiver: 4 packet received.

Code terminated successfully.

(Result)

This the program for sliding window protocol
is successfully executed & verified.

18/9/24