COMPUTER COMMUNICATIONS 200

SEMESTER 1, 2016

Implementation of sliding window protocol

Student Name: Priyank Joshi

Student Number: 17812234

Due date: 23/05/2016

Tutorial time: Thursday 8-10am

I declare that:
•The above information is complete and accurate.
•The work I am submitting is entirely my own, except where clearly indicated otherwise and correctly referenced.
•I have taken (and will continue to take) all reasonable steps to ensure my work is not accessible to any other students who may gain unfair advantage from it.
•I have not previously submitted this work for any other unit, whether at Curtin University or elsewhere, or for prior attempts at this unit, except where clearly indicated otherwise.

I understand that:
•Plagiarism and collusion are dishonest, and unfair to all other students.
•Detection of plagiarism and collusion may be done manually or by using tools (such as Turnitin).
•If I plagiarise or collude, I risk failing the unit with a grade of ANN ("Result Annulled due to Academic Misconduct"), which will remain permanently on my academic record. I also risk termination from my course and other penalties.
•Even with correct referencing, my submission will only be marked according to what I have done myself, specifically for this assessment. I cannot re-use the work of others, or my own previously submitted work, in order to fulfil the assessment requirements.
•It is my responsibility to ensure that my submission is complete, correct and not corrupted.

The "Go-back-N" protocol

The sliding window protocol using cumulative ACK is called as Go- Back – N ARQ protocol. The cumulative acknowledge method has built in error recovery because N Ack lost but Ack n+1 is received which say that to sender come to know that frame N was received at receiver side. The cumulative Ack method is used to improve the reliability and it is preferred in high rate of error environment.

Implementation

Implement the "Sliding Window" protocol on a 7- node network. The topology of the network are given below

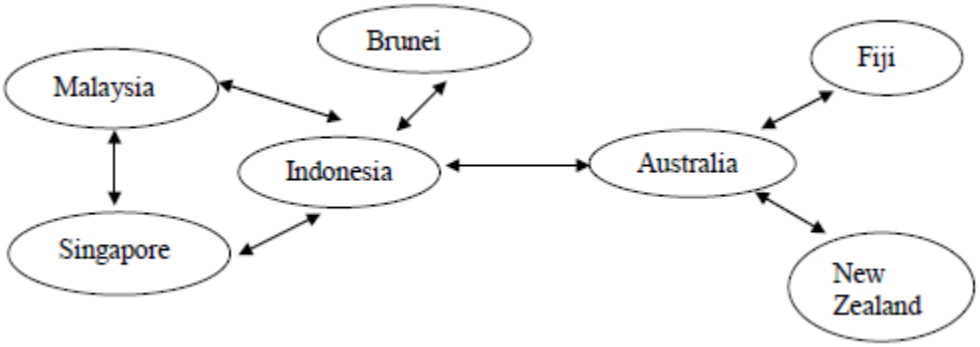

Figure 4: Network

The following files are used to implement the sliding window protocol in CNET. They are

1. assignment.c (implementation file with extension .c)
2. SLIDINGWINDOW(mapping file)

assignment.c

Creation of several structs including FRAMEKIND, that can be either data or acknowledgement. Another struct for PACKET that has len (length) and also msg (message) of type char containing max message size. Thirdly, a struct for FRAME that contains the FRAMEKIND, len, checksum, seqno (sequence number), toadd (to destination address), fromadd (from source address).

Declaration of constants and macros for packet header size, packet size, frame header size, frame size, true, false, bits per frame, max sequence number, and buffersize. Then also

declaration of global variables that are will be used in this protocol. I have used these constants and macros which makes it easier to handle buffersize and also incrementation of sequence number from 0 to 7.

The functions that include in the assignment1.c are as follows:

static void inc(int* seq)

This is utility function that receives an integer and increments by reference, Module according account MAXSEQ, which increments the sequence number. This makes it much simple to increment the sequence number up to a certain point which is 7.

between(int a, int b, int c)

Function that returns TRUE if all three parameters it receives are in Increasing order MAXSEQ module, which is true if a<=b<c if b<c <a if c <a <=b. Note that there are six other combinations, Not met, and therefore the function returns FALSE. The function is used to check whether the number of a frame is within a range, this will be used in physical_ready so that the receiving node can ensure the correct sequence number of ACK or DATA has been received.

application_ready(CnetEvent ev, CnetTimerID time, CnetData data)

This function basically the application layer in which the function waits for the event, EV_APPLICATION, when the event is available, then this function will send a frame with a message from a source node to a specific random address (down from application). An IF statement to check if the buffersize is full then to disable application to destination address. Then use the routing table to determine nodenumber (source address) and destination address and hence transmit the frame accordingly. Therefore, the sender's upper window will advance by making the nextframetosend to be incremented.

transmit_frame(PACKET* packet, size_t packetLength, FRAMEKIND kind, int seqno, FRAME f , int fromaddtemp, int toaddtemp);

This function is used to encapsulate the packet into the frame and then transmit it from source to destination node using the routing table. It also calls getRelayPoint so that for example frame to be sent from Malaysia to Australia then the getRelayPoint is Indonesia, hence that will be the current node, which will transfer the frame to Australia. Therefore, start the timer and if the frame is DATA then print statements indicating from transmission of data from source to destination node and sequence number of the frame. Else if the frame is ACK, then print

statements to show details such as source node and destination node and the sequence number of the frame. Then it will write to physical layer by calling the event.

getRelayPoint(int link, int destadd, int sendadd)

This function is to determine which when fro example Malaysia want to send to Fiji, so it will use link 2, that connects to Indonesia, this will be the next node, it checks the destination address, which is meant for Fiji, and so finds next relay point to send the frame which is Australia in link 4, and then Australia checks that this address is meant for Fiji, which is finally sent to Fiji.

physical_ready(CnetEvent ev, CnetTimerID time, CnetData data)

Once the frame has been received, the receiver will check detect for the checksum of the frame hence if it is bad checksum then we ignore the frame. The receiver will check if the kind of frame is ACK or DATA, and so if it is ACK then it will check if the sequence number is between within the range. If so, then it will print out statements such as from which node the ACK has been received, the sequence number and its checksum. Then we decrement the number of buffer, to make it available for more incoming ACKs and call stop timer. We then contract the sender's window by incrementing ackexpected. Otherwise if the frame to be received is DATA then the receiver will check if the sequence number is within the range, if so then print out statements and then call write application. Else if it is not DATA, then we ignore it.

timeouts(CnetEvent ev, CnetTimerID time, CnetData data)

This is used to handle the timer, for example if source node sends to destination node, then the timer will start, It will handle error if the frame is lost, and source node does not send any more frames, then timer will expire, by printing out statement to resend DATA and then call transmit_frame so that it can go back N frames and retransmit the frames again. Also when a frame has arrived at the destination node, then to call stop timer.

showstate(CnetEvent ev, CnetTimerID time, CnetData data)

This function is to keep track of the ackexpected, nextframetosend and frameexpected in cnet simulator by clicking on the tab "show state"

reboot_node (CnetEvent ev, CnetTimerID time, CnetData data)

This function is like the main() function in C program. This is where each node is initially rebooted by calling its reboot_node function. Allocation of memory is done here for inpacket, outpacket, timers, and arrived. A FOR loop is also introduced so that timer can be reset and also arrived packet is 0 to avoid any errors. This function includes all the event handlers.

SLIDINGWINDOW(mapping file)

The mapping code is explained below Set message rate, propagation delay and frame corruption

messagerate= 100ms,

propagationdelay= 1500ms,

probframecorrupt= 3

messagerate is the rate at which the application layer will generate the messages.

Propagationdelay is the propagation delay in time along a link

probframecorrupt is the probability that a frame on a link will be corrupted.

This file also includes the mapping between each nodes/hosts basically the topology in which the hosts are arranged and connection of links between each nodes. It similar to the assignment specification.

Design issues

The above explanation is the implementation of the sliding window protocol, which is expected. However, it has design issues such as in physical_ready() function, when dealing with receiving DL_ACK, the IF statement between(ackexpected, f.seqno, nextframetosend) does not execute, i.e the if statement is not true and therefore does not print out statements such as node, ack received from source node and frame's checksum. Although, the simulation clearly shows data is being transmitted from source node, but due to this it does not print statements when it is received at the destination node. The same goes with when receiving DL_DATA, the IF statement is not been executed which means is not true, and therefore is unable to print out statements that data received, upto application and the frame's checksum.