DemiBrad

// The mac address provided by TLA (The Layer Above)
short MACaddr_demibrad;
// The stream where our status outputs are printed for TLA
ostream* streamy_demibrad;
// a flag signifying that an ACK for one of our packets has been received.
bool ack_Received_demibrad;
// The address to which an ACK needs to be sent with a special case of
// zero meaning to do nothing.
short MACACK_demibrad;
// The RF layer
RF* RFLayer_demibrad;
// A queue containing all outgoing data
CircularFifo<Packet*, 10> send_Queue_demibrad;
// A queue containing all incoming data
CircularFifo<Packet*, 10> receive_Queue_demibrad;
// A buffer filled with packets to protect them from garbage collectors.
// They will eventually be overwritten.
Packet memory_buffer_demibrad[500];
// The next slot in the packet buffer to be used
int memory_buffer_number_count_demibrad;

/*
 * Initialize the Demibrad Class.
 * Starts the RF layer, a sender thread, and a receiver thread
 */
int dot11_init(short MACaddr, ostream *streamy);
/*
 * debugging garbage
 */
int dot11_command(int cmd, int val);
/*
 * Get the current status of our layer.
 */
int status();
/*
 * pop off the first item on the outgoing data queue.
 * gives back the source addres, destination address, and buffer size
 */
int dot11_recv(short *srcAddr, short *destAddr, char *buf, int bufSize);
/*
 * send a packet to the address provided, with the
 * information, starting at the specified buffer
 * of the specified size.
 */
int dot11_send(short destAddr, char *buf, int bufSize);

void *create_sender_thread(void *cnt)
void *create_and_run_receiver_thread(void *cnt)

pthread :: Listener

- short *MACaddr // users mac address
- *ostream streamy // provided ostream
- incoming_Queue *queue <short, char, int> // a pointer to the incoming data queue
- *short MACACK // a pointer to the address that is associated with the most recent Acknowledgement
- *bool ack_Received // a pointer to the flag for acknowledgment received

+ listener(): int // creates the listener thread and calls listen
- listen() : int // start listening to the RF layer, blocks until data is received
- queue_data() : int // puts the data in the incoming data queue
- read_packet() : int// reads the packet to make sure it is up to spec with our 802.11~ spec and looks to see if it is four us.

pthread :: Sender

- short *MACaddr // users mac address
- *ostream streamy // provided ostream
- outgoing_Queue *queue <short, char, int> // a pointer to the outgoing queue
- *short MACACK // pointer to the address that is associated with the most recent Acknowledgement
- *bool ack_Received // pointer to the flag for acknowledgement received
- packet pachyderm // current packet that is trying to be send

+ sender() : void // initializes the sender thread
- check_Queue_and_ACK_Bool() : bool // checks the acknowledgement flag in DemiBrad and sends and returns true if an acknowledgement needs to be sent
- check_received_ack () : int // looks to see if an ack has been received for a packet that has been sent from the sender thread
- check_send_ack : int // looks to see if the sender needs to send an ack
- send() : int // sends pachyderm as a properly formatted packet
- make_packet(enum frm, bool resen, unsigned short sn, unsigned short dest, unsigned short sendr, *char dta, int CS) : int // sets pachyderm
- resend() : int // re sends a packet

Packet

short frametype; // can either be 1-4 based on the type of frame.
bool resend; // if true, this packet is a resend packet.
unsigned short sequence_number; // the sequence number for the packet.
unsigned short destination; // the destination mac address for everything
unsigned short sender; // the sender mac address
char* data; // a pointer to an char array. It cointains the data that the user wants to send
unsigned int CRC; // currently, you can pass in the CRC, this will eventually be taken out but until CRC is implemented it will remain in.
int bytes_to_send; // this is the size of the data char array.
int frame_size; // this is the size of the frame. It is always 10 more that bytes_to_send.
char physical_data_array[2038]; // contains the physical data

// creates a packet to be sent as data
+ init (unsigned short dest, char* dta, int size, unsigned short madder) : void
// creates a packet to be received
+ Packet::initPacket(char *pac, int byts)
// builds a packet into a byte array and puts in in the buffer
+ buildPacket(char* buffer) : int
// builds an ACK packet
+ initpacketack(unsigned short dest)