Appendix A

Pseudocode of the wlan_mac Process Model in OPNET

static void wlan_frame_transmit ()
{
    char msg_string [120];
    char msg_string1 [120];
    WlanT_Hld_List_Elem* hld_ptr;
    const WlanT_Data_Header_Fields* retx_header_ptr;
    double pkt_tx_time;
    int list_high_index;
    int list_low_index;
    WlanT_Mac_Frame_Type type;
    Boolean pcf_frag_buf_empty;

    /** Main procedure to invoke function for preparing and transmitting the appropriate frames. **/
    /* Check if PCF is currently active and if time to transmit the CFP end frame. If so check if more fragments have to be transmitted. If none then, prepare to send the cfp_end frame to indicate the end of the CFP period. */
    /* Store the size of the PCF fragmentation buffer in a local variable for quick access. */
    /* Check if the transmission of the cf end frame has been enabled. If so, make sure there are no more fragments pending and the PCF fragmentation buffer is empty. */
    /* If the AP needs to ACK to a previously received frame send a CF_end_Ack frame, if not transmit CF end Allocating pool memory to the higher layer data structure type. */
    /* Generate error message and abort simulation if no memory left for data received from higher layer. */
    /* Set up dummy element to see if any more data for station currently being polled */
    /* Set search bound for pcf higher layer data queue */
    /* If a poll fail count reached the max poll fail count or the previous poll was successful and no more data from this station and last data tx was successful and no more fragments exist and no more data exist in the hlk queue */
    /* for this station then next station will start transmission */
    /* Increment polling index to next user. */
    /* Check whether the poll reached the specified limit. */
    /* Reset the relevant flags. */
    /* Set the retry count to the retry limit to drop the packet */
    /* Drop the packet. The function will also reset the counter for failed polls. */
    /* If we finished polling all the pollable STAs in the list but still have some contention free frames to send, then restart */
/* polling the pollable STAs since we still have some CFP time to go. */
/* to go. */
  /* Restart the polling. */
  /* End the CFP prematurely since we have no stations to poll */
  /* and no CF frames to send. Also send an ACK if necessary */
  /* Destroy the dummy higher layer data entry used for searching. */
}
}

/* Determine our data rate for our next poll. This is necessary */
/* if we are an 11g AP serving both 11g and 11b STAs. */
/* First check whether we are polling the same STA. */
  /* Same STA. Don't change the data rate unless */
  /* previously we polled this 11g STA with low data rate */
  /* because that poll had a piggybacked ACK for an 11b STA */
  /* Go back to the regular rate and reset the flag. */
  /* Check whether the new STA 11g enabled. */
  /* Use the regular rate for the new 11g enabled STA */
  /* unless we also need to ACK an 11b STA. */
  /* Adjust the rate and reset the flag. */
  /* We can't increase the rate for the */
  /* current transmission because we also */
  /* need to ACK an 11b STA. Set the flag so */
  /* that we adjust the data rate before */
  /* the next poll. */
  /* The new STA is a non-ERP STA (not 11g enabled). */
  /* Lower our data rate so that it can decode our */
  /* transmission. Pick the highest 11b data rate */
  /* that is lower than our regular 11g data rate. */
  /* Go on using lowered data rate since the new STA */
  /* is not 11g enabled. Reset the flag, which may be */
  /* set. */

/* Re init dummy element for new poll index. */
/* Set the destination address. */
/* First check if this is a retry. */
  /* Destroy the dummy higher layer data entry used for searching. */
  /* Set type to last frame type. */
  /* Retrieve the destination information from the frame. */
  /* Check the ACK status for retransmission. */
    /* The previous message was sent with an ACK, which */
    /* needs to be removed from this retransmission. */
  /* Perform the retransmission. */

/* Check if fragmentation buffer is empty and if there is any data to send */
/* to this station. If no data, send ack / poll as needed. */
  /* Set active poll flag since poll will be transmitted. */
  /* If the AP has a pending ACK to transmit, send Ack-CF poll frame. */
  /* If no pending ACK for this station transmit the poll frame */
  /* Destroy the dummy higher layer data entry used for searching. */

/* If we've come this far, there must be data for this user. */
/* If the fragmentation buffer is empty, get a new packet and */
/* setup fragmentation buffer. Tx of frame is queued outside this else if */
  /* First destroy the dummy higher layer data entry used for searching. */
  /* Get next packet for transmission from the higher layer queue */
/* Make sure destination address matches polling address. */
/* A packet must have been inserted into the queue by the */
/* upper layers after I started polling for a lower */
/* address. Increment an offset to track packets at the */
/* head of the queue that have missed their opportunity */
/* to transmit this CFP. Restore the packet to the */
/* point where it was stored, and get the next packet for */
/* transmission. */

/* Remove packet from higher layer queue. */
/* Setting destination address state variable. */
/* Determine packet size - required to determine fragmentation */
/* Packet seq number modulo 4096 counter. */
/* Packet fragment number is initialized. */

/* Packet needs to be fragmented if it is more than */
/* fragmentation threshold, provided fragmentation is */
/* enabled. Broadcast packets are not fragmented regardless */
/* of their sizes. */

/* Determine number of fragments for the packet */
/* and the size of the last fragment. */
/* If the remainder size is non zero it means that the */
/* last fragment is fractional but since the number */
/* of fragments is a whole number we need to transmit */
/* one additional fragment to ensure that all of the */
/* data bits will be transmitted */
/* If no fragments needed then number of */
/* packets to be transmitted is set to 1. */

/* Storing Data packet id for debugging purposes. */
/* Insert packet to fragmentation buffer. */
/* Computing packet duration in the queue in seconds */
/* and reporting it to the statistics. */
/* Printing out information to ODB. */

/* Store the arrival time of the PCF packet. */
/* Freeing up allocated memory for the data packet removed */
/* from the higher layer queue. */
/* Destroy the dummy higher layer data entry used earlier for searching. */

/* Set active poll flag since poll will be transmitted */
/* Time to transmit fragment - Retries happen automatically automatically. */
/* The order of else if statements here is very important, as */
/* the code uses it to enforce the proper preemption of various */
/* valid frame sequences while preventing the preemption of others. */

/* If not PCF, an Ack needs to be sent for the data prepare Ack for transmission */
/* Break the routine once Ack is prepared to transmit. */

/* Beacon transmission has priority unless we are in the middle of */
/* transmitting fragments of a data packet. */
/* Reset any pending responses since beacon will terminate sequence anyway */
/* Prepare beacon frame to be transmitted */
/* Break the routine once beacon prepared to transmit */

A-3
/* DCF Transmission processing */
/* Send a CTS frame if it is the type of frame we need to send a response of */
/* Break the routine if Cts or Ack is already prepared to transmit. */

/* If it is a retransmission then check which type of frame needs to be */
/* retransmitted and then prepare and transmit that frame. */
/* If the last frame unsuccessfully transmitted was an RTS or a */
/* CTS-to-self then transmit it again. */
/* If our last transmission was a data packet, then it means it was */
/* not acknowledged. Restart the transmission process. Do the same */
/* if we are resuming our retransmission after sending a beacon */
/* frame or a management frame reporting end of CFP. */

/* Check whether we need to start the retransmission with an */
/* RTS message. */
/* Retransmit the RTS frame to again contend for the data. */
/* If we are an ERP-STA, and we are not going to use an */
/* 802.11/11b data rate for the transmission data, and there */
/* are non-ERP STAs in the BSS, then we need to "use protection" */
/* by sending an RTS or CTS-to-self message. */
/* Use the "CTS-to-self" approach if the option is enabled. */
/* Even it is enabled, switch using RTS/CTS for protection, */
/* if our previous trials have failed as suggested in the */
/* 802.11g standard (section 9.2.11), since the BSS can be */
/* suffering from hidden node problem. */

/* Otherwise initiate a RTS/CTS exchange as the protection */
/* mechanism. */
/* Just retransmit the data packet if no protection is needed. */
/* We continue with the retransmission process. Either we have */
/* received the expected CTS for our last RTS before and now we */
/* can retransmit our data frame, or we moved from DCF period */
/* into PCF period and have been polled by the AP for */
/* transmission. In case of PCF, also check whether we have an */
/* ACK to append to our data packet. */

/* If higher layer queue is not empty then dequeue a packet */
/* from the higher layer and insert it into fragmentation */
/* buffer check whether fragmentation and RTS-CTS exchange */
/* is needed based on thresholds */
/* Check if fragmentation buffer is empty. If it is empty */
/* then dequeue a packet from the higher layer queue. */
/* Remove packet from higher layer queue. */

/* Determine packet size to determine later whether fragmentation */
/* and/or rts-cts exchange is needed. */
/* Setting destination address state variable */
/* Packet seq number modulo 4096 counter. */
/* Packet fragment number is initialized. */
/* Packet needs to be fragmented if it is more than */
/* fragmentation threshold, provided fragmentation is */
/* enabled. Broadcast packets are not fragmented regardless */
/* of their sizes. */
/* Determine number of fragments for the packet and the size of the last fragment. */
/* If the remainder size is non zero it means that the last fragment is fractional but since the number of fragments is a whole number we need to transmit one additional fragment to ensure that all of the data bits will be transmitted. */

/* Special case: data size is a multiple of the fragment size, so all the fragments will be the same size. To be consistent with other cases, set remainder size to the size of the last fragment */

/* If no fragments needed then number of packets to be transmitted is set to 1 */

/* Storing Data packet id for debugging purposes. Insert packet to fragmentation buffer. Computing packet duration in the queue in seconds and reporting it to the statistics Printing out information to ODB. Store the arrival time of the packet. */

/* Free up allocated memory for the data packet removed from the higher layer queue. */

/* Lower our data transmission rate, if it is an 11g data rate and we are either an AP or a STA in an IBSS, and there are non-ERP STAs in our BSS and our destination is one of them, so that it can decode our message. */
/* Check whether the destination is 11g enabled. If this is a broadcast transmission, use an 11b data rate since non-ERP STAs are present in the BSS. */
/* Pick the highest 11b data rate that is lower than our regular 11g data rate. */

/* Send RTS if RTS is enabled and packet size is more than RTS threshold. */
/* No RTS message is sent for broadcast packets regardless of their sizes. */
/* Set the flag indicating that an RTS is needed for the current frame due to its size. Prepare RTS frame for transmission. Break the routine as RTS is already prepared. */
/* Reset the flag indicating an RTS was not necessary due to current frame size. */
/* If we are an ERP-STA, and we are not going to use an 802.11/11b data rate for the transmission data, and there are non-ERP STAs in the BSS, then we need to "use protection" by sending an RTS or CTS-to-self message. */
/* Use the "CTS-to-self" approach & send a CTS msg with destination address set to our own address, if CTS-to-self option is enabled or the data packet is a broadcast packet. */
/* Otherwise initiate a RTS/CTS exchange as the protection mechanism. */
/* Exit the function. */
/* Prepare data frame to transmit. First check whether the station has been polled (if it is in CFP). */
/* If there is no data to send select frame response */
/* accordingly if we need to send an ACK back. */
/* We have data to respond to the poll. Also append the ACK */
/* if we have an ACK to respond. */
/* This is a normal DCF transmission. Prepare the frame for transmission */
static void wlan_prepare_frame_to_send (WlanT_Mac_Frame_Type frame_type)
{
    Packet* seg_pkptr;
    OpT_Packet_Size tx_datapacket_size;
    WlanT_Mac_Frame_Type type;
    int i;
    int destination_addr;
    int add_beacon_size;
    double tx_data_rate;
    double duration, mac_delay;
    double total_pk_size;
    double tx_end_time, tx_delay;
    double total_plcp_overhead;
    WlanT_Data_Header_FIELDS* pk_dhstruct_ptr;
    WlanT_Control_Header_FIELDS* pk_chstruct_ptr;
    WlanT_Beacon_Body_FIELDS* pk_bbstruct_ptr;
    Packet* wlan_transmit_frame_ptr;
    char msg_string [120];
    char frame_type_str [32];

    /** Prepare frames to transmit by setting appropriate fields in the  **/
    /** packet format for Data,Cts,Rts or Ack. If data or Rts packet needs **/
    /** to be retransmitted then the copy of the packet is resent.  **/

    /* First initialize the transmission data rate to the lowest supported */
    /* data rate, which is the data rate used for control frames. */
    /* Determine the destination address based on the type of the */
    /* transmission (PCF data transmission by AP or not). */
    /* It this is a CP period and the frame to be transmitted is a data/ACK. */
    /* Adjust the transmission data rate based on the operational speed. */
    /* Set the variable which keeps track of the last transmitted frame. */
    /* If it is a retransmission of a packet. Obtain the frame from the */
    /* the copy pointer which was stored during the previous transmission */
    /* If it is a retransmission then just transmit the previous frame */
    /* Reset header type in case Ack status has changed for frame */
    /* If retry count is non-zero means that the frame is a */
    /* retransmission of the last transmitted frame */
    /* Reset more_data bit in case queue status has changed since last transmission */
    /* If this STA has been polled, and there are additional packets remaining */
    /* Set more data bit to tell AP that STA has more packets */
    /* Printing out information to ODB. */

    /* Calculate NAV duration till the channel will be occupied by */
    /* station. The duration is SIFS time plus the ACK frame time, */
    /* which the station needs in response to the data frame (note: */
    /* no need to check for broadcast packets, since for broadcast */
    /* packets the encapsulating if condition will be never true). */

    /* Since the number of fragments for the last transmitted frame is */
    /* already decremented, there will be more fragments to transmit */
    /* if number of fragments is more than zero. */
    /* If more fragments need to be transmitted then the station */
    /* need to compute the duration until the receipt of the */
    /* the acknowledgement for the next fragment. 224 bits (header*/
    /* size) is the length of the control fields in the data */
    /* frame and needs to be accounted in the duration calculation.*/
/* Set the type of the expected response to "ACK". */
/* Station update its own nav duration during CP */
/* NAV should be updated only during the CP period */
/* During CFP NAV duration is updated only during */
/* the transmission of the beacon frames */
/* Creating transmit data packet type. */

/* Prepare data frame fields for transmission. */
/* Calculate nav duration till the channel will be occupied by */
/* station. The duration is SIFS time plus the ack frame time */
/* which the station needs in response to the data frame. For */
/* broadcast packets, the duration is zero since they are not */
/* acknowledged. */

/* If there is more than one fragment to transmit then remove */
/* fragmentation threshold size length of data from the buffer */
/* for transmission. */
/* Remove next fragment from the fragmentation buffer for */
/* transmission and set the appropriate fragment number. */
/* Indicate in transmission frame that more fragments need */
/* to be sent. */
/* Since more fragments need to be transmitted then the */
/* station need to broadcast the time until the receipt of */
/* the acknowledgement for the next fragment. 224 bits */
/* (header size) is the length of control fields in the */
/* data frame and need to be accounted for in the duration */
/* calculation. */

/* Set fragment number in packet field. */
/* Printing out information to ODB. */
/* Setting packet fragment number for next fragment to be */
/* transmitted. */
/* Remove the last fragment from the fragmentation buffer for */
/* transmission and disable more fragmentation bit. */
/* Printing out information to ODB. */

/* Setting the Header field structure. */
/** if this is the CF period and the STA has been polled */
/** then set the duration to the standard value. */
/* Duration should be set to 32768 during CFP. */
/* This is the CP, so set duration field. */

/* In the BSS network the Data frame is going from AP to sta */
/* then fromds bit is set. */
/* if in the BSS network the Data frame is going from sta to AP */
/* then tods bit is set. */
/* If Infrastructure BSS then the immediate destination */
/* will be Access point, which */
/* then forward the frame to the appropriate destination. */
/* If this STA has been polled, and there are additional packets */
/* remaining */
/* Set more data bit to tell AP that STA has more packets */
/* If we are sending the first fragment of the data fragment for the first */
/* time, then this is the end of media access duration, hence we must */
/* update the media access delay statistics. */
/* Populate the packet fields. */
/* Set the frame control field and nav duration. */
/* The actual data is placed in the Frame Body field. */
/* Add some bulk to the packet to model the transmission delay of PLCP fields accurately which are always transmitted at 1 Mbps regardless of the actual data rate used for data frames. */

/* Expect acknowledgement only for directed frames. */
/* Reset the retry count because we won't await an ACK. */
/* The retry count can be non-zero even for a broadcast frame since it can be proceeded by a CTS-to-self frame */
/* frame in an 11g WLAN, which may have been retransmitted. */

/* Due to possible earlier use of CTS-to-self frame */
/* exchange, reset the rts_sent flag. */
/* Transmission of a broadcast frame is always assumed */
/* successful. Hence, set the flag for CW backoff. */

/* Since the transmission of the higher layer packet is complete, update the queue size information and statistic. */
/* Ack frame is expected in response to data frame. */

/* Make copy of the frame before transmission -- make sure that a packet destined for broadcast addresses is not copied as that would never to destroyed (due to unACKing nature of broadcast traffic). */
/* Station update of its own nav_duration. */

/* Place the transmission data rate and physical layer technology information into the packet. */
/* Update the data traffic sent statistics. */
/* Write a value of 0 for the end of transmission. */

/* We can be sending this data message as a response to a CTS message*/
/* we received. Therefore reset the "frame respond to send" variable. */

/* If there is nothing in the higher layer data queue and fragmentation buffer */
/* then disable the data frame flag which will indicate to the station to wait for the higher layer packet. */

/* If this is a contention free period and need to send a data/ack/poll. */
/* Preserve the frame type being transmitted. */

/* Adjust the transmission data rate based on the operational speed. */
/* Set active poll flag if this is a poll frame. */

/* If it is a retransmission of a packet then no need to prepare data frame. */
/* Creating transmit data packet type. */
/* Prepare data frame fields for transmission. */
/* If there is more than one fragment to transmit and there are equal sized fragments then remove fragmentation threshold size length of data from the buffer for transmission. */
/* Remove next fragment from the fragmentation buffer for transmission and set the appropriate fragment number. */
/* Indicate in transmission frame that more fragments need to be sent */
/* if more than one fragments are left */
/* If no more fragments to transmit then set more */
/* fragment field to be 0 */
/* Set fragment number in packet field */
/* Printing out information to ODB. */
/* Setting packet fragment number for next fragment to be transmitted */
/* Remove last fragments (if any left) from the fragmentation buffer for */
/* transmission and disable more fragmentation bit. */
/* Printing out information to ODB. */
/* Set duration field */
/* During CFP the duration field should read 32768. (Section 7.1.3.2 of spec) */
/* Setting the Header field structure. */
/* In the BSS network the Data frame is going from AP to sta */
/* then from DS bit is set. */
/* if in the BSS network the Data frame is going from STA to AP */
/* then to DS bit is set. */
/* If Infrastructure BSS then the immediate destination */
/* will be Access point, which */
/* then forward the frame to the appropriate destination. */
/* If we are sending the first fragment of the data fragment for the first */
/* time, then this is the end of media access duration, hence we must */
/* update the media access delay statistics. */
/* Set the frame control field. */
/* The actual data is placed in the Frame Body field */
/* Add some bulk to the packet to model the transmission delay */
/* of PLCP fields accurately which are always transmitted at */
/* 1 Mbps regardless of the actual data rate used for data frames. */
/* Make copy of the frame before transmission */
/* If it is a retransmission then just transmit the previous frame. */
/* If retry count is non-zero means that the frame is a */
/* retransmission of the last transmitted frame. */
/* Reset header type in case Ack status has changed for frame */
/* read back duration field for debug stuff. */
/* Printing out information to ODB. */
/* Place the transmission data rate and physical layer */
/* technology information into the packet. */
/* Update the data traffic sent statistics. */
/* Write a value of 0 for the end of transmission. */
/* Only expect Acknowledgement for directed frames. */
/* ACK frame is expected in response to data frame. */
/* Reset the "frame to respond" variable since we have piggy- */
/* backed an ACK to our message if we had to send one. */
/* Preparing acknowledgement frame in response to the data frame */
/* received from the remote stations. */
/* Since an ACK is a control response frame, adjust its */
/* transmission rate based on the data rate of the data frame we */
/* are ACKing, if operating in an 11a or all-11g BSS. Otherwise use */
/* 1 Mbps, the mandatory PHY rate of 802.11/11b. */

/* Creating ACK packet format type. */
/* Adjust the packet size if necessary to model the PLCP overhead */
/* accurately, which is physical layer technology dependent. The */
/* default value is set for infra-red technology. */

/* Setting ACK frame fields. */
/* If there are more fragments to transmit then broadcast the remaining */
/* duration for which the station will be using the channel. */

/* Destination station address. */
/* Setting ACK type. */
/* Setting the accept field to true, meaning the frame is a good frame. */

/* Place the transmission data rate and physical layer */
/* technology information into the packet. */
/* Since no frame is expected, the expected frame type field to nil. */
/* Once Ack is transmitted in response to Data frame then set the frame */
/* response indicator to none frame as the response is already generated */
/* Printing out information to ODB. */

/* Update the control traffic sent statistics. */
/* Write a value of 0 for the end of transmission. */
/* Creating Rts packet format type. */
/* Initializing RTS frame fields. */
/* Type of frame */

/* if in the infrastructure BSS network then the immediate recipient for */
/* the transmitting station will always be an Access point. Otherwise the */
/* frame is directly sent to the final destination. */
/* If Infrastructure BSS then the immediate destination will be Access */
/* point, which then forward the frame to the appropriate destination. */
/* Otherwise set the final destination address. */

/* Source station address. */
/* Setting the RTS frame type. */
/* Setting the accept field to true, meaning the frame is a good frame. */

/* Setting the variable which keeps track of the last transmitted frame */
/* that needs response. */

/* Determining the size of the first data fragment or frame that need */
/* to be transmitted following the RTS transmission. */
/* If there are more than one fragment to transmit then the */
/* data segment of the first data frame will be the size of */
/* fragmentation threshold. The total packet size will be */
/* data plus the overhead (which is 224 bits). */
/* If there is one data frame to transmit then the */
/* data segment of the first data frame will be the size of */
/* the remainder computed earlier. The total packet size */
/* will be data plus the overhead (which is 224 bits). */

/* Station is reserving channel bandwidth by using RTS frame, so */
/* in RTS the station will broadcast the duration it needs to send */
/* one data frame and receive ACK for it. The total duration is the */
/* the time required to transmit one data frame, plus one CTS frame */
/* plus one ACK frame, and plus three SIFS intervals. While  
/* computing the duration, call the two macros at different lines  
/* to assure to use the correct value of the state variables within  
/* the macros.

/* Setting RTS frame fields.  
/* Place the transmission data rate and physical layer technology  
/* information into the packet.

/* Adjust the packet size to accurately model the RTS message and  
/* the PLCP overhead, which is physical layer technology dependent.  
/* The default value for PLCP overhead is set for infra-red technology

/* Station update of its own nav_duration.  
/* CTS is expected in response to RTS.  
/* Printing out information to ODB.

/* Update the control traffic sent statistics.  
/* Write a value of 0 for the end of transmission.  
/* Since we are sending this CTS message not a response, it is a  
/* CTS-to-self message used by ERP STAs (11g stations).

/* Store the type of last transmission. IMPORTANT NOTE: In case of  
/* CTS transmissions, the value of the state variable  
/* last_frametx_type is set to WlanC_Cts ONLY for CTS-to-self  
/* transmissions (i.e. it is not updated for regular CTS messages).

/* Initialize RTS frame fields.  
/* Set the destination address to own address.  
/* Determining the size of the first data fragment or frame that  
/* need to be transmitted following the CTS-to-self transmission.  
/* If there are more than one fragment to transmit then the  
/* data segment of the first data frame will be the size of  
/* fragmentation threshold. The total packet size will be data  
/* plus the overhead (which is 224 bits).  
/* If there is one data frame to transmit then the data segment  
/* of the first data frame will be the size of the remainder  
/* computed earlier. The total packet size will be data plus  
/* the overhead (which is 224 bits).

/* Compute the duration information that will be used by the  
/* recipient MACs to update their NAVs. The duration must include a  
/* SIFS time and the transmission time of the data frame that will  
/* follow this control message. Additionally, another sifs time and  
/* and an ACK transmission time must be included unless the data  
/* packet has a broadcast address, which don't require an ACK.  
/* While computing the duration, call the two macros at  
/* different lines to assure to use the correct value of the  
/* state variables within the macros.

/* Setting CTS frame type.  
/* Initialize the "Accept" field.
/* Setting CTS frame fields. */
/* We expect to receive our own CTS when sending CTS-to-self. */
/* Update the control traffic sent statistics. */
/* Write a value of 0 for the end of transmission. */
/* We need to update our own NAV. */
/* Place the transmission data rate and physical layer technology */
/* information into the packet. */
/* Send a copy of the packet to ourselves directly, since we will */
/* not receive a transmission that is made by our own transmitter. */
/* Add a very small delay to the transmission delay to guarantee */
/* that we receive the copy a moment after our transmitter */
/* completes our transmission. */

/* Printing out information to ODB. */
/* Preparing CTS frame in response to the received RTS frame. */
/* Since an CTS is a control response frame, adjust its */
/* transmission rate based on the data rate of the RTS frame we */
/* are replying, if operating in an 11a or all-11g BSS. Otherwise */
/* use 1 Mbps, the mandatory PHY rate of 802.11/11b. */

/* Creating CTS packet format type. */
/* Adjust the packet size if necessary to model the PLCP overhead */
/* accurately, which is physical layer technology dependent. The */
/* default value is set for infra-red technology. */

/* Initializing CTS frame fields. */
/* Type of frame. */
/* Destination station address. */
/* Station is reserving channel bandwidth by using RTS frame, so */
/* in RTS the station will broadcast the duration it needs to send */
/* one data frame and receive ACK for it. Just subtract the */
/* transmission of the CTS frame from updated NAV. Already waited */
/* SIFS is subtracted within "current_time". */

/* Setting CTS frame type. */
/* Initialize the "Accept" field. */
/* Setting CTS frame fields. */
/* Place the transmission data rate and physical layer technology */
/* information into the packet. */
/* Once CTS is transmitted in response to RTS then set the frame */
/* response indicator to none frame as the response is already generated */
/* No frame is expected once CTS is transmitted. */
/* Printing out information to ODB. */
/* Update the control traffic sent statistics. */
/* Write a value of 0 for the end of transmission. */
/* Create packet container for beacon body. */
/* Initialize the bit count that will be added to the size of the */
/* beacon body to represent the size of the optional beacon frame */
/* body elements. */
/* Create beacon body. */
A-14

/* Timestamp should be set to reference 1st bit of timestamp in */
/* message at antenna (11.1.2.1). To reduce processing, it is */
/* currently set for first bit of MAC frame at antenna (assuming no */
/* PHY delay). */

/* if no PCF, No beacon starts a CFP. */
/* PCF implemented. */
/* When cfp_count is computed as "0" then this beacon */
/* advertises the start of a CFP. Subtract one while finding */
/* out the transmission number of this beacon, since the first */
/* beacon is sent at "beacon_int" seconds instead of 0 seconds. */

/* Set the flag if this beacon will initiate a contention free period */
/* Set CFP period. */
/* Set CFP maximum duration. */
/* If beginning a CFP. */
/* Find time remaining in current CFP. */
/* Add the size of "CF Parameter Set" element to the beacon */
/* size, which is 8 bytes. */

/* If we are an 11g supporting AP, then set the non_erp_present bit */
/* of the beacon if there are some non-ERP STAs in our BSS. */
/* Lock the related mutex before checking the current number of */
/* non-ERP STAs in our BSS. */

/* Check whether there is a change in the count of non-ERP STAs */
/* in our BSS. */
/* We have a new non-ERP STA in our BSS. Set the flag. */
/* Increase the slot time to 20 usec and recompute the */
/* dependent parameters. */
/* Reduce the control frame data rate to 802.11/11b */
/* mandatory data rate. */
/* All the non-ERP STAs have left our BSS. Reset the flag. */

/* Decrease the slot time to 9 usec and recompute the */
/* dependent parameters. */
/* Set our data transmission rate to the original data rate, */
/* since we could be using a lower data rate to communicate */
/* with non-ERP STAs. */

/* Reselect the control frame data rate. Choose the highest */
/* mandatory data rate that is equal to or lower than the */
/* data rate specified for data transmissions. */

/* Unlock the mutex since we are done accessing the BSS info. */
/* Set the non_erp_present bit value of the beacon frame. */

/* Since we are an AP supporting 11g data rates, make sure that */
/* we transmit our beacon messages with the lowest 802.11/11b */
/* mandatory data rate so that if there are any roaming non-ERP */
/* STAs in the network that are in the scanning process, they */
/* can decode our beacons and join into our BSS. */

/* Make additions to the size of beacon due to the 11g specific */
/* elements added to the beacon frame body. 11g APs are assumed */
/* to support 12 data rates. Hence, increase the size of the */
/* "Supported Rates" by six rates (= 6 bytes) and also add an
/* "Extended Supported Rates" elements for the remaining 4
rates, which also becomes 6 bytes. Finally add the sizes of
"DS Parameter Set" and "ERP Information" elements, which are
both 3 bytes.

/* 11a-APs support 8 data rates. Adjust the beacon body size
for additional 6 rates in the "Supported Rates" element.
/* DSSS-APs support 4 data rates. Adjust the beacon body size
for additional 2 rates in the "Supported Rates" element, and
/* for the "DS Parameter Set" element.
/* Add 7 bytes to the beacon size to represent the "FH
"Parameter Set" element, which exists in beacons generated
/* by the APs using frequency-hopping PHYs.

/* If any, add the bits of optional beacon frame body elements to
/* the size of the beacon as bulk size.
/* Use data frame format for beacon frame since we need frame body.
/* Creating transmit data packet type.
/* Set destination address to broadcast since unicast not supported.

/* Prepare data frame fields for transmission.
/* During CFP the duration field should read 32768. (Section 7.1.3.2 of spec)
/* During CP should read zero since broadcast (Section 7.2.3)
/* Setting the Header field structure.

/* This value is checked at the receiving end to see if this frame was intended
/* for this BSS id
/* Management frames (Beacon) never involve DS.

/* Start setting the packet fields.
/* Set the frame control field.
/* If this is the start of the CFP, reset the NAV
/* Any frame sequences in progress will be interrupted anyway.
/* The beacon body is placed in the Packet container.
/* The beacon body "packet" is placed in the Frame Body field

/* Place the transmission data rate and physical layer
/* technology information into the packet.
/* Adjust the packet size if necessary to model the PLCP overhead
/* accurately, which is physical layer technology dependent. The
/* default value is set for infra-red technology.

/* Clear expected frame time since any existing valid frame sequences have
/* been interrupted anyway.

/* Printing out information to ODB.
/* Clear tx beacon flag.

/* Check if the frame type to be transmitted is a data null/cf ack/cf poll
/* Preserve the frame being transmitted
/* Adjust the transmission data rate based on the operational speed.
/* Set active poll flag if this is a poll frame
/* If it is a retransmission of a packet then no need of preparing data frame.
/* Creating transmit data packet type.

/* Prepare data frame fields for transmission.
/* Set packet fragment fields
/* Set duration field */
/* During PCF the duration field should read 32768. (Section 7.1.3.2 of spec) */
/* Setting the Header field structure. */
/* In the BSS network the Data frame is going from AP to STA */
/* then from DS bit is set. */
/* if in the BSS network the Data frame is going from sta to AP */
/* then to DS bit is set. */
/* If Infrastructure BSS then the immediate destination */
/* will be Access point, which */
/* then forward the frame to the appropriate destination. */

/* Set the frame control field. */
/* Need to create dummy "Frame Body" so use beacon frame and */
/* set size to zero. Create packet container for beacon body. */
/* The actual data is placed in the Frame Body field. */
/* If enabled, print out an ODB trace message. */
/* Add some bulk to the packet to model the transmission delay */
/* of PLCP fields accurately which are always transmitted at */
/* 1 Mbps regardless of the actual data rate used for data frames */

/* Make copy of the frame before transmission */
/* If it is a retransmission then just transmit the previous frame */
/* If retry count is non-zero means that the frame is a */
/* retransmission of the last transmitted frame. */
/* Read back duration field for debug stuff. */
/* Re-write the packet type since the poll message sent */
/* earlier may have a piggy-backed ACK, which will not be */
/* repeated in this retransmission. */
/* If enabled, print out an ODB trace message. */

/* Place the transmission data rate and physical layer */
/* technology information into the packet. */
/* Update the data traffic sent statistics. */
/* Write a value of 0 for the end of transmission. */
/* No ACK expected for non-data poll frames but do expect some */
/* type of Data frame in response. */
/* Once Ack is transmitted in response to Data frame then set */
/* the frame response indicator to none frame as the response is already generated */

/* Preparing Contention Free end frame if no more stations */
/* to poll or Cf_End interrupt. */
/* Creating Cf_End packet format type. */
/* Setting ack frame fields. */
/* Set duration field */
/* CF_End duration should always read zero.(Section 7.2.1.6 of spec) */
/* CF End is a broadcast, so set destination address to -1. */
/* The tx address conveys our own BSS ID in the CF-End messages. */
/* Setting frame type. */
/* Setting the accept field to true, meaning the frame is a good frame. */
/* Place the transmission data rate and physical layer technology */
/* information into the packet. */
/* Adjust the packet size to model the Cf_End message and the PLCP */
/* overhead, which is physical layer technology dependent, */
/* accurately. The default value for PLCP overhead is set for */
/* infra-red technology. Also note that the size of CF End message */
/* is equal to the size of the RTS message. */
/* Since no frame is expected, the expected frame type field to nil. */
/* No response is expected so set indicator accordingly */
/* Printing out information to ODB. */
/* Since CFP over, clean up indicators */
/* Check if a PCF beacon has been overrun before clearing pcf_active flag */
/* PCF beacon has been overrun so don't clear flag */
/* Update the control traffic sent statistics. */
/* Write a value of 0 for the end of transmission. */

/* Send packet to the transmitter. */
/* Clear ignore busy flag in case it was set. */
/* Clear PCF side traffic flag in case it was set. */
/* Clear polled flag in case it was set. */