Synchronization Layer: an Implementation Method for Energy Efficient Sensor Stack
draft-cao-lwig-syn-layer-00

Abstract

This document analyzes the problem of energy efficient protocol implementation, and proposes an energy efficient design of mulitiple protocols by utilizing a common shim layer to synchronize the packet receipt and transimission on a single node.

Status of this Memo

This Internet-Draft will expire on April 18, 2013.

Copyright Notice

Table of Contents

1. Introduction

This document analyzes the problem of energy efficient protocol implementation. And inspired by the idea originally proposed in [ref.dunkel], proposes an energy efficient design of mulitiple protocols by utilizing a common shim layer to synchronize the packet receipt and transimission on a single node.

The idea is straightforward. Because different protocol cyclings result into the frequent and out-of-order wake-up of the radio, the energy consumption would be high. By utilizing a shim layer between different applicaiton-layer protocols and the MAC/Link layer, the sending and receiving behavior of different protocols can be synchronized, which will wake up the radio at a low frequency and conserve energy consumption on the sensor.

1.1. Conventions used in this document

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL","SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119]

2. Energy Consumption Analysis

Chipset normally provides different modes of operation that consume different levels of energy. There are normally three modes of operation on a wireless chipset.

Transmit mode: The mode at a node when transmitting a packet.

Receive mode: The mode at a node when receiving a packet.

Idle mode: The mode generally used at a node with the node is neither transmitting nor receiving a packet. This mode consumes power because the node has to listen to the wireless medium continuously in order to detect a packet that it should receive, so that the node can then switch into receive mode.

Sleep mode: Sleep mode has very low power consumption. The network interface at a node in sleep mode can neither transmit nor receive packets; the network interface must be woken up to idle mode first by an explicit instruction from the node.

Sleep mode is the most energy efficient mode. Some transceivers provide different levels of sleeping mode. For example, CC3530 provides three of them, PM1, PM2 and PM3, according to its specificaiton. PM3 is the most efficient mode of them.

Some research studies the energy efficiency of different operations on the sensor. According to the investigation in [ref.icccn], the energy consumption of different operations are as follows.

```
Operation
         /\
    Hum  |*
    Temp |**
    AES  |******
         |
  RX(10B)|***********************
         |
  TX(10B)|*******************************************************
         +-------------------------------------------------------->
                                            Energy Consumption (uJ)
```

From this analysis we can clearly see that the most energy efficient way is to keep the sensor transciever sleep as much as possible.

3. Synchronization Layer Design

The key optimization direction is to reduce the frequency of radio wakeup. But if there are multiple protocols running on a sensor node, the situation is unavoidable. For example, as depicted in Figure 1 , there are three protocols running on a sensor node. The three protocols have their different rate and phase of sending and/or receiving packets. The figure shows an extreme case that their sending phases are totally un-overlapped. In this case, the sensor radio stack should be active roughly 70% of time. This makes the sensor stack not very energy efficient at all.

The idea of handling this problem is to synchronize the transmission of different protocols. We will introduce the synchronization layer and its impact in the following subsections.

```
          /|
   Prtl1   |       __             __
           |      |  |           |  |
           |______|  |___________|  |__________
   Prtl2   |          __              __
           |         |  |            |  |
           |_________|  |____________|  |_________
   Prtl3   |              __              __
           |             |  |            |  |
           |_____________|  |____________|  |_______
   Radio   |       _________       _________
           |      |         |     |         |
           |______|         |_____|         |__________
           |-------------------------------------------------->
                                                           Time
```

Figure 1: Radio Wakeup for multiple protocols

3.1. Synchronization Layer

The synchronization layer is set between the application protocols and MAC/Phy layer. The Syn-layer provides API to the upper layer protocols and has its own frequency of sending and receiving packets.

After synchronization, the protocol's transceiving behavior has been adapted. The synchronization layer keeps its own clock and will wake up the radio once the clock times up.

```
       +-------+           +-------+           +-------+
       | Prtl1 |           | Prtl2 |           | Prtl3 |
       +-------+           +-------+           +-------+
             \                 |                  /
              \_________       |       __________/
                        \      |      /
                  +-------------------------+
                  |         Syn Layer       |
                  +-------------------------+
                               ||
                  +-------------------------+
                  |      MAC/Phy Layer      |
                  +-------------------------+
```

Figure 2: The Synchronization Layer

As envisioned, after synchronization, the protocol's realtime requirement will affected. The protocol's behavior has been delayed until the Syn-layer's clock has been trigger. To support the realtime communication requirement, we can utilize a design of the API between the application-layer protocol and the Syn-layer. If the

protocol in consideration has specific realtime requirement, it can specify this requirement in the API call function.

```
      // protocol can specify the realtime requirment in the API call
      Pull (Key, Flag)
```

3.2. After Synchronization

After the synchronization layer is implemented and enabled, the packet transceiver is synchronized. As shown in Figure 3, the radio wakeup rate is reduced to about 30%.

```
   Prtl1   |       __             __
           |      |  |           |  |
           |______|  |___________|  |__________
   Prtl2   |          __              __
           |         |  |            |  |
           |_________|  |____________|  |_________
   Prtl3   |              __              __
           |             |  |            |  |
           |_____________|  |____________|  |________
   Radio   |       _________       _________
           |      |         |     |         |
           |______|         |_____|         |__________
           |                 ____              ____
   Syn     |                |    |            |    |
           |________________|    |____________|    |_____
           |
           +-------------------------------------------------->
                                                           Time
```

Figure 3: Radio Wakeup for multiple protocols

4. IANA Considerations

This document has no IANA requests.

5. Security Considerations

TBD.

6. References

6.1. Normative References

[ref.dunkel]
Dunkels, A., "The Announcement Layer: Beacon Coordination for the Sensornet Stack. In Proceedings of EWSN 2011".

[ref.icccn]
Margi, C., "Impact of Operating Systems on Wireless Sensor Networks (Security) Applications and Testbeds, ICCCN 2010".

6.2. Informative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

Author's Address

Zhen Cao
China Mobile
Xuanwumenxi Ave. No. 32
Beijing, 100871
China

Phone: +86-10-52686688
Email: zehn.cao@gmail.com, caozhen@chinamobile.com