Linux-wpan: IEEE 802.15.4 and 6LoWPAN in Linux

Linaro Connect BUD17

2017-03-06, Budapest

Stefan Schmidt
stefan@osg.samsung.com
Samsung Open Source Group
Agenda

- Motivation
- Linux-wpan Project
- Hardware
- Configuration
- Communication
- Edge Node OS
Motivation
IEEE 802.15.4

- IEEE specifications for Low-Rate Wireless Personal Area Networks
- Not only low-rate, but also low-power
- Designed for small sensors to run months/years on battery with the right duty cycle
- 127 bytes MTU and 250 kbit/s
- Sometimes confused with ZigBee as it is used as PHY and MAC layer there
6LoWPAN

- A series of IETF specifications, started in 2007
- IPv6 over LoWPAN (IEEE 802.15.4)
- Direct IP addressing of nodes
- Adaptation layer between data-link and network layer
- Address autoconfiguration
- Frame encapsulation and fragmentation
- Header compressions
6LoWPAN

- Physical and MAC layer defined by IEEE 802.15.4 from 2003 onwards (latest update from 2015)
- Series of IETF specifications from 2007 onwards (RFCs 4944, 6282, etc)
The Header Size Problem

- Worst-case scenario calculations
- Maximum frame size in IEEE 802.15.4: 127 bytes
- Reduced by the max. frame header (25 bytes): 102 bytes
- Reduced by highest link-layer security (21 bytes): 81 bytes
- Reduced by standard IPv6 header (40 bytes): 41 bytes
- Reduced by standard UDP header (8 bytes): 33 bytes
- This leaves only **33 bytes** for actual payload
- The rest of the space is used by headers (~ 3:1 ratio)

| Frame Header (25) | LLSEC (21) | IPv6 Header (40) | UDP | Payload (33) |
The Header Size Solution

- IPv6 with link-local and UDP on top
- IPHC with NHC for UDP
- The 48 bytes IPv6 + UDP header could in the best cases be reduced to 6 bytes
- That allows for a payload of *75 bytes* (~ 2:3 ratio)

<table>
<thead>
<tr>
<th>Frame Header (25)</th>
<th>LLSEC (21)</th>
<th>6</th>
<th>Payload (75)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dispatch (1)</td>
<td>LOWPAN_IPHC (1)</td>
<td>LOWPAN_NHC (1)</td>
<td>UDP Ports (1)</td>
</tr>
</tbody>
</table>
Platforms already running Linux would benefit from native IEEE 802.15.4 and 6LoWPAN subsystems
IEEE 802.15.4 transceivers can easily be added to existing hardware designs
Battery powered sensors on the other hand are more likely to run an OS like RIOT or Contiki
Example 1: Google OnHub AP which already comes with, de-activated, IEEE 802.15.4 hardware
Example 2: Ci40 Creator board as home IoT hub
Linux-wpan Project
Linux-wpan Project

- IEEE 802.15.4 and 6LoWPAN support in mainline
- Started in 2008 as linux-zigbee project, from 2012 mainline
- New project name to avoid confusion: linux-wpan
- Normal kernel development model
- Patches are posted and reviewed on the mailing list
- Small community: 2 core devs and ~4 additional people for specific drivers
- Linux-wpan mailing list (~96 people), IRC (~29 people)
Current Status

- ieee802154 layer with softMAC driver for various transceivers
- 6LoWPAN with fragmentation and reassembly (RFC 4944)
- Header compression with IPHC and NHC for UDP (RFC 6282), shared with Bluetooth subsystem
- Link Layer Security
- Testing between Linux, RIOT and Contiki
- Mainline 4.1 onwards recommended
- Active development, newer is better :-(
Hardware
Development Boards

- MIPS based Ci40 Creator (CA-8210)
- Raspberry Pi with Openlabs shield (AT86RF233)
- Transceivers can be hooked up via SPI (all drivers have devicetree bindings)
- ATUSB dongle
- 96 board with on-board transceiver?
Hardware Requirements

- Very easy to add to your current development board or product
- Free SPI port plus some additional free GPIO pins (number varies by transceiver)
- Off the shelf modules or directly added to your design
- Alternatively you could use USB
Devicetree Bindings

- Boards need devicetree support
- All our drivers have bindings
- Example for the at86rf233:

```
&spi {
    status = "okay";
    at86rf233@0 {
        compatible = "atmel,at86rf233";
        spi-max-frequency = <6000000>;
        reg = <0>;
        interrupts = <23 4>;
        interrupt-parent = <&gpio>;
        reset-gpio = <&gpio 24 1>;
        sleep-gpio = <&gpio 25 1>;
        xtal-trim = /bits/ 8 <0x0F>;
    }
};
```
Hardware Support

- Mainline drivers for at86rf2xx, mrf24j40, cc2520, atusb and adf7242
- Pending driver for ca-8210
- Driver for MCR20A in the works
- Old out of tree driver for Xbee
- ATUSB dongle to be used on your workstation
### Transceiver Comparison

<table>
<thead>
<tr>
<th>Chipset</th>
<th>Interface</th>
<th>Driver</th>
<th>2.4 GHz</th>
<th>Sub GHz</th>
<th>ARET</th>
<th>IEEE specs</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADF7242</td>
<td>SPI or PMOD</td>
<td>✔</td>
<td>✔</td>
<td>✘</td>
<td>✔</td>
<td>2003-2006</td>
</tr>
<tr>
<td>AT86RF212</td>
<td>SPI + GPIO</td>
<td>✔</td>
<td>✘</td>
<td>✔</td>
<td>✔</td>
<td>2003-2006</td>
</tr>
<tr>
<td>AT86RF212b</td>
<td>SPI + GPIO</td>
<td>✔</td>
<td>✘</td>
<td>✔</td>
<td>✔</td>
<td>2003-2011</td>
</tr>
<tr>
<td>AT86RF215</td>
<td>SPI + GPIO + LVDS</td>
<td>✘</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>2003-2011, 15.4g</td>
</tr>
<tr>
<td>AT86RF230</td>
<td>SPI + GPIO</td>
<td>✘</td>
<td>✔</td>
<td>✘</td>
<td>✔</td>
<td>2003</td>
</tr>
<tr>
<td>AT86RF231</td>
<td>SPI + GPIO</td>
<td>✔</td>
<td>✔</td>
<td>✘</td>
<td>✔</td>
<td>2003-2006</td>
</tr>
<tr>
<td>AT86RF233</td>
<td>SPI + GPIO</td>
<td>✔</td>
<td>✔</td>
<td>✘</td>
<td>✔</td>
<td>2003-2011</td>
</tr>
<tr>
<td>ATUSB (AT86RF231)</td>
<td>USB</td>
<td>✔</td>
<td>✔</td>
<td>✘</td>
<td>✔</td>
<td>2003-2006</td>
</tr>
<tr>
<td>RZUSB (AT86RF230)</td>
<td>USB</td>
<td>✔</td>
<td>✔</td>
<td>✘</td>
<td>✔</td>
<td>2003</td>
</tr>
<tr>
<td>CA8210</td>
<td>SPI + GPIO</td>
<td>✔</td>
<td>✔</td>
<td>✘</td>
<td>✔</td>
<td>2003-2006</td>
</tr>
<tr>
<td>CC2420</td>
<td>SPI + GPIO</td>
<td>✘</td>
<td>✔</td>
<td>✘</td>
<td>✔</td>
<td>2003</td>
</tr>
<tr>
<td>CC2520</td>
<td>SPI + GPIO</td>
<td>✔</td>
<td>✔</td>
<td>✘</td>
<td>✘</td>
<td>2003-2006</td>
</tr>
<tr>
<td>CC2531</td>
<td>USB</td>
<td>✘</td>
<td>✔</td>
<td>✘</td>
<td>✘</td>
<td>2003-2006</td>
</tr>
<tr>
<td>MRF24J40</td>
<td>SPI + GPIO</td>
<td>✔</td>
<td>✔</td>
<td>✘</td>
<td>✔</td>
<td>2003</td>
</tr>
<tr>
<td>XBee</td>
<td>UART</td>
<td>✘</td>
<td>✔</td>
<td>✘</td>
<td>✔</td>
<td>2003??</td>
</tr>
</tbody>
</table>
Virtual Driver

- Fake loopback driver (similar to hwsim of wireless)
- Great for testing
- Support for RIOT and OpenThread to use this when running as native Linux process
- Will help interoperability testing between the different network stacks in a virtual environment

$ modprobe fakelb numlbs=4
$ Configure e.g. Linux, RIOT, OpenThread and monitor
Configuration
Wpan-tools: iwpan

- Userspace configuration utility
- Netlink interface ideas as well as code borrowed from the iw utility
- Used to configure PHY and MAC layer parameters
- Including: channel, PAN ID, power settings, short address, frame retries, etc
- Packaged by some distributions (Fedora and Debian up to date with 0.7, Ubuntu and OpenEmbedded older versions)
Wpan-tools: wpan-ping

- Ping utility on the IEEE 802.15.4 layer
- Not a full ICMP ping replacement, but good enough for some basic testing and measurements

# run on server side
$ wpan-ping --daemon

# run on client side
Interface Bringup

- The wpan0 interface shows up automatically
- Setting up the basic parameters:
  $ ip link set lowpan0 down
  $ ip link set wpan0 down
  $ iwpan dev wpan0 set pan_id 0xabcd
  $ iwpan phy phy0 set channel 0 26
  $ ip link add link wpan0 name lowpan0 type lowpan
  $ ip link set wpan0 up
  $ ip link set lowpan0 up
Monitoring

- Setting up the interface in promiscuous mode:
  
  $ iwpan dev wpan0 del
  
  $ iwpan phy phy0 interface add monitor%d type monitor
  
  $ iwpan phy phy0 set channel 0 26
  
  $ ip link set monitor0 up
  
  $ wireshark -i monitor0

- No automatic channel hopping (you can change the channel manually in the background)
Communication
AF_INET6 Socket

- Can be used like a normal IPv6 socket
- Transparently handled in adaptation

```c
sd = socket(PF_INET6, SOCK_DGRAM, 0);
dst.sin6_family = AF_INET6;
sendto(sd, ...);
```
AF_IEEE802154 Socket

- Direct IEEE 802.15.4 communication
- Short and extended addressing schemes as well as network PAN ID handling

```c
sd = socket(PF_IEEE802154, SOCK_DGRAM, 0);
dst.family = AF_IEEE802154;
dst.addr.pan_id = 0x0023;
dst.addr.addr_type = IEEE802154_ADDR_LONG;
memcpy(&dst.addr.hwaddr, long_addr, IEEE802154_ADDR_LEN);
```
or
```
dst.addr.addr_type = IEEE802154_ADDR_SHORT;
dst.addr.short_addr = 0x0002;
sendto(sd, ...);
```
Edge Node OS
Edge Node OS

- Linux most likely to serve as border router of such networks
- Different operating system running on the smaller edge nodes of a network
- Picked the ones with IEEE 802.15.4 and 6LoWPAN support
Operating Systems

- Contiki: very fragmented project, many academic or commercial forks
- RIOT: community driven project, testing against Linux-wpan is part of the release testing process
- Zephyr: new network stack not tested yet, something for this week :-)
- mbed OS: Nanostack network stack is closed source, no incentive to test on my side
- OpenThread: open source Thread implementation
<table>
<thead>
<tr>
<th>Feature</th>
<th>Linux</th>
<th>RIOT</th>
<th>Contiki</th>
</tr>
</thead>
<tbody>
<tr>
<td>IEEE 802.15.4: data and ACK frames</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>IEEE 802.15.4: beacon and MAC command frames</td>
<td>✘</td>
<td>✘</td>
<td>✘</td>
</tr>
<tr>
<td>IEEE 802.15.4: scanning, joining, PAN coordinator</td>
<td>✘</td>
<td>✘</td>
<td>✘</td>
</tr>
<tr>
<td>IEEE 802.15.4: link layer security</td>
<td>✔</td>
<td>✘</td>
<td>✔</td>
</tr>
<tr>
<td>6LoWPAN: frame encapsulation, fragmentation, addressing (RFC 4944)</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>6LoWPAN: IP header compression (RFC 6282)</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>6LoWPAN: next header compression, UDP only (RFC 6282)</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>6LoWPAN: generic header compression (RFC 7400)</td>
<td>✘</td>
<td>✘</td>
<td>✘</td>
</tr>
<tr>
<td>6LoWPAN: neighbour discovery optimizations (RFC 6775)</td>
<td>Partial</td>
<td>✔</td>
<td>✘</td>
</tr>
<tr>
<td>RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Mesh link establishment draft</td>
<td>✘</td>
<td>✘</td>
<td>✘</td>
</tr>
</tbody>
</table>
Future
Linux-wpan Future

- Implement missing parts of the IEEE 802.15.4 specification
  - Beacon and MAC command frame support
  - Coordinator support in MAC layer and wpan-tools
  - Scanning
- Add better support for HardMAC transceivers
- Neighbour Discovery Optimizations (RFC 6775), started
- Evaluate running OpenThread on top of linux-wpan
- Configuration interface for various header compression modules
- Expose information for route-over and mesh-under protocols
Summary
Take away

- Running an IEEE 802.15.4 wireless network under Linux is not hard
- Tooling and kernel support is already there
- Border router scenario most likely use case but nodes or routers also possible
Thank you!

http://www.slideshare.net/SamsungOSG
References

- IEEE 802.15.4 specification (PHY and MAC layer)
  http://standards.ieee.org/about/get/802/802.15.html
- RFC 4944: Transmission of IPv6 Packets over IEEE 802.15.4 Networks
- RFC 6282: Compression Format for IPv6 Datagrams over IEEE 802.15.4-Based Networks
- RFC 7400: 6LoWPAN-GHC: Generic Header Compression for IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs)
- Linux-wpan source and project pages
  https://github.com/linux-wpan
  http://wpan.cakelab.org/