

Practical time: THURSDAY 2:00pm

COMPUTER COMMUNICATIONS 200

CC200 Assignment Report

CONNOR BEARDSMORE - 15504319

May 2016

**Source Code Organization**

The source code is organized to reflect a typical layered network architecture. Global variables were permitted to be used for the CNET network simulator and thus, the header file contains the declaration for these functions. Structures for frames and variables for the routing table, node windows and node buffers are all contained within the *assignment.h* header file.

Layers are present in the *assignment.c* from high to low. i.e. from the transport layer down to the data link layer. Only the network and data link layers were fully implemented in this assignment, with CNET calls utilizing the built-in application and physical layer. Wrapper functions were developed for the transport layer, essentially to illustrate the layering, despite these functions performing little work.

**Design Issues**

The sliding window implementation developed utilizes the basic Go-Back-N method, as demonstrated in the lecture slides. The window size represented by the global constant MAX\_SEQ can be modified, with all nodes having the same window size for simplicity. Every node has its own window and a buffer for every link it has.

The sliding window protocol is responsible for storing all sent frames sent for every link, until the receiver acks them. Overflow buffer is only used by nodes in the network that are required to forward frames. It stores frames that are required to be forwarded but cannot be sent at the moment due to a full window. Buffer size is equivalent to the size of the window, arbirtraily.

Next frame to send seqnum contains seqnum of next frame to send. Also represents the next free index in the circular buffer that represents the window. next ack expected is blah, next frame exepected is blah. At the beggining, everything is initialized to 0

*Sending Frames*

After the application layer generates a message, the transport layer reads the message via CNET\_read\_application(). The transport layer then sends the message down to the network layer. The network layer encapsulates the message and utilizes the routing table to determine which route to send the message out on. Once the payload reaches the data link layer it is added to the outgoing window and the window size is incremented by one. It's sequence number is determined using the nextFrameToSend value and the data link layer calls the transmit\_frame() function to send the frame over the physical layer. If the window is full at this stage, the message is added to the overflow buffer and the application layer is disabled for all destinations reached via that link. The frame checksum and size is calculated within transmit\_frame() and the frame is written to the physical layer via CNET\_write\_physical().

*Receiving Frames*

*Data Frames*

When a data frame is received, the first check is to ensure that the checksum is correct. This ensures that the frame was not corrupted. If the checksum is incorrect the frame is dropped and the sender will eventually timeout and re-transmit the corrupted frame. The sequence number is then checked to ensure the frame's sequence number is the one expected. If the number is less than we expected then the frame is a duplicate. The duplicate frame is dropped and acknowledgement for this duplicate send to the sending node. If the sequence number is greater than the expected the frame is also dropped until the expected frame is received.

If the sequence number is equal to the number in frameExpected, the frame is the frame we expected and hence it is processed. An acknowledgement frame is sent and the frame is sent up to the network layer. Two situations can now other. Either the frame is destined for the node it arrived on or it must be forwarded onward. If the frame is destined for the node it arrives on it is unpacked and sent up to the transport layer. The transport layer will then write the message to the application layer and the message has successfully been receiveced. If the frame is too be forwarded, the network layer reroutes the frame and passes it back down to the data link layer to send outwards again.

*Acknowledgement Frames*

When an ack frame arrives, the data link layer confirms that the sequence number is between the ackExpected value and the nextFrameToSend value. If this holds, all frames less than the acks sequence number and greater than ackExpected are implicitly received. Thus, all timers up to the ack received are stop and the number of frames in the window is decreased accordingly. If the buffer is not empty, items are added to the window from the buffer and transmitted. The application layer is also re-enabled if the buffer is now empty.

*Frame Re-transmission*

If a timer timeout occurs before the appropriate acknowledgement frame is received, we assume that the either the frame or the relevant acknowledgement failed to arrive. Thus, we resend the frame and reset the appropriate timer for this frame. The sequence number of the frame is left unchanged. This process will continue until the acknowledgement is successfully received for the frame.

**Error Conditions**

error condition stuff here.

**References**

Ling, Li. *"CNET Tips and Discussions."* Class lecture, Computer Communications from Curtin University, Perth, Australia, April 14 2016.

McDonald, C. "The Cnet Network Simulator (v3.3.3)." The Cnet Network Simulator (v3.3.3). Accessed May 11, 2016.

http://www.csse.uwa.edu.au/cnet/.

Tanenbaum, Andrew S. *Computer Networks.* Upper Saddle River, NJ: Prentice Hall PTR, 1996.