



Chopping cables UWB, WiNET and Wireless USB in Linux

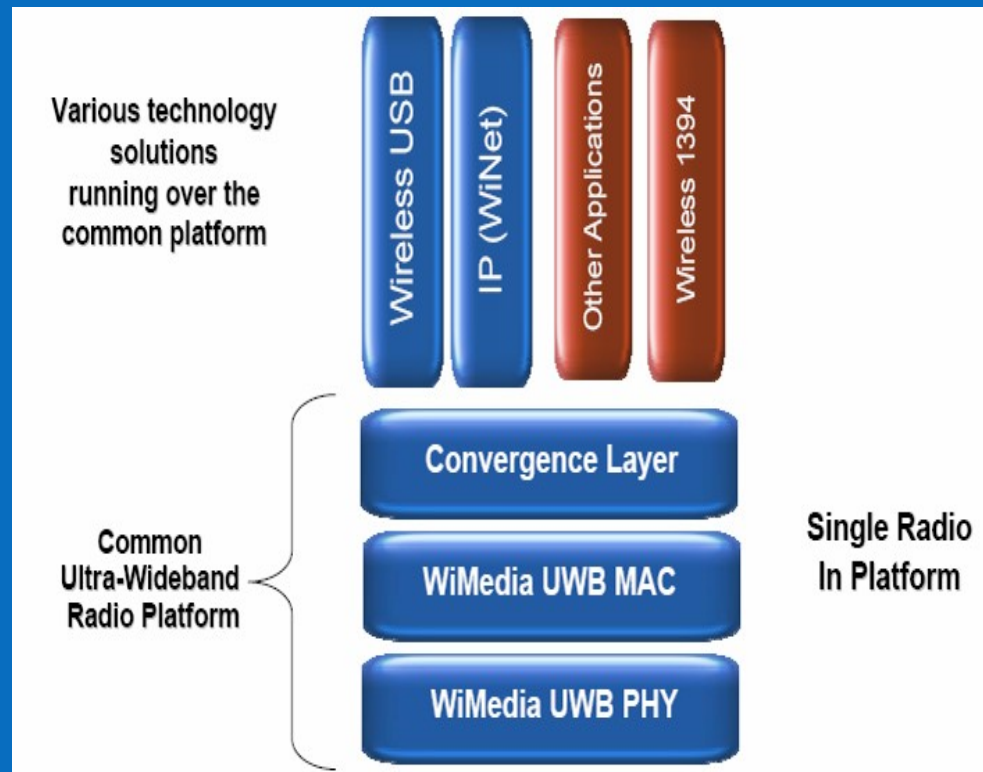
Iñaky Pérez-González <inaky.perez-gonzalez@intel.com>

<http://linuxuwb.org>



What is all this?

- **UWB is a new PAN radio protocol to free your desk and home from data cables**
- **Designed for**
 - low power (~5 orders of magnitude less than my WiFi*)
 - efficient streaming
 - QoS
- **High speed (53-480 Mbps), short range (3m/480Mbps to 10m/100Mbps)**
- **AES-128 encryption**

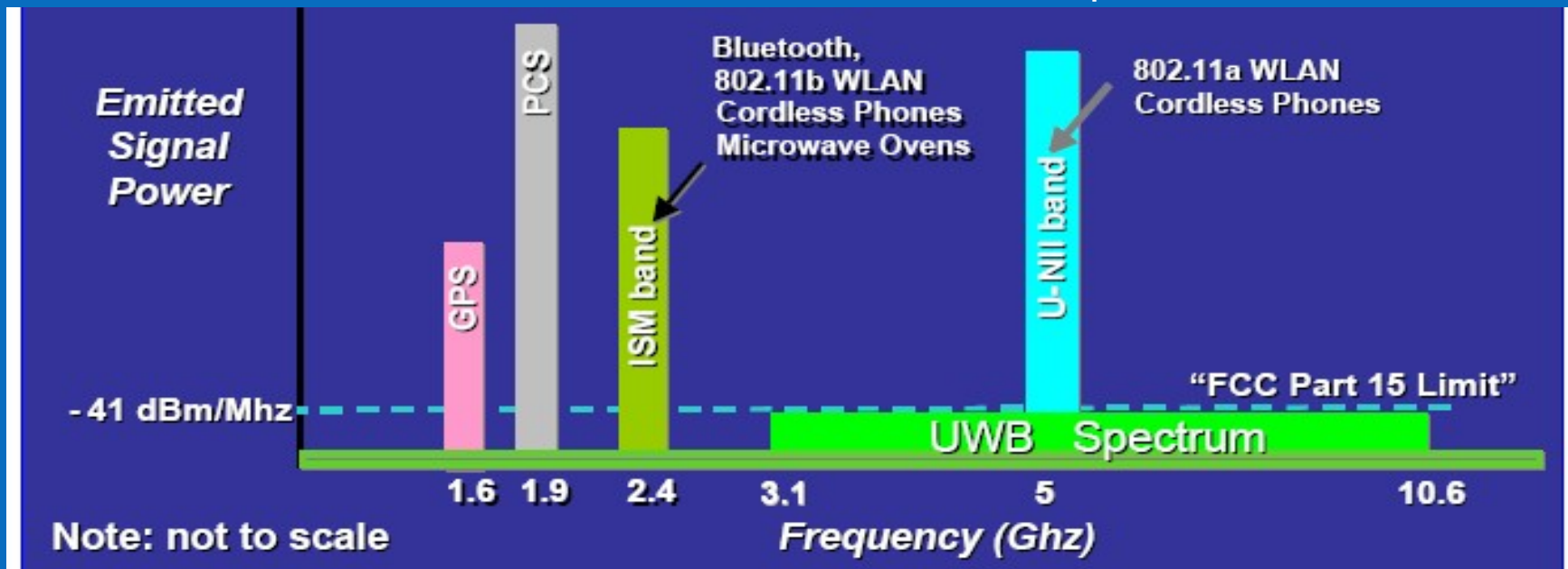


- **Higher level protocols piggyback on top:**
 - Wireless USB
 - WiNET (IP over UWB)
- **As of now, all based in open standards**

(*) very roughly calculated

UWB: gory radio details

- **OFDM in the unlicensed band from 3.1 GHz to 10.6 GHz**
- **Split in *band groups***, each composed of three 528 Mhz subbands (thus a 1584 MHz channel or band group)
- **Low power emission: max is -41 dBm/MHz (0.074 μ W/MHz)**
 - Practical emission per band: 100 μ W (-10 dBm)
 - About three thousand times less than a cell phone



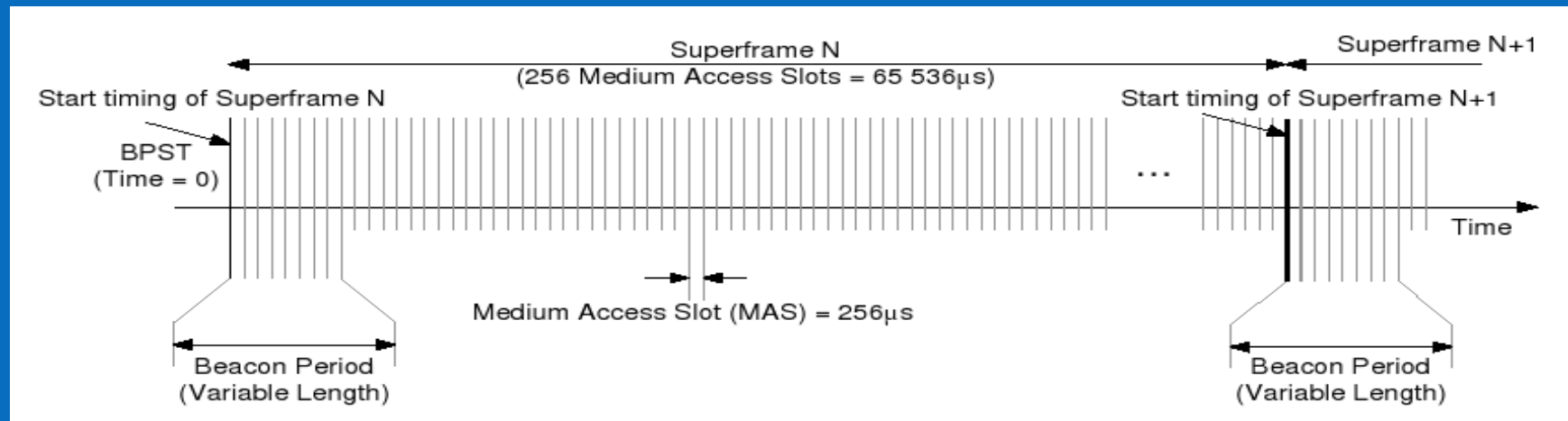
UWB: on the air

- **Radio time divided in Super Frames (SF)**

~65 ms

- One SF is divided in 256 MAS

~256 μ s



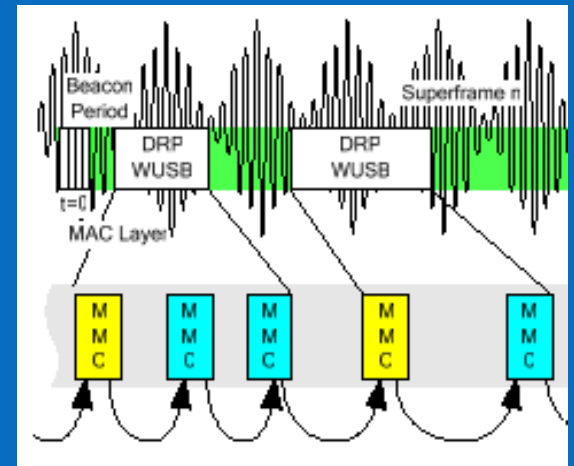
Credit: ECMA-368, Fig 3

- **Medium access rules: devices may transmit...**

- their beacons (during the beacon period) at base rate (53Mbps)
- during MAS reserved to them (Dynamic Reservation Protocol/TDMA)
 - devices negotiate for access to bandwidth
- when the medium is not used: Prioritized Contention Access
 - CSMA methodology: carrier sense and transmit

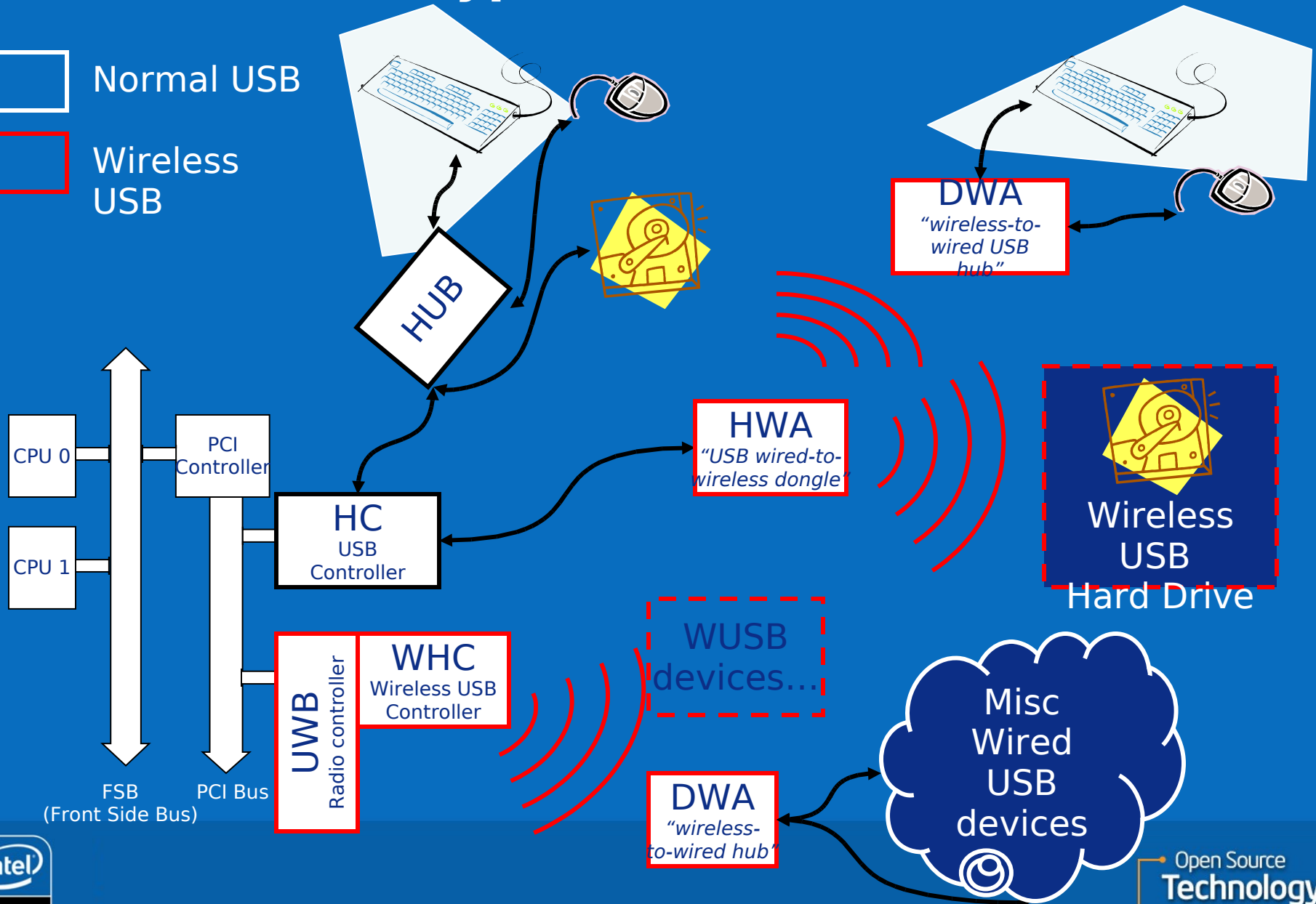
Wireless USB

- **Take USB 2.0, replace cables with UWB, add some glue.**
- **Same USB features (reuse all drivers) plus:**
 - Wireless (duh!) Encryption Dual role support
- **Master/slave model created on top of UWB**
 - Host Controller creates a DRP/TDMA reservation and establishes a WUSB cluster with a linked list of MMC
 - Devices just follow MMCs, no need to know UWB details—simpler than UWB
 - MMCs schedule time for signalling, tx and rx (each allocation is called W_x CTA).
 - Cable concepts (connect, disconnect, reset, etc) are translated to signalling:
 - DNTS, device notifications (device to host)
 - WUSB Information Elements (host to device)



Wireless USB: types of controllers

- Normal USB
- Wireless USB

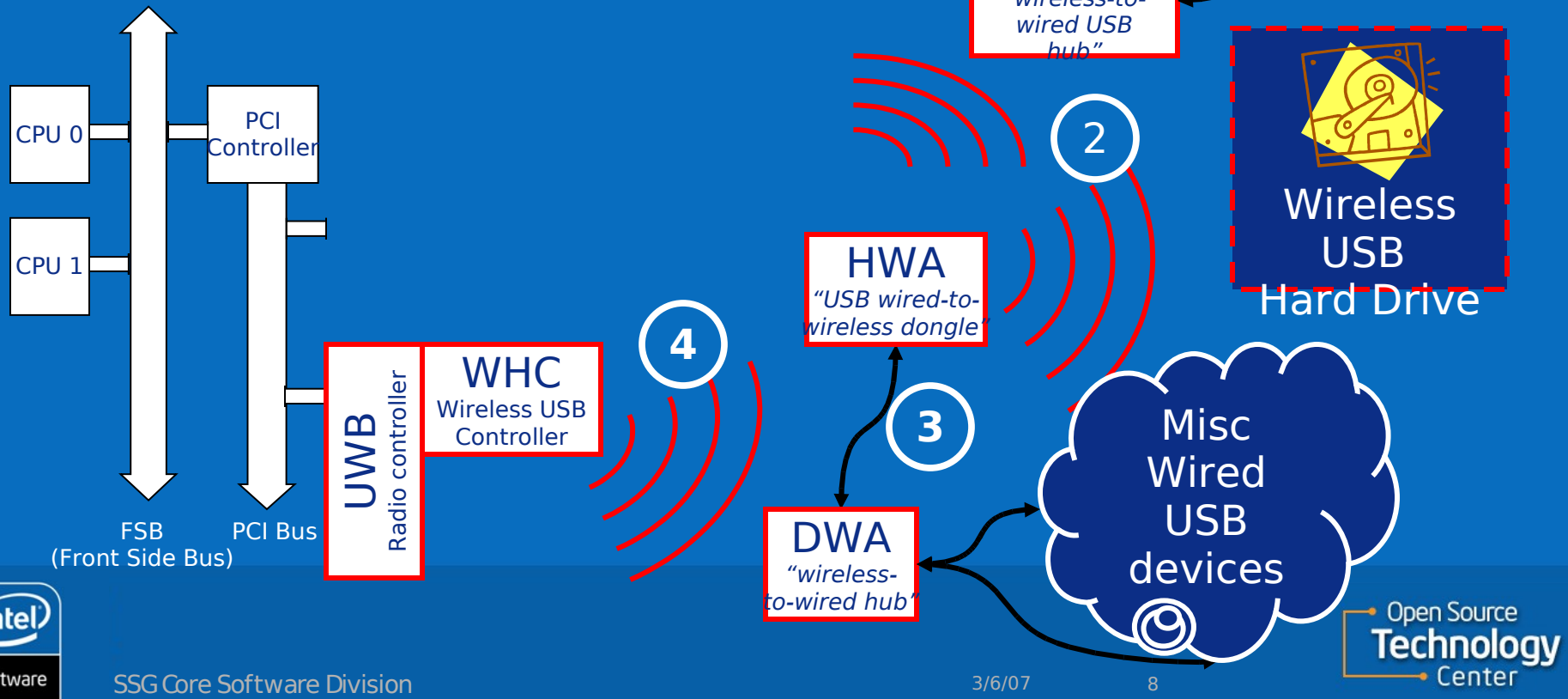


Wireless USB

- **HWA: Host Wired Adapter – the WUSB USB-dongle**
 - WUSB Host Controller connected to the system via USB!
 - Encapsulates WUSB traffic over USB
 - Shows up as a USB host to the system
 - Lots of overhead, but easy to connect **legacy support**
 - Includes a UWB Radio Control interface (WUSB 1.0 mandated)
- **DWA: Device Wired Adapter – a WUSB hub!**
 - Connect all your wired devices wirelessly to your machine
 - Similar to HWA, encapsulates USB traffic over WUSB
 - Also shows up as a USB host
 - For all those devices you have laying around at your desk
- **WHCI: the PCI connected host controller**
 - For new systems, possibly embedded in the chipset
 - Most effective and lowest overhead
 - Also includes a UWB Radio Control interface (WHCI 0.95)

Wireless USB: we can make it a mess

1. USB 2.0 traffic
2. Pipes USB2.0 on WUSB
3. Pipes WUSB (that pipes USB2.0) on USB
4. Pipes USB (that pipes WUSB (that pipes USB)) in WUSB...



Do we need security in WUSB? Why?



Yes, because of human ingenuity (*)



(*) Pic and tagline stolen from John Keys

Wireless “*should be safe enough*” USB

- **Based on a 4-way handshake, one time pads and Connection Contexts**
 - CHID: host ID, doesn't change
 - CHID: device ID, assigned by the host, kept by the device
 - CK: Connection Key, assigned by the host
- **On connection**, if a host knows the device (it has a CC that matches CDID) and the device can prove it has the secret (CC), crypto keys are derived from the CC and secure traffic starts.
- **When the host doesn't know the device**
 - **CBA:** Cable Based Association
 - Out of band – for wired+unwired devices
 - Use the cable to install a CC in the device, then use unwired
 - No man-in-the-middle – the cable is considered secure
 - **Numeric:**
 - User confirms with a 2-4 digits display in the device and host
 - Man-in-the-middle mitigated by digits, distance, explicitness

WiNET: IP over UWB

- **Motto: slap an Ethernet frame inside a UWB frame**
- **Allows to do TCP/IP, bridging, etc, no big news here...**
- **...so it is similar to Wifi, but not the same:**
 - For once, the range is only 10m
 - A brick wall will stop it short...can't share between rooms
 - There are no APs, all is ad-hoc [yaaah!]
 - But it's way faster (and efficient, power wise)
- **Devices group in WSSs (WiNET Service Sets)**
That's a network identified by an ESSID in wifispeak
Devices can associate with one or more at the same time
- **Very useful for streaming**
DVD player streams video to the flat screen TV and audio to the speakers
Beam music from your cellphone to your Hi-Fi speakers

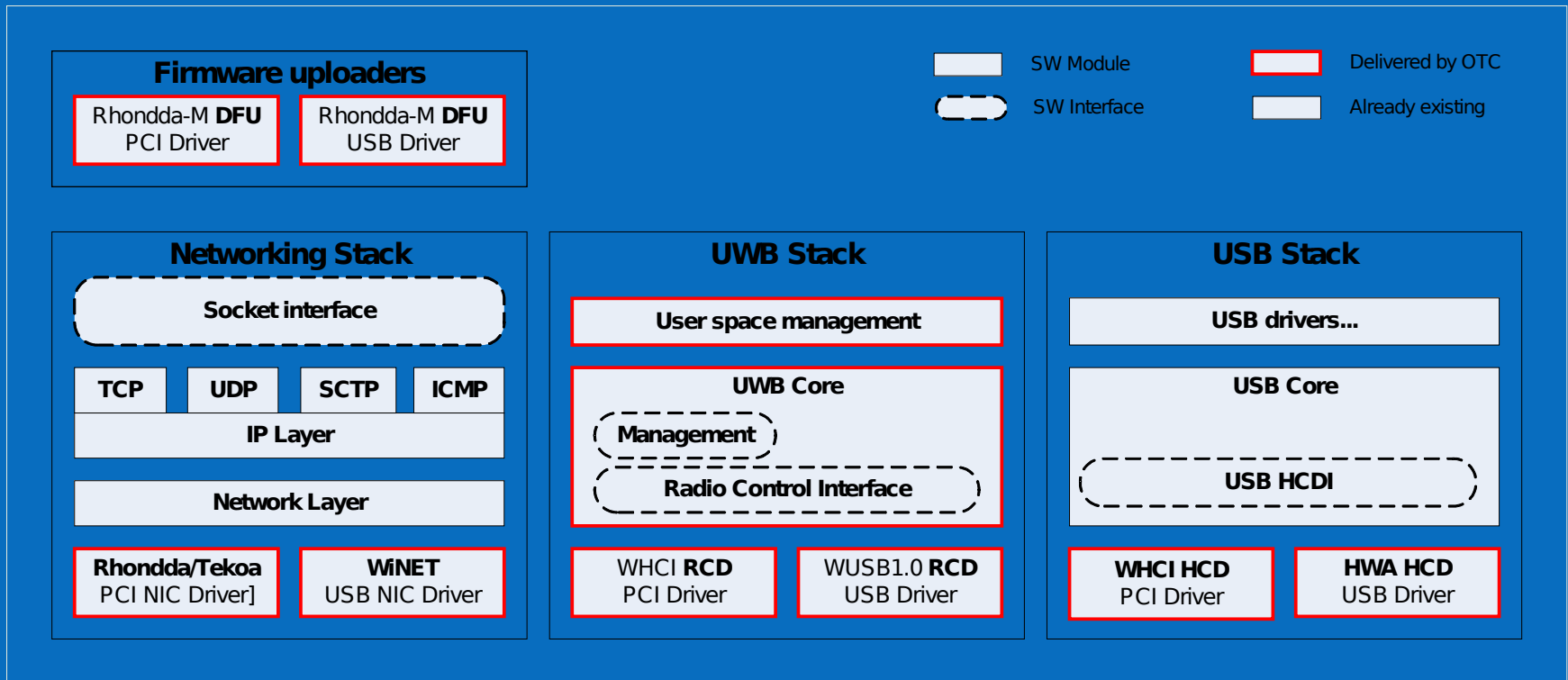
And course, you can use it for net connectivity :)



Example usage models for all this

- **DVD player streaming video to Flat Screen TV and audio to speakers** [no more ratnests in your living room]
- **Cellphone streaming** [to TV, beaming music, video or pics to other phones or computers]
- **Play content to your car entertainment system**
- **Wireless dockstations!** [well, but for the power cable]
- **Coming soon: UWB connection replacing monitor cable**
- **HiFi 2-way headsets**
- **Wireless storage**
- **Wireless Internet access in high-density urban quarters where WiFi is just not feasible.**

Linux drivers stack structure



- **No special tools are required to manipulate the stack** (simple shell commands on sysfs suffice to control it)
- **User space has control over all UWB operation parameters**

Demo of WUSB

- Got none!! [couldn't bring any devices]
- (but I can demo WiNET)...

Using WiNET

Find UWB Radio Control adapters on host A

```
A$ cd /sys/class/uwb_rc && ls  
uwb2/
```

Find your UWB device and MAC address

```
A$ cat uwb2/dev_address uwb2/mac_address  
76:8d  
2b:11:11:86:80:02
```

Look for devices (scan) on channel 13

```
A$ echo 13 0 | cat > uwb2/scan
```

Any devices?

```
A$ ls /sys/class/uwb  
76:8d # Nope! This is us
```



Using it (2) – joining a beacon group

No devices? stop scanning and start beaconing on ch13

```
A$ echo 13 3 | cat > uwb2/scan
A$ echo 13 0 | cat > uwb2/beacon
```

Look for devices (scan) on host B

```
B$ echo 13 0 | cat > uwb0/scan
```

Any devices seen from B?

```
B$ ls /sys/class/uwb
76:8d          86:8b          # Yes, we see A (B is
86:8b)
```

Then tell B to beacon against A, so they join

```
B$ echo 76:8d > uwb0/beacon
```

A and B are now part of the same beacon group



Using it (3) – bringing WiNET up

- **Lazy, let's use ifconfig**

```
A$ ifconfig winet0 192.168.2.1  
B$ ifconfig winet0 192.168.2.2
```

- **Ping**

```
A$ ping 192.168.2.2      # B  
etc, etc
```

- **Now use it as any other TCP/IP connection**

- **Very basic, but it is simply prepared for you to build on top**

Summary

- UWB is a high performance, low power, PAN radio protocol
 - designed for low power consumption and media streaming
 - frees you from data cables
- Wireless USB: USB protocol on top of UWB
 - provides backward compatibility at the software level and legacy platform path (HWA)
 - Encryption should make it as safe as wired
 - Cable *physical signalling* replaced by radio
- WiNET: Ethernet frames encapsulated over UWB
 - Simple protocol, similar in appearance to WiFi, but no APs
 - very useful for piconets, media streaming, high-density urban connectivity
- Linux ready -- <http://linuxuwb.org>
 - To a certain extent! (of course :)

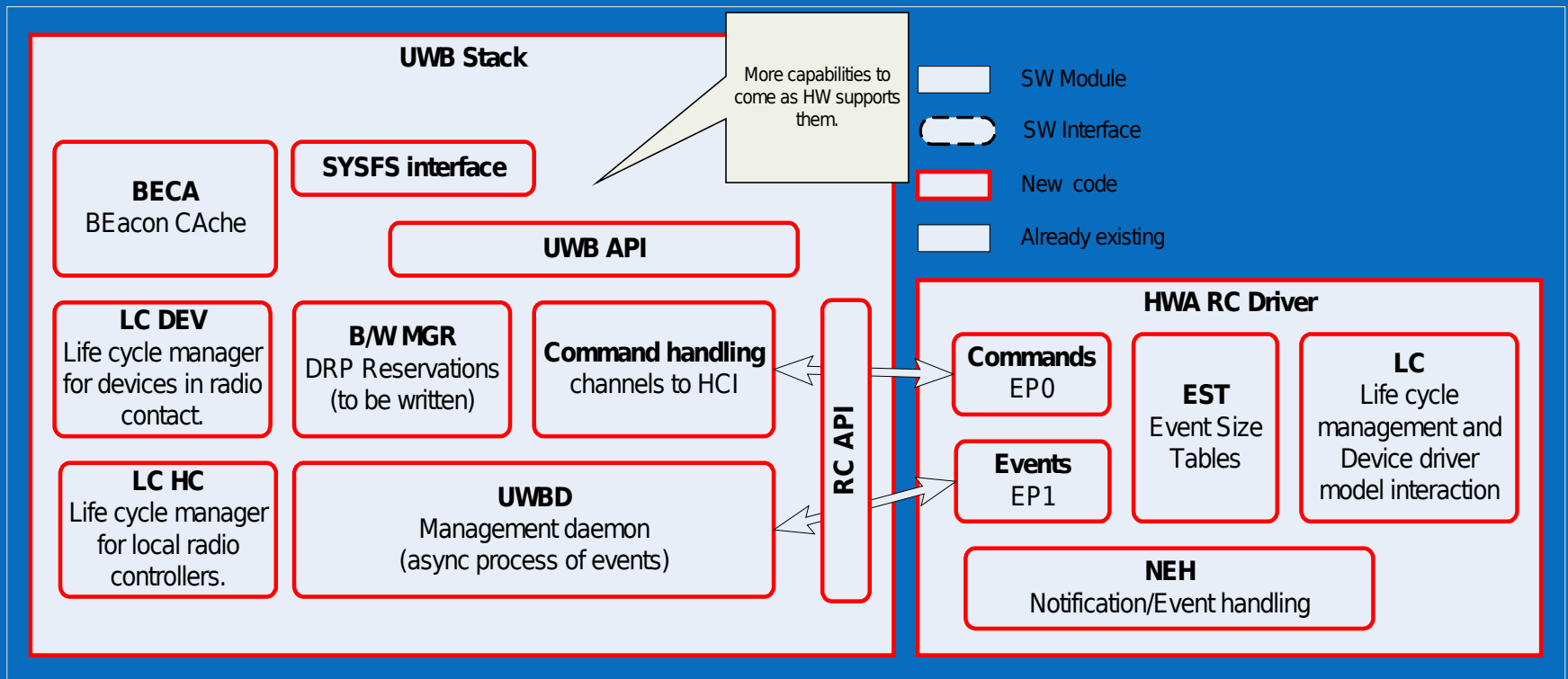
Questions?



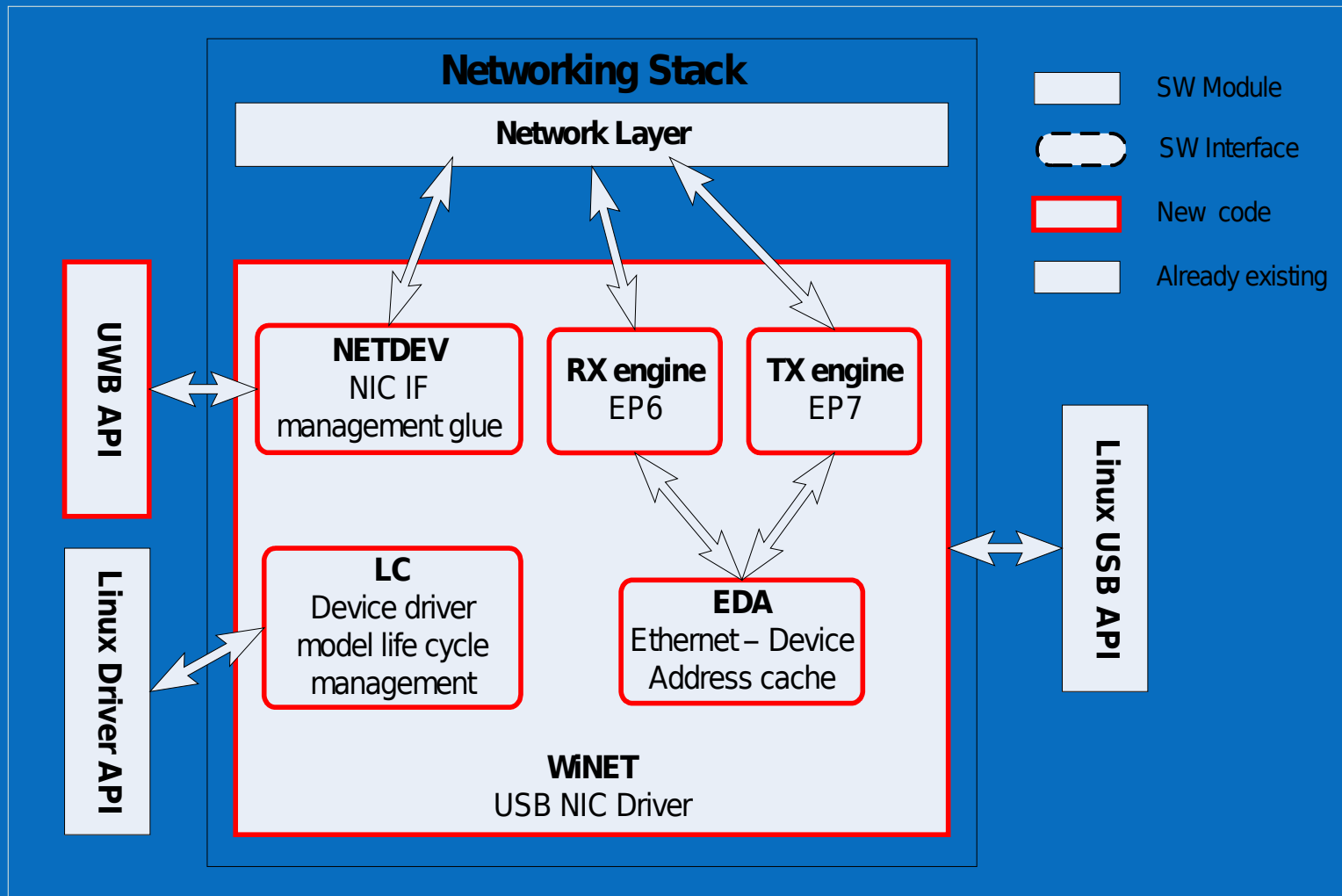
Backup



The UWB stack and HWA RC driver



The WiNET driver



The HWA WUSB Host Controller driver

