

**Assignment 1 Report**

**CZ3006: Net-Centric Computing**

**Academic Year: 2018/2019**

**Semester 1**

**Student Name: Chen Zhiwei**

**Matriculation Number: U1620741J**

**Lab Group: SSP2**

Summary of Tasks Done

I had implemented the Sliding Window Protocol component with Selective Repeat, which is able to withstand quality level 3 of the Network Simulator component.

The features I had implemented in the Sliding Window Protocol component are as follow:

# Full-duplex Data Communication

It’s important to implement this feature, because if 2 separate simplex data channels are used to transmit data between machines, the bandwidth resource will be wasted. In effect, the user is paying for 2 circuits but using only the capacity of 1.

Thus, a better way is to use the same circuit for data in both directions. To achieve full-duplex data communication, the data-link layer must be able to handle both in-coming and out-going data, thereby enabling the machine to act as both the ‘sender’ and the ‘receiver’.

In the implementation, there exist 5 kinds of event, out of which, the “NETWORK\_LAYER\_READY” and “FRAME\_ARRIVAL” are the 2 primary events that allow the full-duplex data communication between 2 machines under Sliding Window Protocol component (damaged or lost frames will be handled by other events, which will be discussed in the later part of this report).

When “NETWORK\_LAYER\_READY” is observed, it means that a new packet has arrived, so the datalink-layer will first fetch this new packet from the network-layer, and then construct the respective frame to be sent to the physical-layer, which will then transmit the frame further. As illustrated, the machine is now acting as the “sender”.

When “FRAME\_ARRIVAL” is observed, it means that a new frame has arrived, so the datalink-layer will first fetch the incoming frame from the physical-layer, and then examine what kind of frame it is – {PFrame.DATA, PFrame.NAK, or PFrame.ACK}. As illustrated, the machine is now acting as the “receiver”, and whichever kind of frame this incoming frame is, it will be handled by the logic in the protocol, which will be discussed in the later part of this report.

# In-order Delivery of Packets to the Network-Layer

The Sliding Window Protocol with Selective Repeat ensures that the order of the packets delivered to the network-layer by doing the follow:

1. Both sender and receiver maintain a window of acceptable sequence numbers.
   1. The sender’s window size starts out at 0 and grows to some predefined maximum, MAX\_SEQ.
   2. The receiver’s window, in contrast, is always fixed in size and equal to MAX\_SEQ.
   3. The receiver has a buffer reserved for each sequence number within its fixed window. Associated with each buffer is a bit (arrived[]) telling whether the buffer is full or empty.
2. Whenever a frame arrives, its sequence number is checked by the function between() to see if it falls within the window.
   1. If so and if it has not already been received, it is accepted and stored. This action is taken without regard to whether or not it contains the next packet expected by the network layer.
   2. Of course, it must be kept within the data link layer and not passed to the network layer until all the lower-numbered frames have already been delivered to the network layer in the correct order.
3. However, non-sequential receive introduces certain problem.
   1. The essence of the problem is that after the receiver advanced its window, the new range of valid sequence numbers overlapped the old one.
   2. Consequently, the following batch of frames might be either duplicates (if all the acknowledgements were lost) or new ones (if all the acknowledgements were received).
   3. The receiver has no way of distinguishing these two cases.
   4. To solve the issue, the protocol implemented must make sure that after the receiver has advanced its window, there is no overlap with the original window.
   5. To ensure that there is no overlap, the maximum window size should be at most half the range of the sequence numbers i.e. (MAX\_SEQ + 1)/2.

# Selective Repeat Retransmission Strategy

As communication channel is inevitably unreliable, it is important to handle error recovery when damaged or lost frames are transmitted.

The essence of selective repeat transmission strategy is as follow:

1. When a bad frame that is received is discarded, the good frames received after it are buffered instead of discarded.
2. When the sender times out, only the oldest unacknowledged frame is retransmitted.
3. If that frame arrives correctly, the receiver can deliver all frames to the network-layer, in sequence, as all the rest of the frames are buffered.
4. Selective repeat is often combined with having the receiver send a negative acknowledgement (NAK) when it detects an error, for example, when it receives a checksum error or a frame out of sequence. NAKs stimulate retransmission before the corresponding timer expires and thus improve performance.

# Synchronization with the Network-Layer by Granting Credits

Initially, the sender’s datalink-layer will inform its network-layer that it has (MAX + 1)/2 buffers available i.e. grant (MAX + 1)/2 credits to the network-layer. As such, the sender’s network-layer will prepare (MAX + 1)/2 packets and send them to the its datalink-layer. The sender’s datalink-layer will then construct respective frames and transmit them to the receivers. Afterwards, whenever a frame arrived with ack piggybacked on it intact, it indicates that the data has been successfully transmitted to the receiver, and thus, the sender’s datalink-layer will grant 1 credit to the network-layer, allowing it to send 1 more packet to the datalink-layer.

# Negative Acknowledgement

Whenever the receiver has reason to suspect that an error has occurred, it sends a negative acknowledgement (NAK) frame back to the sender. Such a frame is a request for retransmission of the frame specified in the NAK. There are two cases when the receiver should be suspicious:

1. A damaged frame has arrived or
2. A frame other than the expected one arrived (potential lost frame).

To avoid making multiple requests for retransmission of the same lost frame, the receiver should keep track of whether a NAK has already been sent for a given frame. The variable no\_nak in protocol 6 is true if no NAK has been sent yet for frame\_expected.

1. If the NAK gets mangled or lost, no real harm is done, since the sender will eventually time out and retransmit the missing frame anyway.
2. If the wrong frame arrives after a NAK has been sent and lost, no\_nak will be true and the auxiliary timer will be started. When it expires, an ACK will be sent to resynchronize the sender to the receiver’s current status.

# Separate Acknowledgement when the Reverse Traffic is Light or None

In protocol 6, ack for the frame received by the sender is supposed to be send to the receiver by piggybacking on the frame to be send to the receiver. This is taking advantage of the reverse traffic, such that bandwidth can be saved. However, when the reverse traffic is light or none, the ack can get delayed infinitely, and that will cause problem.

To solve the problem, after an in-sequence data frame arrives, an auxiliary timer is started by start\_ack\_timer.

1. If no reverse traffic has presented itself before this timer expires, a separate acknowledgement frame is sent. An interrupt due to the auxiliary timer is called an ack\_timeout event. With this arrangement, one-directional traffic flow is now possible because the lack of reverse data frames onto which acknowledgements can be piggybacked is no longer an obstacle. Only one auxiliary timer exists, and if start\_ack\_timer is called while the timer is running, it is reset to a full acknowledgement timeout interval.

It is essential that the timeout associated with the auxiliary timer be appreciably shorter than the timer used for timing out data frames. This condition is required to make sure a correctly received frame is acknowledged early enough that the frame’s retransmission timer does not expire and retransmit the frame.