CZ3006 NET CENTRIC COMPUTING ASSIGNMENT 1

Sharma Shantanu
U1622895F

Lab Group: TS2

1 Summary

The main purpose of this lab is to learn the network protocol hierarchy and to handle ow control as well as error control. The main content is to build a sliding window protocol in the provided communication system. The implemented protocol ensures all the required features listed below. The implementation can handle up to quality level 3 of the Network Simulator, which allows transmitting loose and damaged frames.

a. Full-duplex data communication.

b. In-order delivery of packets to the network-layer.

c. Selective repeat retransmission strategy.

d. Synchronization with the network-layer by granting credits.

e. Negative acknowledgement.

f. Separate acknowledgement when the reverse traffic is light or none.

2 Approaches

2.1 Full-duplex data communication

Motivation to implement full-duplex data transmission arises from the necessity to transmit data bi-directionally. It can be achieved by having two separate simplex data channels. However, two separate physical circuits are required to implement them. It is not the ideal case since it leads to waste of bandwidth. The answer to overcome this bottleneck is to utilize a single circuit for transmitting data in both directions. In this assignment, the above idea is realized by having both the sender and receiver in a Single Protocol 6 function as shown below:

```
    while(true) {

      wait_for_event(event);

      switch (event.type) {

        case (PEvent.NETWORK_LAYER_READY):  // transmit frame
          nbuffered++;
          from_network_layer(out_buf[next_frame_to_send % NR_BUFS])
          send_frame(PFrame.DATA, next_frame_to_send, frame_exp, out_buf);
          // slide window
          next_frame_to_send = (next_frame_to_send + 1) % (MAX_SEQ + 1);
          break;

        case (PEvent.FRAME_ARRIVAL):      // fetch frame
          from_physical_layer(r);
          ...
```

The technique of piggybacking is used to let the acknowledgement of the current received frame to piggybacked on to next outgoing frame. So that the data channel could have a better utilization. The code below shows the implementation.

```
public void send_frame(int frame_kind, int frame_nr, int frame_exp, Packet buffer
  []){
  ...
  s.ack = (frame_exp + MAX_SEQ) % (MAX_SEQ + 1);
  if (frame_kind == PFrame.NAK)  {
    no_nak = false;
  }
  to_physical_layer(s);        // transmit the frame
  ...
}
```

2.2 In-order delivery of packets to the network-layer

SWP allows the frames transmitted to the data link layer in a different order, while it still ensures that the packets sent to network layer are in order. This is realized by using sequence number. Those frames with higher sequence number cannot be delivered to the network layer unless all the frames with lower sequence number have been transmitted successfully. The code below shows the implementation.

```
    //if it's seq is btw the receiver's window
    // store the incoming frame into the buffer
    if (between(frame_exp, r.seq, too_far) && (arrived[r.seq % NR_BUFS] == false)){
        // frames may be accepted in any order
        arrived[r.seq % NR_BUFS] = true;    // mark buffer as full
        in_buf[r.seq % NR_BUFS] = r.info;   // insert data into buffer

        // up to the next not received frame
        while (arrived[frame_exp % NR_BUFS]){
            // pass frames and advance window
            to_network_layer(in_buf[frame_exp % NR_BUFS]);
            no_nak = true;  // allow the protocol to receive NAK
            arrived[frame_exp % NR_BUFS] = false;
            // advance lower edge of receiver's window
            frame_exp = (frame_exp + 1) % (MAX_SEQ + 1);
            // advance upper edge of receiver's window
            too_far = (too_far + 1) % (MAX_SEQ + 1);
            start_ack_timer();
        }
    }
```

2.3 Selective repeat retransmission strategy

Unlike *go back n,* the selective repeat retransmission strategy only requires for the retransmission of the damaged or lost frames, rather than discarding any other subsequent correct frames. The code below shows the implementation.

```
case (PEvent.FRAME_ARRIVAL):
  from_physical_layer(r);     // fetch frame
  if (PFrame.KIND[r.kind].equals("DATA")){
      // send NAK if it's not the expected frame
      if ((r.seq != frame_exp) && no_nak)
          send_frame(PFrame.NAK, 0, frame_exp, out_buf);
      else
          start_ack_timer();
      // up to the next not received frame
      while (arrived[frame_exp % NR_BUFS]){
          // pass frames and advance window
              ...
        }
        ...
  case (PEvent.CKSUM_ERR):
      if (no_nak)
          // damaged frame, so send NAK
          send_frame(PFrame.NAK, 0, frame_exp, out_buf);
      break;
```

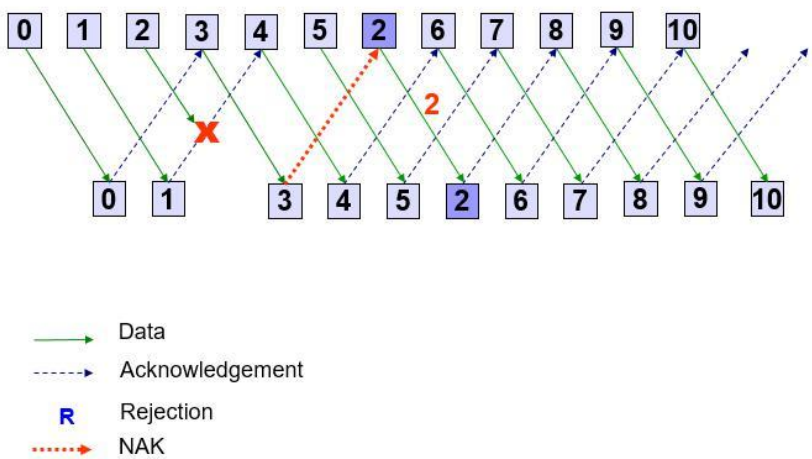

Figure: Selective Repeat Retransmission Strategy

2.4 Synchronization with the network-layer by granting credit

At first, the credit granted to network layer is equal to the window size of the receiver. Once the sender is acknowledged by a successful send, one credit is grand to the network layer. Otherwise, the network layer is enabled for sending data. The code below shows the implementation.

```
enable_network_layer(NR_BUFS);
...
while (between(ack_exp, r.ack, next_frame_to_send)){
  nbuffered --;
  // frame arrive intact so stop the timers
  stop_timer(ack_exp % NR_BUFS);
  // advance the lower edge of the sender's window
  ack_exp = (ack_exp + 1) % (MAX_SEQ + 1);
  enable_network_layer(1);     // get credit
}
```

2.5 Negative acknowledgement

When the receiver receives a damaged frame or an unexpected error, the receiver would send back a negative acknowledgement instead of the normal one, to inform the sender to resend the corresponding frame. The code below shows the implementation.

```
  // send NAK if it's not the expected frame
  if ((r.seq != frame_exp) && no_nak)
    send_frame(PFrame.NAK, 0, frame_exp, out_buf);

    ...

  case (PEvent.CKSUM_ERR):
    if (no_nak)
      // damaged frame, so send NAK
      send_frame(PFrame.NAK, 0, frame_exp, out_buf);
      break;
```

2.6 Separate acknowledgement when the reverse traffic is light or none

Sometimes, the sender might wait until timeout for the acknowledgement. This is inefficient. A better way to resolve this problem is to introduce a timer for the acknowledgement. When a frame is received successfully, start the timer. When the timer times out, transmit the acknowledgement again. The code below shows the implementation.

```
  case (PEvent.FRAME_ARRIVAL):
    from_physical_layer(r);     // fetch frame
      if (PFrame.KIND[r.kind].equals("DATA")){
        // send NAK if it's not the expected frame
        if ((r.seq != frame_exp) && no_nak)
          send_frame(PFrame.NAK, 0, frame_exp, out_buf);
        else     // start the timer for acknowledgement
          start_ack_timer();
...
  case (PEvent.TIMEOUT):
    // the sender doesnt receive any ack for the data, so resend the data
    send_frame(PFrame.DATA, oldest_frame, frame_exp, out_buf);
  break;
```

3 Source Code

The following is the source code for the sliding window protocol implementation - SWP.java.

```java
import java.util.Timer;
import java.util.TimerTask;

/*=================================================================*
 *   File: SWP.java                                                *
 *                                                                 *
 *   This class implements the sliding window protocol             *
 *   Used by VMach class                  *
 *   Uses the following classes: SWE, Packet, PFrame, PEvent,      *
 *                                                                 *
 *   Author: Professor SUN Chengzheng                              *
 *           School of Computer Engineering                        *
 *           Nanyang Technological University                      *
 *           Singapore 639798                                      *
 *=================================================================*/

public class SWP {

/*=============================================================================*
 the following are provided, do not change them!!
 *=============================================================================*/
   //the following are protocol constants.
   public static final int MAX_SEQ = 7;
   public static final int NR_BUFS = (MAX_SEQ + 1)/2;

   // the following are protocol variables
   private int oldest_frame = 0;
   private PEvent event = new PEvent();
   private Packet out_buf[] = new Packet[NR_BUFS];

   //the following are used for simulation purpose only
   private SWE swe = null;
   private String sid = null;

   //Constructor
   public SWP(SWE sw, String s){
      swe = sw;
      sid = s;
   }

   //the following methods are all protocol related
   private void init(){
      for (int i = 0; i < NR_BUFS; i++){
       out_buf[i] = new Packet();
      }
   }

   private void wait_for_event(PEvent e){
      swe.wait_for_event(e); //may be blocked
      oldest_frame = e.seq;  //set timeout frame seq
   }

   private void enable_network_layer(int nr_of_bufs) {
   //network layer is permitted to send if credit is available
       swe.grant_credit(nr_of_bufs);
   }

   private void from_network_layer(Packet p) {
      swe.from_network_layer(p);
   }

   private void to_network_layer(Packet packet) {
       swe.to_network_layer(packet);
   }

   private void to_physical_layer(PFrame fm)  {
      System.out.println("SWP: Sending frame: seq = " + fm.seq +
          " ack = " + fm.ack + " kind = " +
          PFrame.KIND[fm.kind] + " info = " + fm.info.data );
      System.out.flush();
      swe.to_physical_layer(fm);
```

```
  }

  private void from_physical_layer(PFrame fm) {
     PFrame fm1 = swe.from_physical_layer();
 fm.kind = fm1.kind;
 fm.seq = fm1.seq;
 fm.ack = fm1.ack;
 fm.info = fm1.info;
  }


/*==============================================================================
 implement your Protocol Variables and Methods below:
 ==============================================================================*/

 private boolean no_nak = true;
 private Timer[] timer = new Timer[NR_BUFS];
 private Timer ack_timer = new Timer();

 public boolean between(int a, int b, int c){
   return ((a <= b) && (b < c)) || ((c < a) && (a <= b)) || ((b < c) && (c < a));
 }

 public void send_frame(int frame_kind, int frame_nr, int frame_exp, Packet buffer
   []){
   // create a new frame for outbound frame
   PFrame s = new PFrame();
   // define the kind of this frame
   s.kind = frame_kind;
   if (frame_kind == PFrame.DATA){
     s.info = buffer[frame_nr % NR_BUFS];
   }
   s.seq = frame_nr;
   s.ack = (frame_exp + MAX_SEQ) % (MAX_SEQ + 1);
   if (frame_kind == PFrame.NAK)  {
     no_nak = false;
   }
   to_physical_layer(s);       // transmit the frame
   if (frame_kind == PFrame.DATA)  {
     start_timer(frame_nr);
   }
   stop_ack_timer();
 }

 public void protocol6() {
   // outgoing frame's ack number from the inbound data
   int ack_exp = 0;                // lower edge of the sender's window
   // expected frame's seq from the inbound data
   int frame_exp = 0;              // lower edge of the receier's window
   int next_frame_to_send = 0;     // upper edge of the sender's window
   int too_far = NR_BUFS;          // upper edge of the receiver's window
   PFrame r = new PFrame();        // frame for receiving input
   Packet in_buf[] = new Packet[NR_BUFS];     // buffer for inbound data
   boolean arrived[] = new boolean[NR_BUFS];  // arrive or not
   int nbuffered = 0;

   enable_network_layer(NR_BUFS);

   for (int i = 0; i < NR_BUFS; i++){
     arrived[i] = false;           // nothing arrives at first
     in_buf[i] = new Packet();     // initialization of in_buff
   }

   init();                         // initialization of out_buff

```

```
    while(true) {

      wait_for_event(event);

      switch (event.type) {

        case (PEvent.NETWORK_LAYER_READY):
          nbuffered++;
          from_network_layer(out_buf[next_frame_to_send % NR_BUFS]);          // fetch data
          send_frame(PFrame.DATA, next_frame_to_send, frame_exp, out_buf);  // send data
          next_frame_to_send = (next_frame_to_send + 1) % (MAX_SEQ + 1);     // slide window
          break;

        case (PEvent.FRAME_ARRIVAL):
          from_physical_layer(r);     // fetch frame
          if (PFrame.KIND[r.kind].equals("DATA")){
            // send NAK if it's not the expected frame
            if ((r.seq != frame_exp) && no_nak)
              send_frame(PFrame.NAK, 0, frame_exp, out_buf);
            // start the timer for acknowledgement, in case there is no outgoing frame that can be piggybacked
            else
              start_ack_timer();

            // store the incoming frame into the buffer if it's seq is btw the receiver's window
            if (between(frame_exp, r.seq, too_far) && (arrived[r.seq % NR_BUFS] == false)){
              // frames may be accepted in any order
              arrived[r.seq % NR_BUFS] = true;   // mark buffer as full
              in_buf[r.seq % NR_BUFS] = r.info;  // insert data into buffer

              // up to the next not received frame
              while (arrived[frame_exp % NR_BUFS]){
                // pass frames and advance window
                to_network_layer(in_buf[frame_exp % NR_BUFS]);
                no_nak = true;  // allow the protocol to receive NAK
                arrived[frame_exp % NR_BUFS] = false;
                frame_exp = (frame_exp + 1) % (MAX_SEQ + 1);  // advance lower edge of receiver's window
                too_far = (too_far + 1) % (MAX_SEQ + 1);      // advance upper edge of receiver's window
                start_ack_timer();
              }
            }

          }

          //  if receive a NAK signal
          if (PFrame.KIND[r.kind].equals("NAK") && between(ack_exp, (r.ack+1)%(MAX_SEQ+1), next_frame_to_send)){
            // resend the frame
            send_frame(PFrame.DATA, (r.ack + 1) % (MAX_SEQ + 1), frame_exp, out_buf);
          }
          while (between(ack_exp, r.ack, next_frame_to_send)){
            nbuffered --;
            stop_timer(ack_exp % NR_BUFS);    // frame arrive intact so stop the timers
            ack_exp = (ack_exp + 1) % (MAX_SEQ + 1);  // advance the lower edge of the sender's window
            enable_network_layer(1);          // get credit
```

```
          }
          break;

        case (PEvent.CKSUM_ERR):
          if (no_nak)
            // damaged frame, so send NAK
            send_frame(PFrame.NAK, 0, frame_exp, out_buf);
          break;

        case (PEvent.TIMEOUT):
          // the sender doesnt receive any ack for the data, so resend the data
          send_frame(PFrame.DATA, oldest_frame, frame_exp, out_buf);
          break;

        case (PEvent.ACK_TIMEOUT):
          // ack timer expired, send ack again
          send_frame(PFrame.ACK, 0, frame_exp, out_buf);
          break;

        default:
          System.out.println("SWP: undefined event type = " + event.type);
          System.out.flush();
      } // end of switch
    }
  }

/* Note: when start_timer() and stop_timer() are called,
   the "seq" parameter must be the sequence number, rather
   than the index of the timer array,
   of the frame associated with this timer,
  */

  private void start_timer(int seq) {
     // stop the previous indicated timer
     stop_timer(seq);
     // create new timer for sending frames
     timer[seq % NR_BUFS] = new Timer();
     // schedule the task for execution after 500 ms
     timer[seq % NR_BUFS].schedule(new FrameTask(seq), 500);
  }

  private void stop_timer(int seq) {
     try{
       timer[seq % NR_BUFS].cancel();
     } catch(Exception e) {}
  }

  private void start_ack_timer( ) {
     stop_ack_timer();
     ack_timer = new Timer();
     ack_timer.schedule(new AckTask(), 300);
  }

  private void stop_ack_timer() {
     try{
       ack_timer.cancel();
     } catch(Exception e) {}
  }

  class AckTask extends TimerTask {
     @Override
     public void run(){
       swe.generate_acktimeout_event();
     }
  }
```

4 Testing

4.1 Running Progress

The protocol was tested on all the three quality levels and all the testing were passed. The following images shows the running process for the two virtual machines under quality level 3

```
└$ java VMach 1
VMach is making a connection with NetSim...
VMach(52753) <===> NetSim(ZWLori-MAC.local/10.27.156.11:54321)
SWP: Sending frame: seq = 0 ack = 7 kind = DATA info = 0        this is  a test from site 1
SWP: Sending frame: seq = 1 ack = 7 kind = DATA info = 1        the 2nd line
SWP: Sending frame: seq = 2 ack = 7 kind = DATA info = 2        the 3rd line
SWP: Sending frame: seq = 3 ack = 7 kind = DATA info = 3        the 4th line
SWP: Sending frame: seq = 0 ack = 7 kind = NAK info =
SWP: Sending frame: seq = 0 ack = 7 kind = DATA info = 0        this is  a test from site 1
SWP: Sending frame: seq = 1 ack = 7 kind = DATA info = 1        the 2nd line
SWP: Sending frame: seq = 2 ack = 7 kind = DATA info = 2        the 3rd line
SWP: Sending frame: seq = 3 ack = 7 kind = DATA info = 3        the 4th line
SWP: Sending frame: seq = 0 ack = 7 kind = DATA info = 0        this is  a test from site 1
SWP: Sending frame: seq = 0 ack = 7 kind = ACK info =
SWP: Sending frame: seq = 4 ack = 7 kind = DATA info = 4        the 5th line
SWP: Sending frame: seq = 5 ack = 7 kind = DATA info = 5        the 6th line
SWP: Sending frame: seq = 6 ack = 7 kind = DATA info = 6        the 7th line
SWP: Sending frame: seq = 3 ack = 7 kind = DATA info = 3        the 4th line
SWP: Sending frame: seq = 7 ack = 7 kind = DATA info = 7        the 8th line
SWP: Sending frame: seq = 0 ack = 7 kind = DATA info = 8        the 9th line
SWP: Sending frame: seq = 1 ack = 7 kind = DATA info = 9        the 10th line
SWP: Sending frame: seq = 6 ack = 7 kind = DATA info = 6        the 7th line
SWP: Sending frame: seq = 7 ack = 7 kind = DATA info = 7        the 8th line
SWP: Sending frame: seq = 0 ack = 7 kind = DATA info = 8        the 9th line
SWP: Sending frame: seq = 1 ack = 7 kind = DATA info = 9        the 10th line
SWP: Sending frame: seq = 0 ack = 3 kind = ACK info =
SWP: Sending frame: seq = 6 ack = 3 kind = DATA info = 6        the 7th line
SWP: Sending frame: seq = 0 ack = 3 kind = NAK info =
SWP: Sending frame: seq = 2 ack = 3 kind = DATA info = 10       the 11th line
SWP: Sending frame: seq = 3 ack = 3 kind = DATA info = 11       the 12th line
SWP: Sending frame: seq = 0 ack = 3 kind = DATA info = 8        the 9th line
SWP: Sending frame: seq = 1 ack = 3 kind = DATA info = 9        the 10th line
SWP: Sending frame: seq = 4 ack = 3 kind = DATA info = 12       the 13th line
SWP: Sending frame: seq = 5 ack = 3 kind = DATA info = 13       the 14th line
SWP: Sending frame: seq = 6 ack = 3 kind = DATA info = 14       the 15th line
SWP: Sending frame: seq = 3 ack = 3 kind = DATA info = 11       the 12th line
SWP: Sending frame: seq = 0 ack = 7 kind = NAK info =
SWP: Sending frame: seq = 0 ack = 0 kind = NAK info =
SWP: Sending frame: seq = 7 ack = 0 kind = DATA info = 15       the 16th line
SWP: Sending frame: seq = 4 ack = 0 kind = DATA info = 12       the 13th line
SWP: Sending frame: seq = 0 ack = 0 kind = DATA info = 16       the 17th line
SWP: Sending frame: seq = 1 ack = 0 kind = DATA info = 17       the 18th line
SWP: Sending frame: seq = 2 ack = 0 kind = DATA info = 18       the 19th line
SWP: Sending frame: seq = 7 ack = 0 kind = DATA info = 15       the 16th line
```

Figure 1: Virtual Machine 1 Part 1

```
SWP: Sending frame: seq = 6 ack = 3 kind = DATA info = 14       the 15th line
SWP: Sending frame: seq = 3 ack = 3 kind = DATA info = 11       the 12th line
SWP: Sending frame: seq = 0 ack = 7 kind = NAK info =
SWP: Sending frame: seq = 0 ack = 0 kind = NAK info =
SWP: Sending frame: seq = 7 ack = 0 kind = DATA info = 15       the 16th line
SWP: Sending frame: seq = 4 ack = 0 kind = DATA info = 12       the 13th line
SWP: Sending frame: seq = 0 ack = 0 kind = DATA info = 16       the 17th line
SWP: Sending frame: seq = 1 ack = 0 kind = DATA info = 17       the 18th line
SWP: Sending frame: seq = 2 ack = 0 kind = DATA info = 18       the 19th line
SWP: Sending frame: seq = 7 ack = 0 kind = DATA info = 15       the 16th line
SWP: Sending frame: seq = 1 ack = 0 kind = DATA info = 17       the 18th line
SWP: Sending frame: seq = 0 ack = 0 kind = DATA info = 16       the 17th line
SWP: Sending frame: seq = 2 ack = 0 kind = DATA info = 18       the 19th line
SWP: Sending frame: seq = 7 ack = 0 kind = DATA info = 15       the 16th line
SWP: Sending frame: seq = 0 ack = 2 kind = NAK info =
SWP: Sending frame: seq = 0 ack = 4 kind = NAK info =
SWP: Sending frame: seq = 1 ack = 4 kind = DATA info = 17       the 18th line
SWP: Sending frame: seq = 2 ack = 4 kind = DATA info = 18       the 19th line
SWP: Sending frame: seq = 0 ack = 4 kind = DATA info = 16       the 17th line
SWP: Sending frame: seq = 7 ack = 4 kind = DATA info = 15       the 16th line
SWP: Sending frame: seq = 0 ack = 4 kind = ACK info =
SWP: Sending frame: seq = 2 ack = 4 kind = DATA info = 18       the 19th line
SWP: Sending frame: seq = 1 ack = 4 kind = DATA info = 17       the 18th line
SWP: Sending frame: seq = 0 ack = 4 kind = DATA info = 16       the 17th line
SWP: Sending frame: seq = 7 ack = 4 kind = DATA info = 15       the 16th line
SWP: Sending frame: seq = 0 ack = 4 kind = ACK info =
SWP: Sending frame: seq = 2 ack = 4 kind = DATA info = 18       the 19th line
SWP: Sending frame: seq = 1 ack = 4 kind = DATA info = 17       the 18th line
SWP: Sending frame: seq = 0 ack = 4 kind = DATA info = 16       the 17th line
SWP: Sending frame: seq = 7 ack = 0 kind = DATA info = 15       the 16th line
SWP: Sending frame: seq = 0 ack = 0 kind = NAK info =
SWP: Sending frame: seq = 0 ack = 0 kind = ACK info =
SWP: Sending frame: seq = 2 ack = 0 kind = DATA info = 18       the 19th line
SWP: Sending frame: seq = 1 ack = 0 kind = DATA info = 17       the 18th line
SWP: Sending frame: seq = 0 ack = 0 kind = DATA info = 16       the 17th line
SWP: Sending frame: seq = 7 ack = 0 kind = DATA info = 15       the 16th line
SWP: Sending frame: seq = 0 ack = 0 kind = ACK info =
SWP: Sending frame: seq = 2 ack = 0 kind = DATA info = 18       the 19th line
SWP: Sending frame: seq = 1 ack = 0 kind = DATA info = 17       the 18th line
SWP: Sending frame: seq = 0 ack = 0 kind = DATA info = 16       the 17th line
SWP: Sending frame: seq = 7 ack = 0 kind = DATA info = 15       the 16th line
SWP: Sending frame: seq = 0 ack = 4 kind = NAK info =
SWP: Sending frame: seq = 0 ack = 4 kind = ACK info =
SWP: Sending frame: seq = 3 ack = 6 kind = DATA info = 19       the 20th line
SWP: Sending frame: seq = 4 ack = 6 kind = DATA info = 20       the 21th line
```

Figure 2: Virtual Machine 1 Part 2

```
SWP: Sending frame: seq = 0 ack = 4 kind = ACK info =
SWP: Sending frame: seq = 3 ack = 6 kind = DATA info = 19       the 20th line
SWP: Sending frame: seq = 4 ack = 6 kind = DATA info = 20       the 21th line
SWP: Sending frame: seq = 5 ack = 6 kind = DATA info = 21       the 22th line
SWP: Sending frame: seq = 6 ack = 6 kind = DATA info = 22       the 23th line
SWP: Sending frame: seq = 0 ack = 6 kind = NAK info =
SWP: Sending frame: seq = 3 ack = 6 kind = DATA info = 19       the 20th line
SWP: Sending frame: seq = 0 ack = 6 kind = ACK info =
SWP: Sending frame: seq = 7 ack = 6 kind = DATA info = 23       the 24th line
SWP: Sending frame: seq = 0 ack = 6 kind = DATA info = 24       the 25th line
SWP: Sending frame: seq = 5 ack = 6 kind = DATA info = 21       the 22th line
SWP: Sending frame: seq = 1 ack = 2 kind = DATA info = 25       the 26th line
SWP: Sending frame: seq = 0 ack = 2 kind = NAK info =
SWP: Sending frame: seq = 6 ack = 2 kind = DATA info = 22       the 23th line
SWP: Sending frame: seq = 0 ack = 3 kind = ACK info =
SWP: Sending frame: seq = 7 ack = 3 kind = DATA info = 23       the 24th line
SWP: Sending frame: seq = 0 ack = 3 kind = NAK info =
SWP: Sending frame: seq = 0 ack = 3 kind = DATA info = 24       the 25th line
SWP: Sending frame: seq = 2 ack = 3 kind = DATA info = 26       the 27th line
SWP: Sending frame: seq = 3 ack = 3 kind = DATA info = 27       the 28th line
SWP: Sending frame: seq = 1 ack = 3 kind = DATA info = 25       the 26th line
SWP: Sending frame: seq = 4 ack = 7 kind = DATA info = 28       the 29th line
SWP: Sending frame: seq = 5 ack = 7 kind = DATA info = 29       the 30th line
SWP: Sending frame: seq = 6 ack = 7 kind = DATA info = 30       the 31th line
SWP: Sending frame: seq = 0 ack = 2 kind = ACK info =
SWP: Sending frame: seq = 3 ack = 2 kind = DATA info = 27       the 28th line
SWP: Sending frame: seq = 6 ack = 2 kind = DATA info = 30       the 31th line
SWP: Sending frame: seq = 4 ack = 2 kind = DATA info = 28       the 29th line
SWP: Sending frame: seq = 5 ack = 2 kind = DATA info = 29       the 30th line
SWP: Sending frame: seq = 3 ack = 2 kind = DATA info = 27       the 28th line
SWP: Sending frame: seq = 5 ack = 2 kind = DATA info = 29       the 30th line
SWP: Sending frame: seq = 6 ack = 2 kind = DATA info = 30       the 31th line
SWP: Sending frame: seq = 4 ack = 2 kind = DATA info = 28       the 29th line
SWP: Sending frame: seq = 6 ack = 2 kind = DATA info = 30       the 31th line
SWP: Sending frame: seq = 7 ack = 2 kind = DATA info = 31       the 32th line
SWP: Sending frame: seq = 0 ack = 2 kind = DATA info = 32       the 33th line
SWP: Sending frame: seq = 1 ack = 2 kind = DATA info = 33       the 34th line
SWP: Sending frame: seq = 0 ack = 2 kind = NAK info =
SWP: Sending frame: seq = 6 ack = 2 kind = DATA info = 30       the 31th line
SWP: Sending frame: seq = 7 ack = 2 kind = DATA info = 31       the 32th line
SWP: Sending frame: seq = 0 ack = 2 kind = DATA info = 32       the 33th line
SWP: Sending frame: seq = 1 ack = 2 kind = DATA info = 33       the 34th line
SWP: Sending frame: seq = 7 ack = 2 kind = DATA info = 31       the 32th line
SWP: Sending frame: seq = 6 ack = 2 kind = DATA info = 30       the 31th line
SWP: Sending frame: seq = 0 ack = 2 kind = DATA info = 32       the 33th line
SWP: Sending frame: seq = 0 ack = 2 kind = NAK info =
SWP: Sending frame: seq = 6 ack = 2 kind = DATA info = 30       the 31th line
SWP: Sending frame: seq = 7 ack = 2 kind = DATA info = 31       the 32th line
SWP: Sending frame: seq = 0 ack = 2 kind = DATA info = 32       the 33th line
SWP: Sending frame: seq = 1 ack = 2 kind = DATA info = 33       the 34th line
SWP: Sending frame: seq = 7 ack = 2 kind = DATA info = 31       the 32th line
SWP: Sending frame: seq = 6 ack = 2 kind = DATA info = 30       the 31th line
SWP: Sending frame: seq = 0 ack = 2 kind = DATA info = 32       the 33th line
SWP: Sending frame: seq = 1 ack = 2 kind = DATA info = 33       the 34th line
SWP: Sending frame: seq = 6 ack = 2 kind = DATA info = 30       the 31th line
SWP: Sending frame: seq = 0 ack = 2 kind = DATA info = 32       the 33th line
SWP: Sending frame: seq = 1 ack = 2 kind = DATA info = 33       the 34th line
SWP: Sending frame: seq = 7 ack = 2 kind = DATA info = 31       the 32th line
SWP: Sending frame: seq = 2 ack = 2 kind = DATA info = 34       the last line
SWP: Sending frame: seq = 2 ack = 2 kind = DATA info = 34       the last line
SWP: Sending frame: seq = 2 ack = 2 kind = DATA info = 34       the last line
```

Figure 3: Virtual Machine 1 Part 4

```
ve/cz3006/ass1
└$ java VMach 2                                                    130 ↵
VMach is making a connection with NetSim...
VMach(52770) <===> NetSim(ZWLori-MAC.local/10.27.156.11:54321)
SWP: Sending frame: seq = 0 ack = 7 kind = DATA info = 0        this is  a test from site 2
SWP: Sending frame: seq = 1 ack = 7 kind = DATA info = 1        the 2nd line
SWP: Sending frame: seq = 2 ack = 7 kind = DATA info = 2        the 3rd line
SWP: Sending frame: seq = 3 ack = 7 kind = DATA info = 3        the 4th line
SWP: Sending frame: seq = 0 ack = 7 kind = NAK info =
SWP: Sending frame: seq = 0 ack = 7 kind = ACK info =
SWP: Sending frame: seq = 3 ack = 7 kind = DATA info = 3        the 4th line
SWP: Sending frame: seq = 2 ack = 7 kind = DATA info = 2        the 3rd line
SWP: Sending frame: seq = 0 ack = 7 kind = DATA info = 0        this is  a test from site 2
SWP: Sending frame: seq = 1 ack = 7 kind = DATA info = 1        the 2nd line
SWP: Sending frame: seq = 0 ack = 2 kind = ACK info =
SWP: Sending frame: seq = 0 ack = 2 kind = NAK info =
SWP: Sending frame: seq = 3 ack = 5 kind = DATA info = 3        the 4th line
SWP: Sending frame: seq = 2 ack = 5 kind = DATA info = 2        the 3rd line
SWP: Sending frame: seq = 1 ack = 5 kind = DATA info = 1        the 2nd line
SWP: Sending frame: seq = 0 ack = 5 kind = DATA info = 0        this is  a test from site 2
SWP: Sending frame: seq = 0 ack = 5 kind = NAK info =
SWP: Sending frame: seq = 3 ack = 5 kind = DATA info = 3        the 4th line
SWP: Sending frame: seq = 1 ack = 5 kind = DATA info = 1        the 2nd line
SWP: Sending frame: seq = 2 ack = 5 kind = DATA info = 2        the 3rd line
SWP: Sending frame: seq = 0 ack = 5 kind = DATA info = 0        this is  a test from site 2
SWP: Sending frame: seq = 0 ack = 5 kind = ACK info =
SWP: Sending frame: seq = 4 ack = 7 kind = DATA info = 4        the 5th line
SWP: Sending frame: seq = 5 ack = 7 kind = DATA info = 5        the 6th line
SWP: Sending frame: seq = 6 ack = 7 kind = DATA info = 6        the 7th line
SWP: Sending frame: seq = 7 ack = 7 kind = DATA info = 7        the 8th line
SWP: Sending frame: seq = 4 ack = 7 kind = DATA info = 4        the 5th line
SWP: Sending frame: seq = 0 ack = 7 kind = NAK info =
SWP: Sending frame: seq = 0 ack = 2 kind = ACK info =
SWP: Sending frame: seq = 0 ack = 2 kind = NAK info =
SWP: Sending frame: seq = 6 ack = 2 kind = DATA info = 6        the 7th line
SWP: Sending frame: seq = 4 ack = 2 kind = DATA info = 4        the 5th line
SWP: Sending frame: seq = 7 ack = 2 kind = DATA info = 7        the 8th line
SWP: Sending frame: seq = 5 ack = 2 kind = DATA info = 5        the 6th line
SWP: Sending frame: seq = 0 ack = 3 kind = DATA info = 8        the 9th line
SWP: Sending frame: seq = 1 ack = 3 kind = DATA info = 9        the 10th line
SWP: Sending frame: seq = 2 ack = 3 kind = DATA info = 10       the 11th line
SWP: Sending frame: seq = 3 ack = 3 kind = DATA info = 11       the 12th line
SWP: Sending frame: seq = 0 ack = 3 kind = NAK info =
SWP: Sending frame: seq = 4 ack = 6 kind = DATA info = 12       the 13th line
SWP: Sending frame: seq = 0 ack = 6 kind = NAK info =
```

Figure 4: Virtual Machine 2 Part 1

```
SWP: Sending frame: seq = 1 ack = 3 kind = DATA info = 9        the 10th line
SWP: Sending frame: seq = 2 ack = 3 kind = DATA info = 10       the 11th line
SWP: Sending frame: seq = 3 ack = 3 kind = DATA info = 11       the 12th line
SWP: Sending frame: seq = 0 ack = 3 kind = NAK info =
SWP: Sending frame: seq = 4 ack = 6 kind = DATA info = 12       the 13th line
SWP: Sending frame: seq = 0 ack = 6 kind = NAK info =
SWP: Sending frame: seq = 0 ack = 6 kind = ACK info =
SWP: Sending frame: seq = 1 ack = 6 kind = DATA info = 9        the 10th line
SWP: Sending frame: seq = 4 ack = 6 kind = DATA info = 12       the 13th line
SWP: Sending frame: seq = 3 ack = 6 kind = DATA info = 11       the 12th line
SWP: Sending frame: seq = 2 ack = 6 kind = DATA info = 10       the 11th line
SWP: Sending frame: seq = 0 ack = 6 kind = ACK info =
SWP: Sending frame: seq = 3 ack = 6 kind = DATA info = 11       the 12th line
SWP: Sending frame: seq = 1 ack = 6 kind = DATA info = 9        the 10th line
SWP: Sending frame: seq = 4 ack = 6 kind = DATA info = 12       the 13th line
SWP: Sending frame: seq = 2 ack = 6 kind = DATA info = 10       the 11th line
SWP: Sending frame: seq = 5 ack = 6 kind = DATA info = 13       the 14th line
SWP: Sending frame: seq = 6 ack = 6 kind = DATA info = 14       the 15th line
SWP: Sending frame: seq = 7 ack = 6 kind = DATA info = 15       the 16th line
SWP: Sending frame: seq = 0 ack = 6 kind = DATA info = 16       the 17th line
SWP: Sending frame: seq = 7 ack = 6 kind = DATA info = 15       the 16th line
SWP: Sending frame: seq = 5 ack = 6 kind = DATA info = 13       the 14th line
SWP: Sending frame: seq = 6 ack = 6 kind = DATA info = 14       the 15th line
SWP: Sending frame: seq = 0 ack = 6 kind = DATA info = 16       the 17th line
SWP: Sending frame: seq = 7 ack = 6 kind = DATA info = 15       the 16th line
SWP: Sending frame: seq = 5 ack = 6 kind = DATA info = 13       the 14th line
SWP: Sending frame: seq = 0 ack = 6 kind = DATA info = 16       the 17th line
SWP: Sending frame: seq = 6 ack = 6 kind = DATA info = 14       the 15th line
SWP: Sending frame: seq = 1 ack = 6 kind = DATA info = 17       the 18th line
SWP: Sending frame: seq = 2 ack = 6 kind = DATA info = 18       the 19th line
SWP: Sending frame: seq = 3 ack = 6 kind = DATA info = 19       the 20th line
SWP: Sending frame: seq = 4 ack = 6 kind = DATA info = 20       the 21th line
SWP: Sending frame: seq = 1 ack = 6 kind = DATA info = 17       the 18th line
SWP: Sending frame: seq = 2 ack = 6 kind = DATA info = 18       the 19th line
SWP: Sending frame: seq = 3 ack = 6 kind = DATA info = 19       the 20th line
SWP: Sending frame: seq = 4 ack = 6 kind = DATA info = 20       the 21th line
SWP: Sending frame: seq = 1 ack = 6 kind = DATA info = 17       the 18th line
SWP: Sending frame: seq = 3 ack = 6 kind = DATA info = 19       the 20th line
SWP: Sending frame: seq = 4 ack = 6 kind = DATA info = 20       the 21th line
SWP: Sending frame: seq = 2 ack = 6 kind = DATA info = 18       the 19th line
SWP: Sending frame: seq = 0 ack = 2 kind = ACK info =
SWP: Sending frame: seq = 5 ack = 2 kind = DATA info = 21       the 22th line
SWP: Sending frame: seq = 6 ack = 2 kind = DATA info = 22       the 23th line
SWP: Sending frame: seq = 7 ack = 2 kind = DATA info = 23       the 24th line
SWP: Sending frame: seq = 0 ack = 2 kind = DATA info = 24       the 25th line
SWP: Sending frame: seq = 1 ack = 6 kind = DATA info = 17       the 18th line
SWP: Sending frame: seq = 3 ack = 6 kind = DATA info = 19       the 20th line
SWP: Sending frame: seq = 4 ack = 6 kind = DATA info = 20       the 21th line
SWP: Sending frame: seq = 2 ack = 6 kind = DATA info = 18       the 19th line
SWP: Sending frame: seq = 0 ack = 2 kind = ACK info =
SWP: Sending frame: seq = 5 ack = 2 kind = DATA info = 21       the 22th line
SWP: Sending frame: seq = 6 ack = 2 kind = DATA info = 22       the 23th line
SWP: Sending frame: seq = 7 ack = 2 kind = DATA info = 23       the 24th line
SWP: Sending frame: seq = 0 ack = 2 kind = DATA info = 24       the 25th line
SWP: Sending frame: seq = 0 ack = 2 kind = NAK info =
SWP: Sending frame: seq = 1 ack = 2 kind = DATA info = 25       the 26th line
SWP: Sending frame: seq = 2 ack = 2 kind = DATA info = 26       the 27th line
SWP: Sending frame: seq = 0 ack = 4 kind = ACK info =
SWP: Sending frame: seq = 0 ack = 4 kind = NAK info =
SWP: Sending frame: seq = 7 ack = 5 kind = DATA info = 23       the 24th line
SWP: Sending frame: seq = 0 ack = 5 kind = DATA info = 24       the 25th line
SWP: Sending frame: seq = 0 ack = 5 kind = NAK info =
SWP: Sending frame: seq = 3 ack = 5 kind = DATA info = 27       the 28th line
SWP: Sending frame: seq = 4 ack = 5 kind = DATA info = 28       the 29th line
SWP: Sending frame: seq = 5 ack = 5 kind = DATA info = 29       the 30th line
SWP: Sending frame: seq = 6 ack = 5 kind = DATA info = 30       the 31th line
SWP: Sending frame: seq = 3 ack = 5 kind = DATA info = 27       the 28th line
SWP: Sending frame: seq = 0 ack = 7 kind = ACK info =
SWP: Sending frame: seq = 0 ack = 7 kind = NAK info =
SWP: Sending frame: seq = 7 ack = 7 kind = DATA info = 31       the 32th line
SWP: Sending frame: seq = 0 ack = 2 kind = NAK info =
SWP: Sending frame: seq = 4 ack = 2 kind = DATA info = 28       the 29th line
SWP: Sending frame: seq = 5 ack = 2 kind = DATA info = 29       the 30th line
SWP: Sending frame: seq = 6 ack = 2 kind = DATA info = 30       the 31th line
SWP: Sending frame: seq = 0 ack = 2 kind = DATA info = 32       the 33th line
SWP: Sending frame: seq = 1 ack = 2 kind = DATA info = 33       the 34th line
SWP: Sending frame: seq = 2 ack = 2 kind = DATA info = 34       the last line
SWP: Sending frame: seq = 0 ack = 2 kind = ACK info =
SWP: Sending frame: seq = 0 ack = 5 kind = NAK info =
SWP: Sending frame: seq = 0 ack = 6 kind = NAK info =
SWP: Sending frame: seq = 0 ack = 0 kind = ACK info =
SWP: Sending frame: seq = 0 ack = 0 kind = NAK info =
SWP: Sending frame: seq = 0 ack = 0 kind = ACK info =
SWP: Sending frame: seq = 0 ack = 1 kind = ACK info =
SWP: Sending frame: seq = 0 ack = 1 kind = NAK info =
SWP: Sending frame: seq = 0 ack = 1 kind = ACK info =
SWP: Sending frame: seq = 0 ack = 2 kind = ACK info =
SWP: Sending frame: seq = 0 ack = 2 kind = NAK info =
SWP: Sending frame: seq = 0 ack = 2 kind = ACK info =
```

4.2 Final Result

The following images show the contents of the receiving files after the transmission progress.

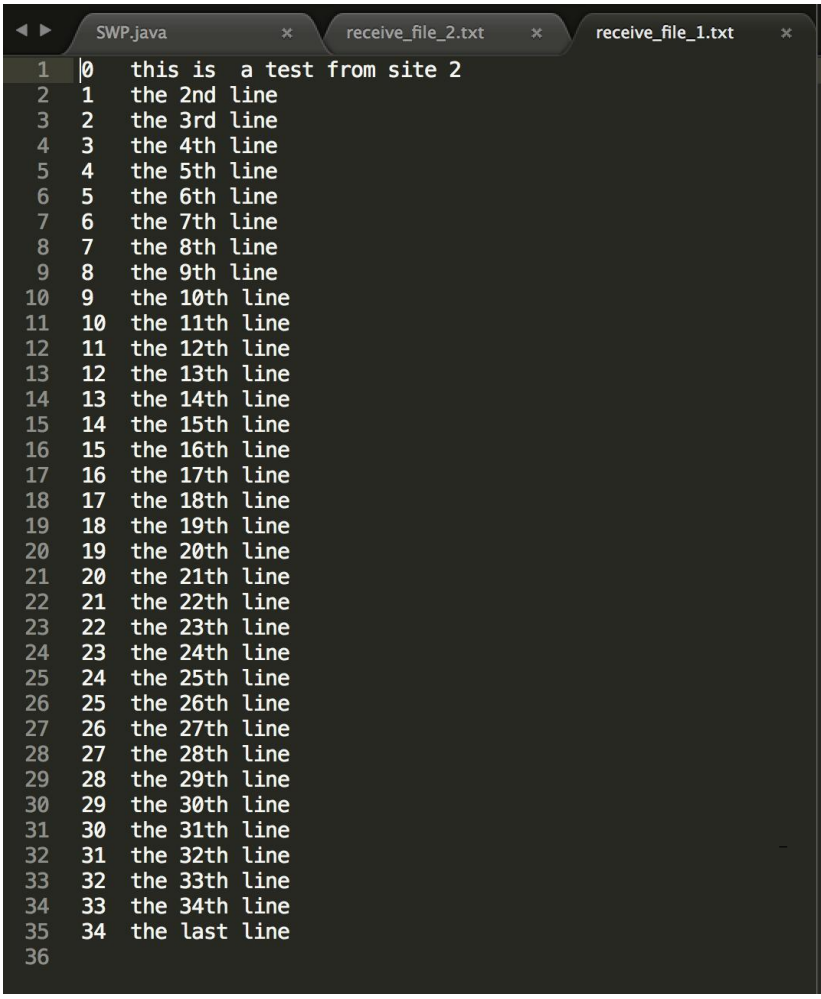

Figure: Receive_file_1.txt

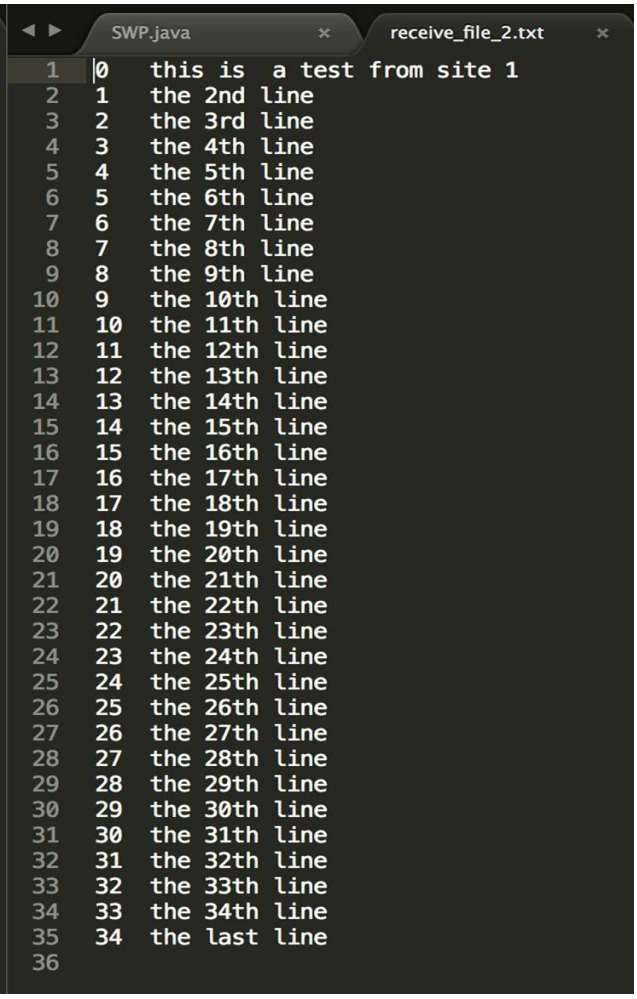

Figure: Receive_file_2.txt

Since the text in receive _file 1_txt represents the content received by Virtual Machine 1, it should be the same as the content in the sending _le from Virtual Machine 2. Similarly, what Virtual Machine 2 receives should be the same as the one stored in send file 1_txt. As shown in the screen-shots, the result for the transmission is exactly as expected.

Appendix

The contents from the following websites help reinforce my understanding in doing this assignment.

1. https://www.techopedia.com/definition/24204/network-layer
2. https://en.wikipedia.org/wiki/Sliding_window_protocol
3. http://techdifferences.com/difference-between-go-back-n-and-selective-repeat-protocol.html
4. https://en.wikipedia.org/wiki/Data_link_layer
5. http://www.ccs-labs.org/teaching/rn/animations/gbn_sr/