IEEE 802.15.4 Developer's Guide

Introduction

The IEEE 802.15.4 working group focuses on standardization of the bottom two layers: Medium Access Control (MAC) and Physical access (PHY). And there are mainly two options available for upper layers:

- ZigBee - proprietary protocol from the ZigBee Alliance
- 6LoWPAN - IPv6 networking over low rate personal area networks

The goal of the Linux-wpan is to provide a complete implementation of the IEEE 802.15.4 and 6LoWPAN protocols. IEEE 802.15.4 is a stack of protocols for organizing Low-Rate Wireless Personal Area Networks.

The stack is composed of three main parts:

- IEEE 802.15.4 layer; We have chosen to use plain Berkeley socket API, the generic Linux networking stack to transfer IEEE 802.15.4 data messages and a special protocol over netlink for configuration/management
- MAC - provides access to shared channel and reliable data delivery
- PHY - represents device drivers

Socket API

```
int sd = socket(PF_IEEE802154, SOCK_DGRAM, 0);
```

The address family, socket addresses etc. are defined in the include/net/af_ieee802154.h header or in the special header in the userspace package (see either https://linux-wpan.org/wpan-tools.html or the git tree at https://github.com/linux-wpan/wpan-tools).

6LoWPAN Linux implementation

The IEEE 802.15.4 standard specifies an MTU of 127 bytes, yielding about 80 octets of actual MAC payload once security is turned on, on a wireless link with a link throughput of 250 kbps or less. The 6LoWPAN adaptation format [RFC4944] was specified to carry IPv6 datagrams over such constrained links, taking into account limited bandwidth, memory, or energy resources that are expected in applications such as wireless Sensor Networks. [RFC4944] defines a Mesh Addressing header to support sub-IP forwarding, a Fragmentation header to support the IPv6 minimum MTU requirement [RFC2460], and stateless header compression for IPv6 datagrams (LOWPAN_HC1 and LOWPAN_HC2) to reduce the relatively large IPv6 and UDP headers down to (in the best case) several bytes.

In September 2011 the standard update was published - [RFC6282]. It deprecates HC1 and HC2 compression and defines IPHC encoding format which is used in this Linux implementation.

All the code related to 6lowpan you may find in files: net/6lowpan/* and net/ieee802154/6lowpan/*

To setup a 6LoWPAN interface you need: 1. Add IEEE802.15.4 interface and set channel and PAN ID; 2. Add 6lowpan interface by command like: # ip link add link wpan0 name lowpan0 type lowpan 3. Bring up 'lowpan0' interface

Drivers

Like with WiFi, there are several types of devices implementing IEEE 802.15.4. 1) 'HardMAC'. The MAC layer is implemented in the device itself, the device exports a management (e.g. MLME) and data API. 2) 'SoftMAC' or just radio. These types of devices are just radio transceivers possibly with some kinds of acceleration like automatic CRC computation and comparation, automagic ACK handling, address matching, etc.

Those types of devices require different approach to be hooked into Linux kernel.

HardMAC

See the header include/net/ieee802154_netdev.h. You have to implement Linux net_device, with .type = ARPHRD_IEEE802154. Data is exchanged with socket family code via plain sk_buffs. On skb reception skb->cb must contain additional info as described in the struct ieee802154_mac_cb. During packet transmission the skb->cb is used to provide additional data to device's header_ops->create function. Be aware that this data can be overridden later (when socket code submits skb to qdisc), so if you need something from that cb later, you should store info in the skb->data on your own.

To hook the MLME interface you have to populate the ml_priv field of your net_device with a pointer to struct ieee802154_mlme_ops instance. The fields assoc_req, assoc_resp, disassoc_req, start_req, and scan_req are optional. All other fields are required.

SoftMAC

The MAC is the middle layer in the IEEE 802.15.4 Linux stack. This moment it provides interface for drivers registration and management of slave interfaces.

NOTE: Currently the only monitor device type is supported - it's IEEE 802.15.4 stack interface for network sniffers (e.g. WireShark).

This layer is going to be extended soon.

See header include/net/mac802154.h and several drivers in drivers/net/ieee802154/.

Fake drivers

In addition there is a driver available which simulates a real device with SoftMAC (fakelb - IEEE 802.15.4 loopback driver) interface. This option provides a possibility to test and debug the stack without usage of real hardware.

Device drivers API

The include/net/mac802154.h defines following functions:

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\networking\[linux-master][Documentation][networking]ieee802154.rst, line 120)

Unknown directive type "c:function".

```
.. c:function:: struct ieee802154_dev *ieee802154_alloc_device (size_t priv_size, struct ieee802154_ops
```

Allocation of IEEE 802.15.4 compatible device.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\networking\[linux-master][Documentation][networking]ieee802154.rst, line 124)

Unknown directive type "c:function".

```
.. c:function:: void ieee802154_free_device(struct ieee802154_dev *dev)
```

Freeing allocated device.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\networking\[linux-master][Documentation][networking]ieee802154.rst, line 128)

Unknown directive type "c:function".

```
.. c:function:: int ieee802154_register_device(struct ieee802154_dev *dev)
```

Register PHY in the system.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\networking\[linux-master][Documentation][networking]ieee802154.rst, line 132)

Unknown directive type "c:function".

```
.. c:function:: void ieee802154_unregister_device(struct ieee802154_dev *dev)
```

Freeing registered PHY.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\networking\[linux-master][Documentation][networking]ieee802154.rst, line 136)

Unknown directive type "c:function".

```
.. c:function:: void ieee802154_rx_irqsafe(struct ieee802154_hw *hw, struct sk_buff *skb, u8 lqi)
```

Telling 802.15.4 module there is a new received frame in the skb with the RF Link Quality Indicator (LQI) from the hardware device.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\networking\[linux-master][Documentation][networking]ieee802154.rst, line 141)

Unknown directive type "c:function".

```
.. c:function:: void ieee802154_xmit_complete(struct ieee802154_hw *hw, struct sk_buff *skb, bool ifs_ha
```

Telling 802.15.4 module the frame in the skb is or going to be transmitted through the hardware device

The device driver must implement the following callbacks in the IEEE 802.15.4 operations structure at least:

```
struct ieee802154_ops {
     ...
     int     (*start)(struct ieee802154_hw *hw);
     void    (*stop)(struct ieee802154_hw *hw);
     ...
```

```
        int     (*xmit_async)(struct ieee802154_hw *hw, struct sk_buff *skb);
        int     (*ed)(struct ieee802154_hw *hw, u8 *level);
        int     (*set_channel)(struct ieee802154_hw *hw, u8 page, u8 channel);
        ...
};
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\networking\[linux-master][Documentation][networking]ieee802154.rst, line 160)

Unknown directive type "c:function".

```
.. c:function:: int start(struct ieee802154_hw *hw)
```

Handler that 802.15.4 module calls for the hardware device initialization.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\networking\[linux-master][Documentation][networking]ieee802154.rst, line 164)

Unknown directive type "c:function".

```
.. c:function:: void stop(struct ieee802154_hw *hw)
```

Handler that 802.15.4 module calls for the hardware device cleanup.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\networking\[linux-master][Documentation][networking]ieee802154.rst, line 168)

Unknown directive type "c:function".

```
.. c:function:: int xmit_async(struct ieee802154_hw *hw, struct sk_buff *skb)
```

Handler that 802.15.4 module calls for each frame in the skb going to be transmitted through the hardware device.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\networking\[linux-master][Documentation][networking]ieee802154.rst, line 173)

Unknown directive type "c:function".

```
.. c:function:: int ed(struct ieee802154_hw *hw, u8 *level)
```

Handler that 802.15.4 module calls for Energy Detection from the hardware device.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\networking\[linux-master][Documentation][networking]ieee802154.rst, line 178)

Unknown directive type "c:function".

```
.. c:function:: int set_channel(struct ieee802154_hw *hw, u8 page, u8 channel)
```

Set radio for listening on specific channel of the hardware device.

Moreover IEEE 802.15.4 device operations structure should be filled.