CS 164 Programming Project – Winter 2005

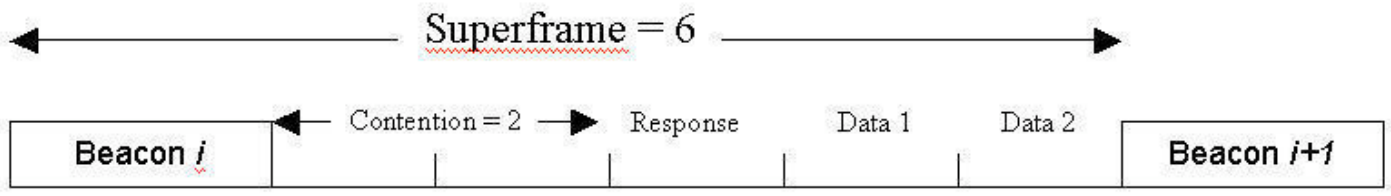
I. Introduction

The IEEE 802.15.3 standard[1] defines the data link layer functions for wireless personal area networks (WPANs), called ***piconets***. A piconet is a group of devices located within a few feet of each other that communicate over a shared high-bandwidth, short-range radio channel. Piconets are intended to replace the clutter of wires in wearable computers (i.e., replacing the serial cable that connects your PDA talking to your cell phone) or home entertainment systems (i.e., replacing the USB, firewire, DVI, or similar cables for connecting your DVD player to your HDTV). One device in the piconet, called the ***piconet controller*** or ***PNC***, serves as its "master" and is responsible for controlling the operation of the piconet. All other devices are "slaves", which must ask for permission from the PNC before they can send any data (except the request message, of course!).

In this project, you must implement a simplified version of the 802.15.3 data link layer protocol. You create an application program that represents the operation of one device in the piconet (either the PNC or a slave device), and test your program by running experiments where multiple copies of your program communicate by passing messages (represented as character strings) among themselves through Unix sockets. The format of each message is similar to the format of the corresponding frame transmitted in the 802.15.3 network. To simulate the broadcast nature of the wireless channel, each "slave" application program must establish a separate socket connection with the "master" application program, and the "master" sends a copy of each message (either locally generated by the master, or incoming from slave *i*) to all slaves (except *i*).

II. Superframes

All frame transmissions in the piconet follow a strict schedule defined by the PNC. This schedule follows a repeating pattern called a ***superframe***. A real piconet allows variable-length frames, so the schedule includes the number of microseconds allocated to each frame. However, to simplify the problem we will assume that the transmission time for every frame is some fixed constant called ***txTime***, and we will allocate one ***"slot"*** (of duration equal to one *txTime*) on the channel to the transmission of each frame. Thus, each superframe consists of a set of consecutive slots, as shown in the following diagram.



The first slot of every superframe is always a ***beacon*** frame sent by the PNC. The beacon frames are used by the PNC to support two different functions: (a) to inform all "alien" devices of the existence of a functioning piconet in this neighborhood, and (b) to tell all of its associated "slaves" of the schedule for using the channel during the remainder of this superframe. The ***contention period*** begins one *txTime* after the start of the beacon frame and includes one or more slots. The slaves can use these contention slots to send unscheduled control frames to the PNC (e.g., requests to join the piconet, or to receive time allocations in future superframes for transmitting data frames). The next slot after the contention period is a one-slot ***response period*** reserved for the PNC, where it sends responses to all requests received during the contention period. Each of the remaining slots is ***allocated to a data stream*** by the PNC, based

on time allocation requests it received during previous superframes.

In a real piconet, the transmission time for each frame is much too short (i.e., several microseconds) for you to watch the execution of the protocol as members of the piconet attempt to exchange control messages and transmit data. Furthermore, since the protocol operation is defined in terms of timers and message transfer events among multiple simultaneously-executing programs (each of which may be running on a different host), you can't use a symbolic debugger to slow things down by stepping through the code one statement at a time. Therefore, in this assignment you will ***artificially slow down the execution speed of your program by accepting txTime as a command line option***. Assume that txTime represents the transmission time for a frame in milliseconds, and that its default value is 1000 (or one second). Note that, smaller values for *txTime* are useful for reducing the execution time for a big experiment, and large values will make it easier to keep track of the order of events while you are debugging.

We will also ignore the effects of propagation delay, so you can assume that a frame that is transmitted by one device at time *t* (in units of *txTime*) will also be received by all other devices at the same time *t*. Because we have defined *txTime* to be very large compared to the actual transmission time for a frame, this assumption means that you can ignore the distinction between the starting time and ending time for a given frame transmission. In other words, suppose the PNC starts transmitting a beacon frame at time *t.* In this case, a slave wishing to transmit a request during the first contention slot should begin transmitting its control frame at time (*t*+1), which is equivalent to sleeping for one *txTime* between receiving the beacon frame and sending the control frame.

III. Frame Format

In this project, we will represent each ***frame*** as a variable-length character-string that shows how the actual (i.e., binary) frame format might be displayed by some symbolic protocol-debugging tool. For example, if the actual frame contains a 3-bit type field, then your program should represent that field as a character string of length 3, where each bit is represented as the character "0" or the character "1". We will also simplify the message format in comparison to section 7.2 of the IEEE 802.15.3 standard by skipping certain parts of the frame header, and replacing the CRC error check field (which is hard to compute in software) by a fixed-format ***end-of-frame*** sentinel, as described below. (For transparency, we will use character stuffing to prevent data in other fields within the frame from being misinterpreted as the ***end-of-frame*** sentinel.) For concreteness, we will show some examples of frames in the following discussion.

III.a. Overall Frame Structure

For this assignment, every frame will consist of exactly these three parts:

1) The ***frame header***, which consists of the following fields from beginning to end:

   a) A three-bit ***frame type***, which must be one of the following four values. (Please note that the values in Table 39 of the IEEE standard are shown with the first bit on the right. Beware!)

      i) ***beacon frame***, represented by type-string "000", which is sent only by the PNC;

ii) ***immediate ACK frame***, represented by type-string "100", which is returned by the receiver of another frame type, if requested by the sender;

iii) ***command frame***, represented by type-string "110", which is sent by a slave to the PNC during the contention period to request a future time slot allocation for sending a data frame; and

iv) ***data frame***, represented by type-string "001", which is sent by one node to any other node during a reserved time slot.

b) A one-bit ***ACK policy***, indicating that receiver should not send any ACK for this frame (if its value is "0") or is expected to send an immediate ACK (if its value is "1").

c) The ***numeric piconet ID***, a decimal number represented as a two-character string between "00" and "99".

d) The ***destination ID***, either a decimal number represented as a two-character string between "00" and "99", or the reserved *broadcast address* of "FF".

e) The ***source ID***, a decimal number represented as a two-character string between "00" and "99".

f) The ***stream ID*** for this particular session between the given source and destination, a decimal number represented as a two-character string between "00" and "99".

2) The ***payload***, which is a variable-length character string with a maximum length of 32, whose meaning depends on the frame type.

3) The ***end-of-frame sentinel*** (substituting for the CRC), which is the fixed four-character string "2005". To support data transparency, the sender and receiver will use ***character stuffing*** to prevent this string from appearing by accident in some other location within the frame. Thus, whenever the transmitter sees the string "200" anywhere within the frame, it inserts the character "6" after the second zero. For example, stuffing the string "2005" converts it to "20065", and stuffing "2006" converts it to "20066".

For example, the string "001123456789abcdef2005" (shown with the beginning on the left-hand side) would be decoded as the following frame format:

- type = "001", which indicates a data frame
- ACK policy = "1", so an immediate ACK is expected from the receiver,

- Piconet ID = 32, (remember that the string is being displayed with the beginning on the left, but the low-order digit of a number is always shown on the right)
- Destination ID = 54
- Source ID = 76
- Stream Number = 98
- Payload = "abcdef"
- End-of-frame sentinel "2005"

III.b. Beacon Frames

The PNC broadcasts a beacon frame at the beginning of each superframe. Note that the ACK policy field in the frame header is *always zero* and the destination ID is *always* "FF" for beacon frames. The payload of each beacon frame is divided into two parts:

a) ***Advertisement of global synchronization parameters*** associated with this piconet. For this assignment, we will include the following entries:

i) The ***superframe duration*** (in units of ***txTime***), from the start of this beacon frame to the start of the next beacon frame. We will use a decimal number between 1 and 15, represented as a two-character string between "10" and "51". (Don't forget the convention we're using in this document for the display of strings!)

ii) The ***length of the contention period*** (in units of ***txTime***). We will use a single decimal digit, and fix the value to "1" unless you implement the ***extra credit option*** described below. In that case, values between "2" and "8" are allowed.

b) A sequence of ***information elements.*** For this assignment, the only type of information element we consider is a ***channel time allocation*** of one slot granted by the PNC to each active stream during the current superframe. Thus, the contents of each information element consists of only the following items:

i) The ***destination ID***, either a decimal number represented as a two-character string between "00" and "99".

ii) The ***source ID***, a decimal number represented as a two-character string between "00" and "99".

iii) The ***stream ID*** for this particular session between the given source and destination, a decimal number represented as a two-character string between "00" and "99".

iv) The ***starting time offset*** (in units of *txTime*) for this allocation.

Note that all of these times are calculated relative to the start of the beacon frame, and that the PNC allocates channel time consecutively, starting from the end of the ***contention and response period***.

For example, the payload for the beacon frame in the diagram shown above might be the string "501123456420064665" (shown with the beginning on the left), which we could decode to find the following list of parameters:

- Superframe duration = 06
- Contention period = 2
- First destination ID = 21
- First source ID = 43
- First stream number = 65
- First starting offset = 4
- Second destination ID = 02
- Second source ID = 40 (notice that the string has a stuff bit in the middle of this field)
- Second stream number = 66
- Second starting offset = 5

III.c. Immediate ACK Frames

The immediate ACK does *not* contain *any* payload field. Instead, it contains only a frame header followed directly by the ***end-of-frame*** sentinel. The frame header is the same as the incoming data frame, except the ***ACK policy*** is changed from "1" to "0", and the values of the ***destination ID*** and ***source ID*** fields are swapped.

Immediate ACK frames are also unique in terms of when they are transmitted. They are *not* sent in the free-for-all contention period, nor are they given a separate allocation of channel time. Instead, each immediate ACK is transmitted right after the associated data frame before its same channel time allocation expires.

III.d. Command Frames

The 802.15.3 standard supports numerous commands. In this assignment, we will consider only a small subset from the complete list. In each case, the payload field consists of a 16-bit command type (which we will represent as a two-character string of hexadecimal digits), followed by a set of parameters unique to each command.

The ***association request*** frame (command type "00") is used by an "alien" device that wants to join a nearby piconet, which it discovers after hearing the associated beacon frames being sent

by its PNC. Note that all other frames except an association request control frame will be rejected by the PNC until the device successfully joins this piconet. This command has only one parameter, the timeout value (in units of ***txTime***), which is a decimal number represented as a two-character string between "01" and "99". If the PNC cannot detect any activity by the device during the timeout period, it will automatically drop the device from its list of piconet members. The PNC replies by sending an ***association response*** frame (command type "10") with two parameters: an adjusted timeout value (in units of ***txTime***) in case the PNC wants to reduce it to a smaller value, followed by a one-character reason code (i.e., "0" means the device has successfully joined the piconet, whereas "8" means the PNC is rejecting the request).

The ***disassociation request*** frame (command type "10") may be sent by either the PNC or the device to end its membership in the piconet. The only parameter is a one-character reason code (i.e., "0" means the timeout expired, "3" means the PNC is shutting down the entire piconet, and "4" means the device wants to remove itself from the piconet). There is no explicit response command associated with this type of request. However, the sender can force the receiver to return an immediate ACK frame if desired, by setting the ACK policy to "1" in the frame header for this command.

The ***channel time request*** frame (command type "21") is sent by a "slave" that wants permission to send one or more data frames that belong to a particular stream. In this assignment, we consider only requests for sending asynchronous data, so there is no need to consider the more complex features of the 802.15.3 protocol for handling periodic scheduling of data that belongs to a stream of time-sensitive data (e.g., real time voice or video, say). However, we will include the concept of stream IDs anyway, since the concept may be helpful for designing your PNC scheduler. Thus, for this assignment we will modify the channel time request command to use the following list of parameters after the command type field:

1) ***Number of streams*** for which this frame is requesting additional channel time, which is a decimal number represented as a single character between "1" and "4",

2) A ***request block*** for each of those streams mentioned above:

   a) A one-digit ***request number***, which is a temporary stream ID controlled by the slave to allow it to separate the responses it receives for multiple simultaneous requests with an unassigned stream ID.

   b) Its ***stream ID***, which is a two-character string that contains either the special reserved *unassigned value* "EF" if this is the first request being made for a new stream, or the previously-assigned two-digit number between "00" and "99" for an existing stream.

   c) The ***total number of data frames*** requested for this stream at this time, which is a decimal number represented as a single character between "1" and "9".

The PNC sends a separate ***channel time response*** frame (command type "31") for each request block. The response has three parameters:

1) The one-digit ***request number*** copied from the request block.

2) The associated ***stream ID***, which could have been previously assigned, or newly assigned through this response frame. If the request is rejected, the stream ID is set to the *unassigned value*.

A two-digit ***reason code***, to indicate whether the request was successful (reason code set to "0") or unsuccessful (reason code set to "21").

The entire payload field of each data frames is just application data. No parameters are required.

IV. Piconet Formation

An application running on some device may decide to instruct its 802.15.3 compatible network interface to either *create a new piconet* (and thus, as its first member, to act as its PNC, at least initially), or to *join an existing piconet*. Your program must support the same two choices by accepting the command line options "-master" or "-slave". In addition, the option "-ID nn" tells the program its two-digit ID, while the options "-localport x", "-remote_IP y", and "-remoteport z" allow it to establish the necessary socket connection(s) to establish the piconet. Finally, each slave uses the option "-f file" to find the name of a text file that contains a sequence of activities it must perform during this experiment. Each activity is represented by one line in the file, which contains the following information:

- Since the first activity is always joining the piconet, the first line is just the time at which the slave sends its association request message.
- The remaining activities are all data transfers leaving this slave, each of which includes the following items:
  - The starting time (in units of *txTime*) for this data transfer
  - The piconet ID for the receiver of this data transfer
  - The name of the file that contains the data being sent.

Note that your program can assume that whoever set up the experiment has correctly filled in all the necessary information. In particular, if the activity is to sent a particular data file to ID *x* at time *t*, then the device *x* will be connected to the piconet at time *t*.

---

[1] IEEE standards documents are downloadable for free if you answer a few simple questions. For more info, see their website: http://standards.ieee.org/getieee802/index.html