06/09/2024

Experiment No.: 07 Flow control at Data Link Layer

Aim:
Write a program to implement flow control at data link layer using sliding window protocol. Simulate the flow of frames from one node to another.

Create a sender program with following features.

1. Input window size from the user.
2. Input a Text Message from the user.
3. Consider 1 character per frame.
4. Create a field of frames with the following fields [Field Frame no, data]
5. Send the frames. [Print the output on screen and save it in a file called sender_Buffer]
6. Wait for the acknowledgement from the receiver. [Induce delay in the program]
7. Reader a file called Receiver_Buffer.
8. Check ACK field for the Acknowledgement number.
9. If the Acknowledgement number is as expected, send new set of frames accordingly, else if NACK is received, resend the frames accordingly.

Create a receiver file with following features.

1. Reader a file called sender_Buffer.
2. Check the frame no.
3. If the frame no are as expected, write the appropriate ACK no. in the Receiver_Buffer file. Else write NACK no. in the Receiver_Buffer file.

Program:

sender.py:

```
import time
import random

def sender (win
    frames=[]
    # Prepare
    for i, ch
        fram
    #Write f
    with ope
        for

    start=0
    while s
        # Get
        window
        print (

        # Write
        with

        time.

        # Read
        try:
            with

        #c
        if
```

ontrol at Data Link

plement flow control
sliding window
of frames from one

th following features:
e user.
the user.
ne.
ith the following

tput on screen and
Buffer]
om the receiver.

Buffer.
edgement number.
's as expected,
ingly, else if
mes accordingly.
wing features:
s.

d, write the
er_Buffer file.
er_Buffer file.

Program:

sender.py:

```
import time
import random
def sender(window_size, text_message):
    frames=[]
    # Prepare frames from the text message
    for i, char in enumerate(text_message):
        frames.append([i, char])
    #Write frames to sender_Buffer.txt
    with open("sender_Buffer.txt", "w") as file:
        for frame in frames:
            file.write(f"{frame[0]},{frame[1]}\n")
    start=0
    while start < len(frames):
        # Get the current window of frame to send
        window = frames[start: start + window_size]
        print(f"sending frames: {window}")
        # Write the window to sender_Buffer.txt
        with open("sender_Buffer.txt", "w") as file:
            for frame in window:
                file.write(f"{frame[0]},{frame[1]}\n")
        time.sleep(1)
    # Read the acknowledgment from Receiver_Buffer.txt
    try:
        with open("Receiver_Buffer.txt", "r") as file:
            ack_line = file.readline().strip()
        #check if ack_line is valid
        if ack_line:
            ack_no = int(ack_line.split(",")[0])
            print(f"ACK received for frame {ack_no}")
```

```
            if ack_no >= start:
                start = ack_no + 1
            else:
                raise ValueError("Empty acknowledgment line")
        except (ValueError, IndexError):
            print(f"Invalid or empty ack. line, resending
                frames starting from {start}")
    print("All frames sent successfully!")

# Example usage
window_size = int(input("Enter window size: "))
text_message = input("Enter the text message: ")
sender(window_size, text_message)
```

Receiver_Buffer.py:

```
import time

def read_sender_buffer(filename="sender_Buffer.txt"):
    with open(filename, 'r') as f:
        frames = f.readlines()
    # Parse each line assuming the format is
"frame_no", character"
    passed_frame = []
    for line in frames:
        if line.strip():
            try:
                parts = line.strip().split(",")
                frame_no = int(parts[0])
                char = parts[1]
                passed_frames.append(frame_no, cha
            except (IndexError, ValueError):
                print(f"skipping malformed line:
                    {line}")
    return passed_frames
```

```
def write_receiver_bu
    with open(file
        for ack
            f.wri
def receiver():
    expected_frame
    while True:
        frames =
        print(f"
        ack_list
        for fram
            if

            els

            write

            time

# Main driver
if __name__ =
    receiver()
```

```
def write_receiver_buffer(ack_list, filename=
                          'Receiver_Buffer.txt'):
    with open(filename, 'w') as b:
        for ack in ack_list:
            f.write(f"{ack}, Ack In")
def receiver():
    expected_frame_no = 0
    while True:
        frames = read_sender_buffer()
        print(f"Received frames: {frames}")
        ack_list = []
        for frame_no, data in frames:
            if frame_no == expected_frame_no:
                print(f"Frame {frame_no}
                      received successfully")
                ack_list.append(expected_frame_
                expected_frame_no += 1
            else:
                print(f"Frame {frame_no} out of
                      order, expecting frames {expected_
                      frame_no}.")
                ack_list.append(expected_frame_no-
        write_receiver_buffer(ack_list)
        time.sleep(2)
# Main driver
if __name__ = "__main__":
    receiver()
```

Output:

Sender:

Enter window size: 2

Enter the text message: Hello

Receiver:

Received frames: [(0, 'H'), (1, 'e')]

Frame 0 received successfully

Frame 1 received successfully

Received frames: [(0, 'H'), (1, 'e')]

Frame 0 out of order, expecting frame 2

Frame 1 out of order, expecting frame 2.

:

:

:

Result:

Thus, the program for Sliding Window Protocol is successfully executed and the output is verified.

Exp No.: 8

a) Aim:

Simula

using cis

Procedure:

1. Create

4 PCs as

2. Check

(Optional)

3. Assign

PC0

PC1

PC2

PC3

4. Right

Run th

> enal

conf

vlan

nam

exit

v la

nam

exit

inte

sui

exit