The ieee802154 and 6lowpan Kernel Subsystems

LPC 2019
IoT Microconference

2019-09-09, Lisbon, Portugal

Stefan Schmidt
s.schmidt@samsung.com
Samsung Open Source Group

Who am I

- FOSS Contributor since 2006
- EFL Developer and Release Manager
- Linux Kernel ieee802154 subsystem maintainer
- Team lead Open Source Group Samsung Research UK

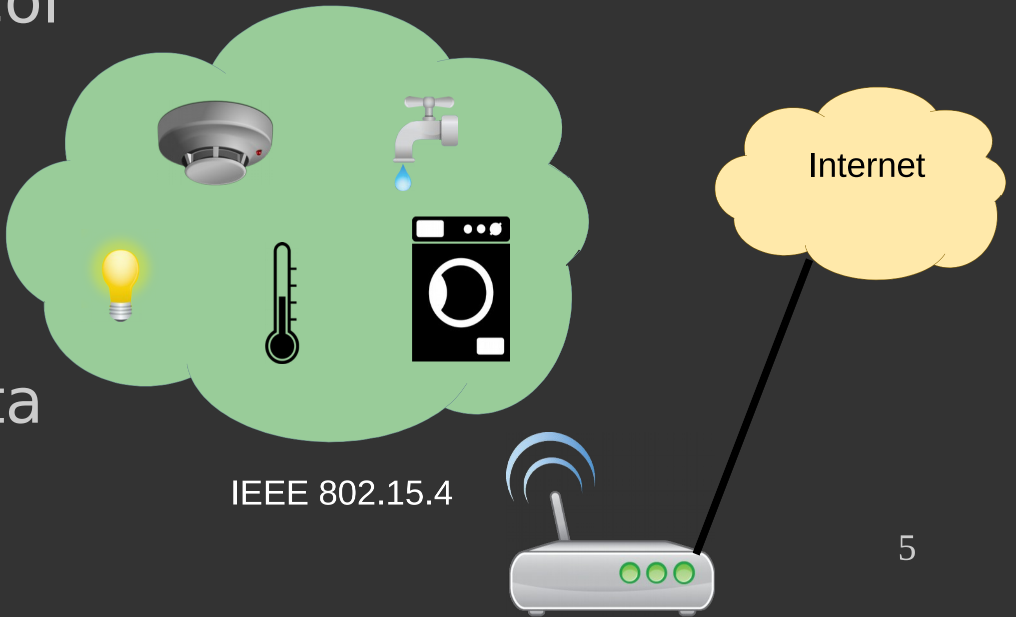
Agenda

- IPv6 over LoWPAN
- Routing
- Linux-wpan
- RTOS Systems

IPv6 over LoWPAN (6LoWPAN)

Motivation 6LoWPAN

- Things might have restricted wireless connectivity
- Using IPv6 instead of something proprietary allows the **usage of existing and proven protocols** driving the Internet
- **But** unmodified TCP/IP protocol headers can clash with MTU limitations
- Things often only need to transfer small amounts of data



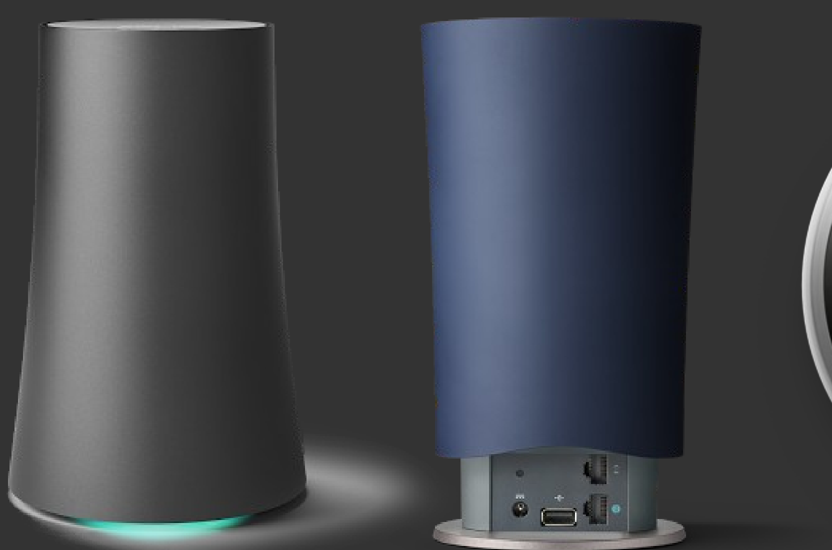
Movement Towards IP

- Many company grown network stacks moving towards IP
- Switching to make use of the success of IP
- The name Internet of Things already implies that it should be modelled after the Internet
- Direct addressing of nodes and re-use of proven protocols
- But TCP/IP is not one size fits all
- **Adaptations needed for size, reduce of header overhead, UDP to avoid latencies, etc**

Products

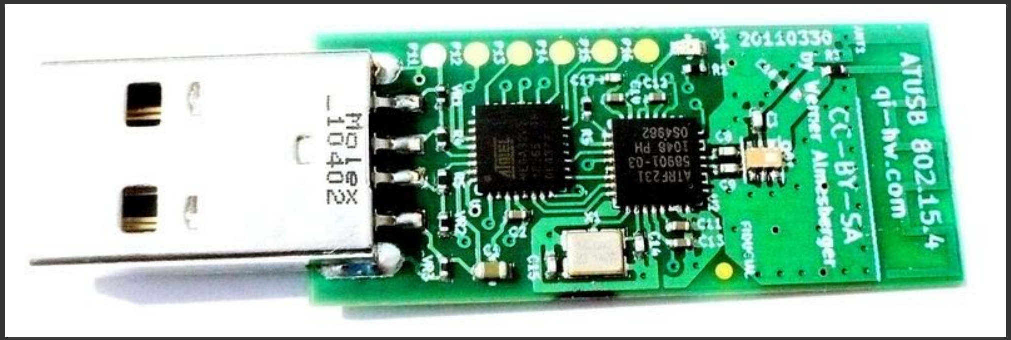
- Products with IEEE 802.15.4 transceivers
- Using 6LoWPAN or some version of Thread
- Nest Thermostat and Protect
- Google WiFi / OnHub router
- IKEA Tradfri system

THREAD



Development Boards

- Ci40 Creator (CA-8210)
- Raspberry Pi’s with Openlabs shield (AT86RF233)
- Transceivers can be hooked up via SPI
  (drivers have devicetree bindings)
- ATUSB USB dongle



IEEE 802.15.4

- IEEE specifications for Low-Rate Wireless ~~Personal Area~~ Networks (LoWPAN)
- Not only low-rate, but also low-power
- PHY and MAC layer with star and peer-to-peer topologies
- Addressing but no routing defined
- Mesh routing possible with layers on top
- Designed for small sensors to run months/years on battery with the right duty cycle
- **127 bytes MTU** and 250 kbit/s
- Often mixed-up with ZigBee as it is used as PHY and MAC layer
- Compared to Bluetooth it is older than BTLE and less complex

6LoWPAN

- Physical and MAC layer defined by IEEE 802.15.4
- Series of IETF specifications from 2007 onwards (RFCs 4944, 6282, etc)
- Goal was to use IPv6 in sensor networks based on IEEE 802.15.4
- Direct IP addressing of nodes

- Adaptation layer between data-link and network layer
- **Address auto-configuration**
- **Frame encapsulation and fragmentation**
- **Header compressions**

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

Address Auto-configuration & Fragmentation

Stateless address auto-configuration:

- Used for IPv6 networks without DHCP
- Based on layer 2 address
- Extended address uses EUI-64 as is
- Short address uses EUI-48 to EUI-64 mapping
  (16 Bit PAN+16 Bit zero+16 Bit short address)

Fragmentation:

- IPv6 requires the link to allow for a MTU of at least 1280 bytes
- Impossible to handle in the 127 bytes MTU of IEEE 802.15.4
- 6LoWPAN adds a 11 bit fragmentation header allows for 2048 bytes
- Fragmentation should still be avoided for best performance

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)

Frame Header (25)	LLSEC (21)	6	Payload (75)

Dispatch (1)	LOWPAN_IPHC (1)	LOWPAN_NHC (1)	UDP Ports (1)	UDP Checksum (2)

More 6lo Adaptations

- Specifications work in progress for other L2 technologies
- Handling of different address or MTU sizes (fragmentation)
- IPHC and other compressions as the main benefit
- IPv6 over Bluetooth LE (RFC7668), 6lowpan shared with BlueZ
- NFC
- DECT Ultra Low Energy
- PowerLine (PLC)
- 6loBAC: Token passing network on RS-485
- 802.11ah: low energy and long distance WiFi

Routing: Mesh-under and Route-over

Mesh-under

- Allow for fast forwarding of packets without travelling the IP stack
- IEEE 802.15.4 does not include mesh routing in the MAC specification
- Thus the mesh implementations is an extra layer above the MAC but below the network layer
- Various (proprietary) implementations (e.g. WirelessHART, ZigBee mesh, RF mesh, etc)
- IEEE 802.15.5 can also to be used for mesh on top of 15.4
- 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

RPL

- Routing protocol for low power and lossy networks
- IETF approach, route over protocol
- IPv6 Routing Protocol for Low-Power and Lossy Networks (RFC6550, RFC6553)
- RPL uses option in the Hop-by-Hop header of IPv6
- Constructs a directed acyclic graph in an attempt to minimize routing costs
- Implementations in RIOT, Contiki, Zephyr, etc
- Unstrung as Linux user-space reference
- Rpld as alternative released two weeks ago https://github.com/linux-wpan/rpld

Thread

- Mesh network specification from Thread Group
- OpenThread implementation from NestLabs
- Routing Information Protocol (RIP) algorithms are used, but not RIP itself
- Distribution of route information is handled by mesh link establishment (MLE, IETF draft)
- MLE allows router to update the tables of routing costs periodically in a compressed form
- Due to MLE no on-demand route discovery is needed

Linux-wpan

Samsung Open Source Group

Why linux-wpan?

- Goal: IEEE 802.15.4 and 6LoWPAN support in mainline
- 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 (SPI + few GPIOs)
- Battery powered sensors are more likely to run an OS like RIOT, Contiki or Zephyr, but they need a border router
- Started in 2008 as linux-zigbee project, from 2012 mainline (renamed to linux-wpan)

Current Status

- 6LoWPAN with fragmentation and reassembly (RFC 4944)
- Header compression with IP header compression (IPHC) and next header compression (NHC) for UDP

  (RFC 6282), shared with Bluetooth subsystem
- ieee802154 layer with softMAC drivers for at86rf2xx, mrf24j40, cc2520, atusb, adf7242, ca8210 and mcr20a
- Hwsim virtual driver module for testing
- USB dongle to be used on your workstation
- Link Layer Security

Interface Bringup

- The wpan0 interface shows up automatically
- Ieee802154 specific configuration over netlink, e.g. with wpan-tools
- 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 (changing the channel manually in the background is possible)

AF_INET6 Socket

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

```
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

```
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, ...);
```

SAMSUNG
Open Source Group

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
- Configuration interface for various header compression modules
- Expose information for route-over and mesh-under protocols (started with LQI already)
- Work on rpld to understand and support route over use cases

RTOS Systems

RTOS Systems

- Various real time operating systems support IEEE 802.15.4 and 6lowpan
- RIOT
- Contiki
- Zephyr
- OpenThread
- MbedOS (nanostack finally open source from mbed-os-5.7 onwards)

Comparison

Feature	**Linux**	**RIOT**	Contiki	Zephyr	Thread
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	✓	✓	✓	✓	✓
6LoWPAN: IP header compression (RFC 6282)	✓	✓	✓	✓	✓
6LoWPAN: next header compression, UDP only (RFC 6282)	✓	✓	✓	✓	✓
6LoWPAN: generic header compression (RFC 7400)	✗	✗	✗	✗	✗
6LoWPAN: neighbor discovery optimizations (RFC 6775)	Partial	✓	✗	✗	✗
RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks	✓	✓	✓	✓	✗
Mesh link establishment draft	✗	✗	✗	✓	✓

Take away

- 6LoWPAN and associated specifications allow the use of IPv6 on constrained networks
- Running a IEEE 802.15.4 wireless network is not hard
- Linux tooling and kernel support is already there
- Various RTOS implementations to choose from as counterparts on a simple Thing

Thank you!

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
  https://tools.ietf.org/html/rfc4944
- RFC 6282: Compression Format for IPv6 Datagrams over IEEE 802.15.4-Based Networks
  https://tools.ietf.org/html/rfc6282
- RFC 7400: 6LoWPAN-GHC: Generic Header Compression for IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs)

- Linux-wpan source (wpan-tools & rpld) and project pages
  https://github.com/linux-wpan
  http://wpan.cakelab.org/

Bonus Slides

Bonus: Homemade 6lowpan network

- For the makers and hackers
- IKEA Tradfri system based on ieee802154 and Zigbee Light Link protocol
- Cheap 15.4 based equipment (light bulbs, dimmer buttons, remotes, motion sensors)
- The cheapest light bulb (~10 USD) can be opened and flashed without destroying it
- Initial support for the chip in RIOT-OS opens up many possibilities for your own homemade 6lowpan network
- Silicon Labs EFR32 microcontroller (ARM Cortex M4, 256 kB flash, 32 kB RAM)
- No playground for Linux, but OpenThread should be possible
- https://github.com/basilfx/TRADFRI-Hacking/

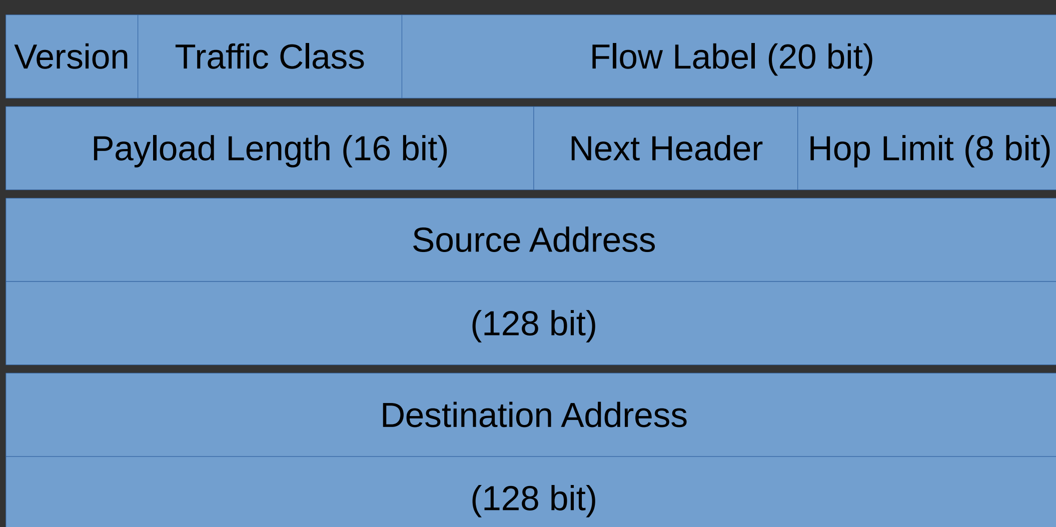
IPv6 Header Compression (IPHC)

IPHC (RFC6282)

- Deprecates HC1 & HC2 compressions from RFC4944
- Better compression for global and multicast address, not only link-local
- Compress header fields with common values: version, traffic class, flow label, hop limit
- 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

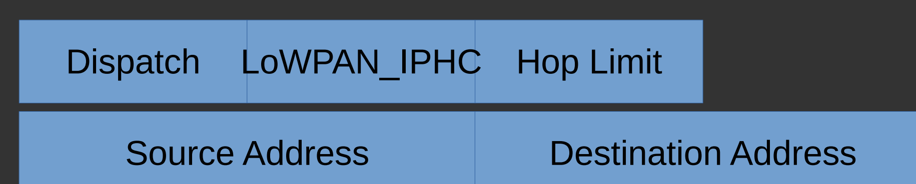
IPHC & NHC

- 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
  - Prefixes can be shared through context ID table (e.g. subnet of your cloud infrastructure)
  - Extended: EUI-64 L2 address use as is, short: pseudo 48 bit address



6LoWPAN Header IPHC link-local (2 bytes)

6LoWPAN Header IPHC multi-hop (7 bytes)



Generic Header Compression

- Generic approach instead of defining a scheme for each header
- Plugging into NHC
- Useful for header like payload e.g. DTLS or RPL (addresses elided from dictionary)
- 6CIO option in neighbour discovery messages to indicate support
- LZ-77 style compression with byte codes (RFC7400)
  - Appending zeroes, back referencing to a static dictionary and copy

Samsung Open Source Group

Kselftest support

- Hwsim will be hooked up with kselftest to give an easy way of regression testing
- Will be used in review process
- Can be used when doing a bigger network stack re-work
- Basic suite of tests to start with (ieee802154 frames in different sizes, 6lowpan packets in different sizes, header compression on and off)