RT-Link API

September 21, 2006

Contents

1 RT-Link Protocol for Sensor Networks

RT-Link is a TDMA-based link protocol for wireless sensor networks. All packet exchanges occur in well-defined time slots. As shown in Figure 1, each fixed length time-cycle consists of 32 frames and each frame consists of 32 slots. A node may either transmit or receive within a slot. RT-Link supports software-based in-band time sync. The cycle begins when a beacon is received from the coordinator and repeats as long as a beacon update is received periodically. Nodes also implicitly pass time synchronization onto another node using the current slot in the packet header. This implicit time synchronization can cascade across multiple hops. Each slot is 6ms long, making a frame 192ms long and a cycle consisting of 1024 slots is 6.144 seconds long.

The maximum raw data rate supported by the radio transceiver on each FireFly board is 250Kbps. It therefore takes 4.096ms to send a maximum sized 128-byte packet. The remaining time in each 6ms slot is required for packet processing by the link layer and also to maintain guard time between slots.

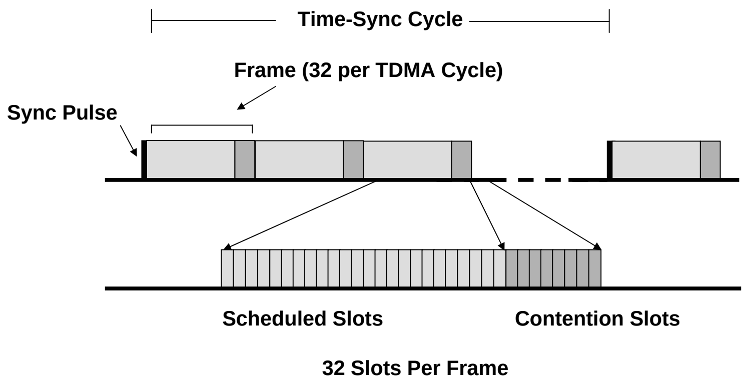

Figure 1: TDMA slots.

2 Protocol overview

RT-Link supports two types of slots: Scheduled Slots (SS) within which nodes are assigned specific transmit and receive time slots and (b) a series of unscheduled or Contention Slots (CS) where nodes, which are not assigned slots in the SS, select a transmit slot at random as in slotted-Aloha. Nodes operating in SS are provided timeliness guarantees as they are granted exclusive access of the shared channel and hence enjoy the privilege of interference-free and hence collision-free communication. While the support of SS and CS are similar to 802.15.4, RT-Link is designed for operation across synchronized multi-hop networks.

After an active slot is complete, the node schedules its timer to wake up just before the expected time of next active slot and promptly switches to sleep mode. The common packet header includes a 32-bit transmit and 32-bit receive bit-mask to indicate during which slots of a node is active.

3 Using RT-Link

```
#include <rt_link.h>
#include <rt_packet.h>

RF_TX_INFO rfTxInfo;
uint8_t tx_buf[RF_MAX_PAYLOAD_SIZE];
```

You must include rt-link and define any transmit buffers for your tasks.

3.1 Initialization

```
void rtl_init(rtl_node_mode_t mode);
// E.g. rtl_init (RTL_MOBILE);
// mode can be {RTL_MOBILE, RTL_GUEST, RTL_MEMBER, RTL_COORDINATOR}

void rtl_start();

void rtl_set_channel(uint8_t chan);
// E.g. rtl_set_channel(0x13);
```

3.2 Setting up a schedule

Contention Slots:

```
void rtl_set_contention(uint8_t slot, uint8_t rate);
```

E.g.

```
void rtl_set_contention(8, 1);  // 8 contention slots at rate 1
```

Scheduled slots:

```
uint8_t rtl_set_schedule (rtl_rx_tx_t rx_tx, uint8_t slot, uint8_t rate);
```

E.g.

```
rtl_set_schedule( RTL_RX, MY_RX_SLOT, 1 );
rtl_set_schedule( RTL_TX, MY_TX_SLOT, 1 );
```

RT-Link supports multiple rates of operation. Setting the rate index in $rtl\_set\_schedule$ or $rtl\_set\_contention$ will activate the slot on the frames based on the table below. For example, $rtl\_set\_schedule(RTL\_TX, 4, 3)$; assigns slot 4 as a transmit slot and activates it on every 4th frame. So in one cycle of 32 frames, this slot will be active on 8 frames (i.e. frames 0, 4, 8, 12, 16, 20, 24 and 28).

```
uint8_t rtl_clr_schedule (rtl_rx_tx_t rx_tx, uint8_t slot);
uint8_t rtl_get_schedule (uint8_t slot);
```

Rate Index	Frame Interval	Active Frames/ Cycles	Max. Goodput (Kbps)
0	-	0	0
1	1	32	149.3
2	2	16	74.6
3	4	8	37.3
4	8	4	18.6
5	16	2	9.3
6	32	1	4.6

Table 1: RT-Link Multi-rate support.

3.3 Checking Slot Status

```
uint8_t rtl_check_tx_status (uint8_t slot);}
```

This function returns 0 if there is no pending transmit on slot *slot* or 1 if there is a packet waiting to be sent.

```
uint8_t rtl_check_rx_status ();
```

This function returns 1 if a new packet was received and buffered, or 0 if there are no new packets.

3.4 Sending a Packet

```
uint8_t rtl_tx_packet (RF_TX_INFO *tx, uint8_t slot);
```

E.g.

```
rfTxInfo.pPayload=tx_buf;                // set the tx buffer
rfTxInfo.length=PKT_DATA_START+size;     // set the size
rtl_tx_packet( &rfTxInfo, MY_TX_SLOT );  // pass to the link layer
```

3.5 Receiving a Packet

```
if( rtl_check_rx_status()!=0 )
   {
      printf( "RX slot %d size %d: ",rtl_rx_slot,rtl_rfRxInfo.length );
      for(i=PKT_DATA_START; i<rtl_rfRxInfo.length; i++ )
         {
            putchar(rtl_rfRxInfo.pPayload[i] );
         }
      putchar('\r'); putchar('\n');

      rtl_release_rx_packet();
      // Release tells the link layer that you are done with the packet
      // so that the memory can be over written with new packets.  If
      // you do not call this function before new incomming packets arrive,
      // they will be dropped.
   }
```

3.6 Suspending on TDMA slot events

These functions are used to suspend tasks until a network event occurs.

```
rtl_wait_until_rx_packet()
uint8_t rtl_wait_until_rx_or_tx();
uint8_t rtl_wait_until_rx_packet();
uint8_t rtl_wait_until_tx_done(uint8_t slot);
uint8_t rtl_wait_until_global_slot(uint16_t slot);
uint8_t rtl_wait_until_local_slot(uint8_t slot);
```