Project Report 2

Milestones Achieved

1) Executing a test application

The applications CntToLedsAndRfm (a data transmitting application) and RfmToLeds was compiled and tested successfully.

2) Files required for implementing the Active message Layer

We observed that both the applications used a module called GenericComm which is interfaced with the AMStandard module, responsible for implementing the active message layer. It was seen that this module is used for sending active message both to the UART and the Radio. We commented out the UART part and compiled the files.

3) The data structure of the packet transmitted by the mote

```
typedef struct TOS_Msg {
    uint8_t length;
    uint8_t fcfhi;
    uint8_t fcflo;
    uint8_t dsn;
    uint16_t destpan;
    uint16_t addr;
    uint8_t type;
    uint8_t group;
    int8_t data[28];
    uint8_t strength;
    uint8_t lqi;
    bool crc;
    bool ack;
    uint16_t time;
} __attribute((packed))  TOS_Msg;
```

4) Functioning of the Active message Layer on the Mote

Data Reception

When a message is received at the MAC layer, it sends an interrupt to the Active message layer and the active message layer calls the function in the application layer to read the buffer. Once the function in the application layer returns the FIFO in the chip is cleared and is ready to receive new data.

Data Transmission

Here the application layer of the software passes the data to the active message layer along with the address and the length of the data, which in turn passes the data to the MAC layer.

5) Extracting C code from the active message layer

The C Code was extracted from the file but it was found that the transmission involved passing of the packet to the FIFO and the reception involved the generation of an interrupt to read from it. Hence we did not pursue the compilation of the code to transport it to the Zigbee board.

6) The process of message reception and transmission on the Zigbee MAC layer was studied and the key functions involved were identified.

Data reception on the M16C side:

Upon reception of a MAC data frame, the MAC layer on the M16C side issues a call to the application layer via the *mcpsDataIndication(MCPS_DATA_INDICATION *pMDI)* function. This function must be implemented by the higher application layer. The function parameter is a pointer to the data indication structure. This data can then be processed at the application layer.

Data transmission on the M16C side:

The function *mcpsDataRequest* is used to transmit a data packet at the application layer. The parameters for this function are:

addrModes- Address mode for source and destination.
srcPanId- Source PAN identifier
**pSrcAddr*- Pointer to the source address (short or extended)
destPanId- Destination PAN identifier
**pDestAddr*- Pointer to the destination address (short or extended)
msduLength- The length of pMsdu[]
***pMsdu*- **A pointer to the packet payload**
msduHandle- A handle to this packet which is used later on with *mcpsPurgeRequest* and *mcpsDataConfirm*
txOptions- Transmission options.

Upon completion of the data transmission the MAC layer issues a call to the higher application layer via the *mcpsDataConfirm* function. This function must be implemented by the application layer. The parameters for this function are:

status- The data confirm status (whether a success, failure, frame too long, security check failure etc.).
msduHandle- The handle of the confirmed frame from the *mcpsDataRequest()* function.

7) The data structure of the Zigbee MAC layer was identified.

```
typedef struct {
   BYTE    srcAddrMode;
   WORD    srcPanId;
   ADDRESS srcAddr;
   BYTE    dstAddrMode;
   WORD    dstPanId;
   ADDRESS dstAddr;
   UINT8   mpduLinkQuality;
   BOOL    securityUse;
   UINT8   aclEntry;
   INT8    msduLength;
   BYTE    pMsdu[aMaxMACFrameSize];  ← Data payload
} MCPS_DATA_INDICATION;
```

Problems Encountered

Since the TelosB mote is Zigbee compliant, we ran a test application using the mote as a transmitter and the Zigbee board as the receiver. We tried to see if the MAC layer of the Zigbee board on the M16C side received any data stream sent out by the mote. Using the KD30 debugger it was observed that when the controller code was executed on the Zigbee board, an interrupt was generated by the receiver chip for the first time, after which no interrupt was generated. Even during the first interrupt the FIFO did not seem to have any data in it.

Proposed Solution

1> Putting the code on two Zigbee boards and making them communicate with each other and identifying the execution path taken by the receiver.
2> Observing what packet type is being passed by the transmitter to the mac layer of the Zigbee and possibly implementing the same structure on the mote. We suspect that some part of the structure might be used by the MAC layer for decoding the information, a failure of which is causing the reception problem.
3> Looking at the MAC header added by the TelosB mote, to see if it conforms to the Zigbee specifications.
4> To study the initialization of the CC240 chip on both the side to see if it is the same.
5> To investigate how the data is read from the port on the Zigbee side.

Milestones

1st week (Nov 1 – 7)

We will be to try to send some data from the mote to the zigbee board which will continuously generate interrupts.

2nd week (Nov 8-14)

The next step will be to send some data from the zigbee to the mote and try to parse the various fields received.

3rd week (Nov 15-21)

We will try to send data from the zigbee board to the mote and interpret it.

4th week (Nov 22-28)

We will write a sample application for bidirectional communication between the mote and the zigbee board.

Individual Member contribution:

Trushant:

Played a key role in studying of the zigbee code and identifying the functions involved when a message is received or sent.(Milestones 5 through 7)

Manan:

Manan studied the Active message layer on the mote, extracting the C code and finding the data structure transmitted by the mote. (Milestones 1 through 4).

The updated webpage can be viewed at http://www4.ncsu.edu/~mmshah/Project.htm