COMP535 PA# 1 Group 14

Files altered and newly included:
udp.c
cli.c
cli.h
windowmanager.c
windowmanager.h

General design notes:

This design ended up being more elaborate than a single stop and wait outstanding packet, but provides better performance in that a user is able to send messages while waiting for an acknowledgement for an outstanding packet. All functionality was implemented over the existing udp structure in gnc – so when you call gnc in RDP mode using -r, you're now using the "reliable mode" built on top. A window system was implemented, where outstanding packets were put into a data structure (rdp_pkt) that was placed in a window. A thread started in the gnc command (in the cli.c file) was responsible for this window's maintenance, ensuring timer values were updated and packets with expired timers were resent and restarted.

Theoretically users may also handle messages received out of order, as the acknowledgement system is based on packet message data alone. The window checks off packets that have received ACKs by matching their payload, this means that messages holding the same data may be *acknowledged out of order.* This was deemed irrelevant in this design context, where a message with the same payload/data was considered to be the same, and could be ACK'd interchangeably in our window without consequence.

The window.c file contains all functions relevant to this window's maintenance, including creating, adding and deleting packets. It also contains the thread's functions in window maintenance, updating timers and checking for expired packets to be resent.

The cli.c file implements a layer of extra functionality over the basic UDP functionality at the application level. It contains application level functions to utilize information from the udp packets in order to discern packets received as messages from ACKs. The method rdp_send wraps the udp_send capacity with extra functionality in the form of remembering messages that need to be ack'd (by adding them to the window). The new rdp_recv_callback function discerns ACKs from messages and responds to them accordingly. The send_ack function creates a special packet along with the original payload for acknowledgement.

To create an ACK, the sender temporarily changes its source port number so that the first bit is set to 1, while the original source port could still be extracted. The receiver then checks if a message received was an ack, extracts the relevant information (here the payload) and removes the packet waiting in its own window.

How to Use:
-one router launches "gnc -r IP_ADDRESS_OTHER_ROUTER PORT"
-other router launches "gnc -r -l PORT"

A number of messages may be sent awaiting acknowledgement, up to the WINDOW_SIZE defined in windowmanager.c. **If you'd like to have more information about what's going on, there's a folder of "verbose versions" that include print statements about the current operations.** The window holding the unack'd packets is printed every time a a packet is added to the window.

Implementation Details:

UDP.C

The only change to this file was to the constants UDP_LOCAL_PORT_RANGE_START to 0x7f9b (32667) and UDP_LOCAL_PORT_RANGE_END to 0x7fff (32767). This enabled us to reserve a bit in the header source port bits for flagging an ACK (higher that 32767 means we take up our ACK flag with port representation).

CLI.C

int rdp_send(struct udp_pcb *pcb , struct pbuf *p)
// wrapper for udp_send that also adds a packet to the window awaiting acknowledgement
// calls add_pkt_window and udp_send
// returns an int use for error check

void rdp_recv_callback(void *arg, struct udp_pcb *pcb, struct pbuf *p, uchar *addr, uint16_t port_raw);
//callback function for receipt of a message
//determines whether a message received was ack or msg via src port value
//if ack, deletes pkt from window, if msg, sends and ack
//calls del_pkt_window, send_ack and udp functions

void send_ack(struct udp_pcb *pcb, struct pbuf *p)
//function called to send an ack upon receipt of a message
//construct a new pcb w special flags in the port #
//does this by temporarily modifying the src port
//resends original payload
//calls udp_send

WINDOWMANAGER.C

//THREAD FUNCTIONS
void thread_timer_actions();
//main function to manage timers using a thread created in cli.c
//while the global var is set to not shut down
//thread updates timers in timer_update
//if there are expired timers in window, thread resends and restarts timer

void timer_update();
//function updates all rdp_pkt structs in window
//updates timer values based on time
//executed by thread

void timeout_event_handler();
//handler for time_out events
//a function that checks for timed-out packets in the window
//if a packet timer is out (indicated in time_left of the rdp_pkt struct)
//calls udp_send using info from rdp_pkt in window
//also restarts the timer

//UTILITY FUNCTIONS
int add_pkt_window(struct udp_pcb *pcb, struct pbuf *p);
//when a packet is sent must be added to list of unacked packets (window)
//"window" = list of unacked packets w timers, awaiting acking receipt
//creates rdp_pkt structure, adds to list
// undefined behaviour if window overflows
// window size defined as constant WINDOW_SIZE

int del_pkt_window(struct udp_pcb *pcb, struct pbuf *p);
//upon receipt of an ack, delete_pkt is called to remove an entry a packet
//from the window of packets awaiting an ack
//delete packet by freeing memory and reiniting to 0 so we know it's empty
//if del_pkt called there should be a packet awaiting an ack
//if no packet found, returns an error

//HELPER FUNCTIONS
struct rdp_pkt * create_pkt(struct udp_pcb *pcb, struct pbuf *p);
//creates a rdp_pkt struct by copying from current pbuf
//this struct is entry placed in window for ack waiting
//returns a pointer to it
//ensured to deep copy payload so can match later
//memory allocated here freed in del_pkt

int pkt_match(struct udp_pcb *pcb, struct pbuf *p, struct rdp_pkt * packet);
//helper function to find a matching packet in window
//check if rdp_pkt is a match based on payload
//logic is that if you ack the wrong packet bc same payload doesn’t really matter

void print_window();
//helper function to print window for debug