Samsung Open Source Group

Run Your Own 6LoWPAN Based IoT Network

OpenIoTSummit Europe

2016-10-11, Berlin

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

Agenda

- Motivation
- Linux-wpan Project
- Wpan-tools
- Hardware and Basic Setup
- Communication with RIOT and Contiki
- Link Layer Security
- Routing: Route-over and Mesh-under

Demo Show Case

- Demonstration at the ELC-E Show Cases
- Linux-wpan on a Raspberry Pi
- RIOT on Particle Photon node
- JerryScript (JS engine) on both, communicating over 6LoWPAN
- Tetris network game

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 years on battery with the right duty cycle
- 127 bytes MTU and 250 kbit/s
- PHY and MAC layers used in ZigBee

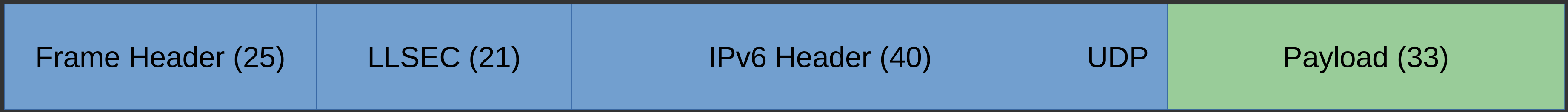
6LoWPAN

- Physical and MAC layer defined by IEEE 802.15.4 from 2003 onwards
- Series of IETF specifications from 2007 onwards (RFCs 4944, 6282, etc)

L5 Application Layer	Application	Application
L4 Transport Layer	TCP | UDP | ICMP	UDP | ICMPv6
L3 Network Layer	IP	IPv6
		6LoWPAN
L2 Data Link Layer	Ethernet MAC	IEEE 802.15.4 MAC
L1 Physical Layer	Ethernet PHY	IEEE 802.15.4 PHY

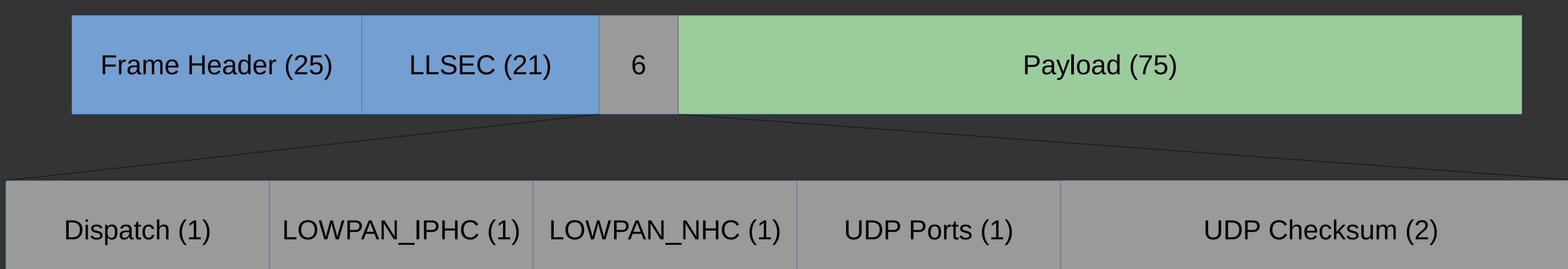
The Header Size Problem

- 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)



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)



Linux-wpan

- Platforms already running Linux would benefit from native 802.15.4 and 6LoWPAN subsystems
- 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, 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 Linux
- Started in 2008 as linux-zigbee project on SourceForge
- First steps of mainlining in 2012
- New project name to avoid confusion: linux-wpan
- New maintainer: Alexander Aring, Pengutronix
- Normal kernel development model
- Patches are posted and reviewed on the mailing list

Linux-wpan Community

- Small community: 2 core devs and ~4 additional people for specific drivers
- Linux-wpan mailing list (~94 people)
- #linux-wpan on Freenode (~25 people)
- https://github.com/linux-wpan (no PR model)
- http://wpan.cakelab.org used for wpan-tools releases

SAMSUNG
Open Source Group

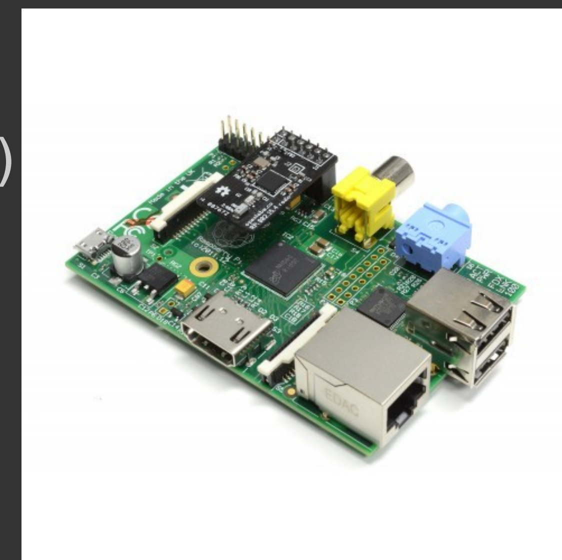
Current Status

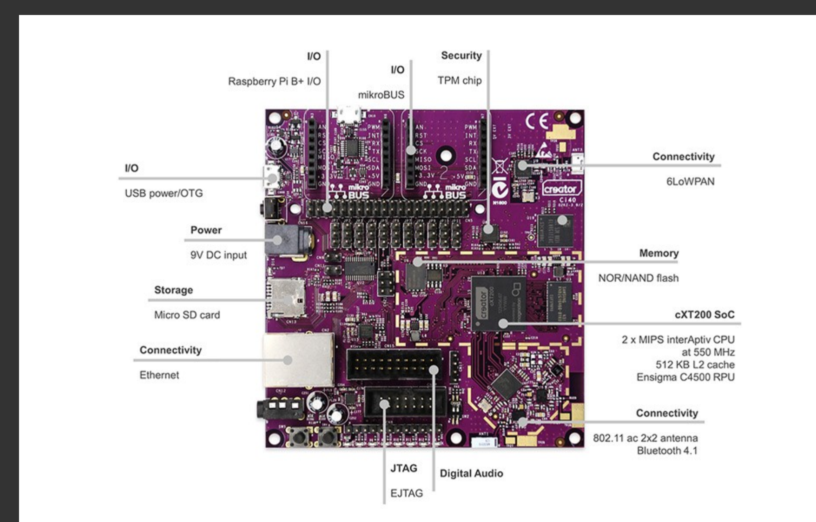
- ieee802154 layer with softMAC driver for various transceivers
- 6LoWPAN with fragmentation and reassembly (RFC 4944)
- Header compression with IPHC and NHC for UDP (RFC 6282), shared with BT subsystem
- Link Layer Security
- Testing between Linux, RIOT and Contiki
- Mainline 4.1 onwards recommended

SAMSUNG
Open Source Group

Development Boards

- Ci40 Creator (CA-8210)
- Raspberry Pi with Openlabs shield (AT86RF233)
- ARTIK 5/10 (802.15.4 network soc)
- Various transceivers can be hooked up via SPI (all drivers have devicetree bindings)
- ATUSB USB dongle



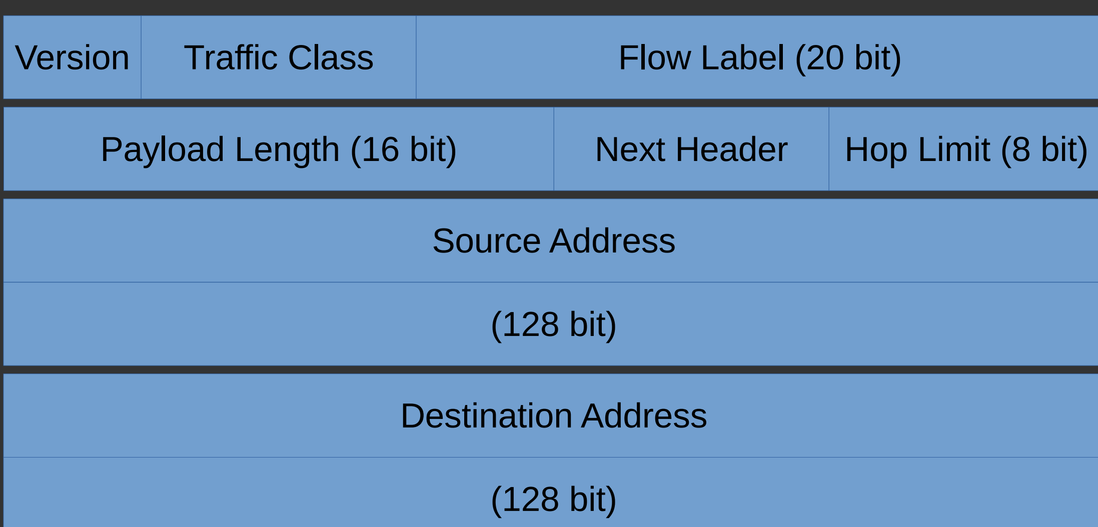


6LoWPAN Fragmentation

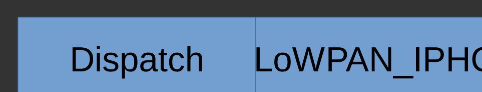
- IPv6 requires the link to allow for a MTU of at least 1280 bytes
- This is impossible to handle in the 127 bytes MTU of IEEE 802.15.4
- 6LoWPAN 11 bit fragmentation header allows for 2048 bytes packet size with fragmentation
- But fragmentation can still lead to bad performance in lossy networks, best to avoid it in the first place

IPv6 Header Compression (IPHC)

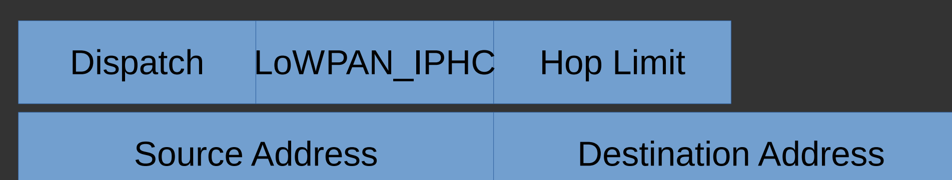
- Defining some default values in IPv6 header
  - Version == 6, traffic class & flow-label == 0, hop-limit only well-known values (1, 64, 255)
  - Remove the payload length (available in 6LoWPAN fragment header or data-link header)
- IPv6 stateless address auto configuration based on L2 address
  - Omit the IPv6 prefix (global known by network, link-local defined by compression (FE80::/64)
  - Extended: EUI-64 L2 address use as is
  - Short: pseudo 48 bit address based short address: PAN_ID:16 bit zero:SHORT_ADDRESS



6LoWPAN Header IPHC link-local (2 bytes)



6LoWPAN Header IPHC multi-hop (7 bytes)



Next Header Compression

- NHC IPv6 Extension Header compression (RFC6282)
  - Hop-by-Hop, Routing Header, Fragment Header, Destination Options Header, Mobility Header
- NHC UDP Header compression (RFC6282)
  - Compressing ports range to 4 bits
  - Allows to omit the UDP checksum for cases where upper layers handle message integrity checks
- GHC: LZ-77 style compression with byte codes (RFC7400)
  - Appending zeroes, back referencing to a static dictionary and copy
  - Useful for DTLS or RPL (addresses elided from dictionary)

Wpan-tools

lwpan

- 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 setting, short address, frame retries, etc
- Version 0.7 with network namespace support released two weeks ago
- Packaged by some distributions (Fedora and Debian up to date, Ubuntu on 0.5, OpenSUSE, Gentoo, Arch, etc missing)

Wpan-ping

- Ping utility on the 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

  $ wpan-ping –-count 100 –extended –-address 00:11:22:33:44:55:66:77

SAMSUNG
Open Source Group

Hardware and Basic Setup

SAMSUNG
Open Source Group

Hardware Support

- Mainline drivers for at86rf2xx, mrf24j40, cc2520, atusb and adf7242
- Pending driver for ca-8210
- Old out of tree driver for Xbee
- Most transceiver easy to hook up to SPI and some GPIOs
- ATUSB available as USB dongle to be used on your normal workstation (sold out but a new batch is being produced)

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-tpio = <&gpio 25 1>;
        xtal-trim = /bits/ 8 <0x0F>;
    };
};
```

SAMSUNG
Open Source Group

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 interop testing between the different network stacks in an virtual environment

$ modprobe fakelb numlbs=4

$ Configure for Linux, RIOT, OpenThread and monitor

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

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 with RIOT & Contiki

Samsung Open Source Group

RIOT

- “The friendly Operating System for the Internet of Things” (LGPL)
- Testing against Linux-wpan part of the release testing process for RIOT
- Active developer discussions and bug fixing between projects

Contiki

- “The Open Source OS for the Internet of Things” (BSD)
- Very fragmented project
- Sadly many forks for academic or commercial purpose which have a hard time to get merged
- Still an important role as IoT OS for tiny devices

Samsung Open Source Group

Comparison

Feature	Linux	RIOT	Contiki
IEEE 802.15.4: data and ACK frames	✔	✔	✔
IEEE 802.15.4: beacon and MAC command frames	✘	✘	✘
IEEE 802.15.4: scanning, joining, PAN coordinator	✘	✘	✘
IEEE 802.15.4: link layer security	✔	✘	✔
6LoWPAN: frame encapsulation, fragmentation, addressing (RFC 4944)	✔	✔	✔
6LoWPAN: IP header compression (RFC 6282)	✔	✔	✔
6LoWPAN: next header compression, UDP only (RFC 6282)	✔	✔	✔
6LoWPAN: generic header compression (RFC 7400)	✘	✘	✘
6LoWPAN: neighbour discovery optimizations (RFC 6775)	Partial	✔	✘
RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks	✔	✔	✔
Mesh link establishment draft	✘	✘	✘

Others

- Mbed OS from ARM: network stack is closed source so nothing to test against
- Zephyr: network stack from Contiki used right now but a new one is planned
- OpenThread: Open Source implementation of the Thread protocol

Link Layer Security

SAMSUNG Open Source Group

Link Layer Security

- Specified by IEEE 802.15.4
- It defines confidentiality (AES-CTR), integrity (AES CBC-MAC) and encryption and authentication (AES CCM) security suites
- Key handling, key exchange, roll over, etc is not defined
- Tested Linux against Linux and Contiki 3.0
- No way to test against RIOT as they have no LLSEC support right now

LLSEC Linux-wpan

- Needs the llsec branch in wpan-tools for configuration
- CONFIG_IEEE802154_NL802154_EXPERIMENTAL

$ iwpan dev wpan0 set security 1

$ iwpan dev wpan0 key add 2 $KEY 0 $PANID 3 $EXTADDR

$ iwpan dev wpan0 seclevel add 0xff 2 0

$ iwpan dev wpan0 device add 0 $PANID $SHORTADDR $EXTADDR 0 0

LLSEC Contiki 3.0

- You need the following Contiki build options configured in your project-conf.h to make use of LLSEC with network wide key:

```
#define NETSTACK_CONF_LLSEC noncoresec_driver
#define LLSEC802154_CONF_SECURITY_LEVEL FRAME802154_SECURITY_LEVEL_ENC_MIC_32

#define NONCORESEC_CONF_KEY {   \
0x00, 0x01, 0x02, 0x03, \
0x04, 0x05, 0x06, 0x07, \
0x08, 0x09, 0x0A, 0x0B, \
0x0C, 0x0D, 0x0E, 0x0F, \
}
```

Routing: Mesh-under and Route-over

Mesh-under

- Allows fast forwarding of packets in a mesh without travelling the IP stack
- IEEE 802.15.4 does not include mesh routing in the MAC specification
- Thus the mesh implementations sit above the MAC but below the network layer
- Various (proprietary) implementations
- 6LoWPAN specification has a field for mesh headers
- No support in Linux-wpan for mesh header as of now
- Lost fragments of bigger packets will cause troubles
- Mesh Link Establishment draft at IETF

SAMSUNG
Open Source Group

RPL

- IPv6 Routing Protocol for Low-Power and Lossy Networks (RFC6550)
- Route over protocol
- Implementations in RIOT and Contiki
- Unstrung as Linux userspace reference
- Bit rotted in-kernel RPL demo patches out there

Future

Linux-wpan Future

- Implement missing parts of the 802.15.4 specification
  - Beacon and MAC command frame support
  - Coordinator support in MAC layer and wpan-tools
  - Scanning
- Improve existing drivers and add support for new hardware
- 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

- Pictures
- http://downloads.qi-hardware.com/people/werner/wpan/web/atusb-pcba-small.jpg
- https://creativecommons.org/licenses/by-sa/3.0/