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
+ dot11_init(short MACaddr, ostream *streamy): int
// debugging garbage
+ dot11_command(int cmd, int val) : int
// 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
+ dot11_recv(short *srcAddr, short *destAddr, char *buf, int bufSize) : int
// send a packet to the address provided, with the
// information, starting at the specified buffer
// of the specified size.
+ dot11_send(short destAddr, char *buf, int bufSize) : int
// Create a sender thread that will look for data in the
// outgoing data queue and send it to the specified address
- *create_sender_thread(void *cnt) : void
// create a receiver thread that receives data and puts it in
// the received data queue.
- *create_and_run_receiver_thread(void *cnt) : void

pthread :: Listener

// a pointer to our MAC address
unsigned short* MACaddrList;
// a pointer to our output stream
ostream *streamy;
// a pointer to the MAC address of the most
// recent sender of data that has not been sent an ACK yet
// or assuming that none need to be sent a
// special case of zero should be used to indicate this
unsigned short* MACACKList;
// a pointer to a boolean that indicates whether or not a ACK has been recived
bool* ackReceivedL;
//size guarenteed to hold all properly formated packets
static const int MAXPACKETSIZE = 2048;
// buffer for the incoming packets
char buf[MAXPACKETSIZE];
//a queue for the outgoing data
CircularFifo<Packet*,10>* daLoopLine;
//the reference to the RF layer
RF* daRF;
//the reference to the RF layer
int bytesReceived;

+ Listener(RF* RFLayer, CircularFifo<Packet* ,10>* incomingQueue, unsigned short* sendFlag, bool*
- receivedFlag, unsigned short myMAC);
- int queue_data();
- int read_Packet();
+ int UltraListen();

pthread :: Sender

//Pointer to the RF layer
- RF* theRF;
//Our MAC address
- short macAddr_Sender;
//ostream provided to us
- ostream* dataStream;
//A queue to check for outgoing data
- CircularFifo<Packet*,10>* infoToSend;
//Pointer to flag for received acks
- bool* ackReceived;
//Pointer to destination addr to send Ack
- unsigned short* ackToSend;
//The packet to send
- Packet pachyderm;
//The sequence number for transmitted packets
- unsigned short seqNum;
- static const unsigned short MAXSEQNUM = 4095;
// Wait time (second) to check again
// if the network is free
- static const unsigned int SLEEPTIME = 1;
//The byte array to be transmitted on RF
- char* frame;

//Looks for outgoing data packets to send
- check_QueueToSend():int
//Looks to see if an ack for the most recently
transmitted message has been received
- check_ReceivedAck():unsigned short
//Looks for destination address to send an ack to
- check_SendAck():unsigned short
- buildFrame(short frm, bool resend, unsigned short seqNum, unsigned short destAddr, char* data, int CS, int size):int;
//Transmits a frame
- send(Packet theFrame):int
// Increments the sequences
// number till MAXSEQNUM then resets
- incrementSeqNum():int
//Retransmits a frame
- resend():int

Packet

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

// 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
+ initPacket(char *pac, int byts): int
// 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) : int