HUAWEI

IoT Networking Workshop

2021-07-13, NetdevConf 0x15, virtual

Agenda

19:15 – 19:25	Welcome, session format (Stefan Schmidt)
19:25 – 09:45	SocketCAN and J1939 subsystem session (Oleksij Rempel & Marc Kleine-Budde)
19:45 – 20:05	hwsim tutorial and RIOT-OS interfacing (Alexander Aring)
20:05 – 10:25	IoT Gateway Blueprint discussion with (Stefan Schmidt)
20:25 – 20:45	Misc topics (Stefan Schmidt) - updated IoT devices (by PC) - More IPv6 IoT protocols (Matter, etc) - ieee802154 mlme work - Native ieee802154 transport for OpenThread

Times in CEST timezone

Session Format

- Chair: Stefan Schmidt
- More interactions, less classroom style
- Small set of slides for an status update or problem statement
- Time for discussion

SocketCAN and J1939

Oleksij Rempel – ore@pengutronix.de
Marc Kleine-Budde – mkl@pengutronix.de

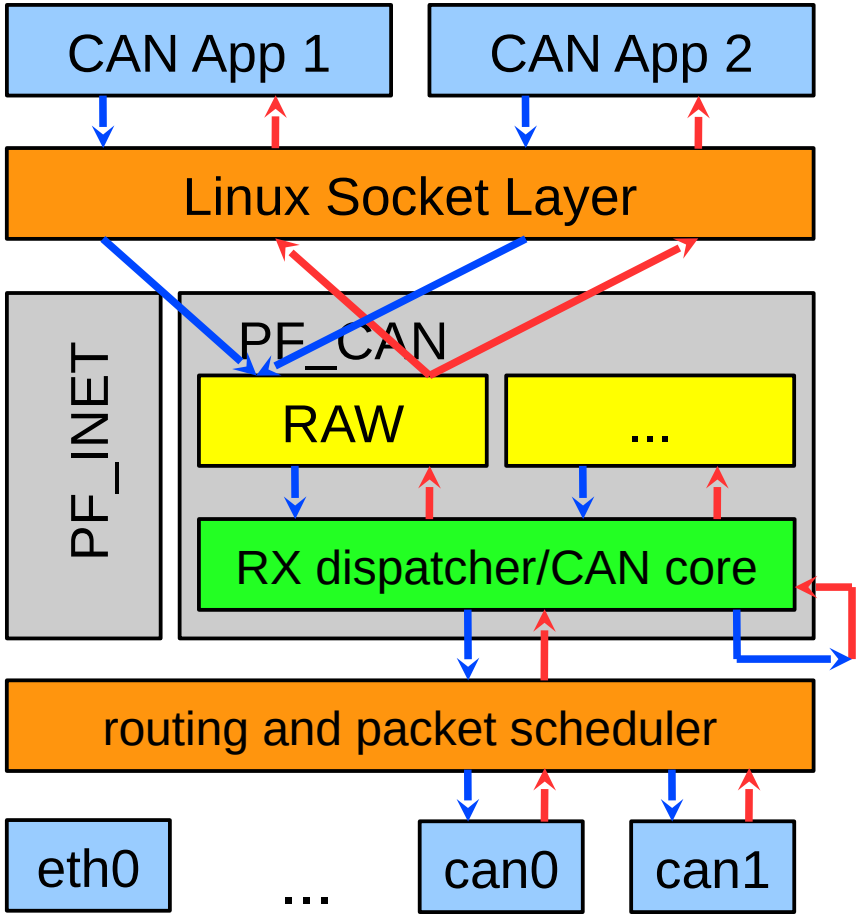
... spend some words on CAN

- CAN != Ethernet
- 2 wire cable
- broadcast medium
  - CSMCA (Carrier Sense Multiple Collision Avoidance)
- multi master bus
  - 100base-T1 is single master bus, 10base-T1s multi drop bus
- Speed:
  - up to up to 1 Mbit/s (CAN-2.0)
  - 8...12 Mbit/s (CAN-FD, CAN-XL)

... spend some words on CAN

- 11 or 29 bit address (CAN-ID)
- prioritization of CAN frames by CAN-ID
- Payload:
  - 8 Bytes (classic CAN-2.0)
  - 64 Bytes (CAN-FD), CAN-FD is compatible with existing CAN 2.0 networks.
  - 2048 Bytes (CAN-XL, ~~SiG is "near completion",~~ CAN XL plugfest at June 16, 2021)
    - https://www.bosch-semiconductors.com/news/t-newsdetailpage-4.html
    - https://can-newsletter.org/engineering/engineering-miscellaneous/200103_2020s-decade-welcome-can%20xl_cia

SocketCAN isn‘t Rocket Science!



- HW abstraction for different controllers
- Currently supported socket types: CAN_RAW, CAN_BCM, CAN_J1939
- CAN-2.0 and CAN-FD - seamless integration, configurable on socket level

What is SAE J1939?

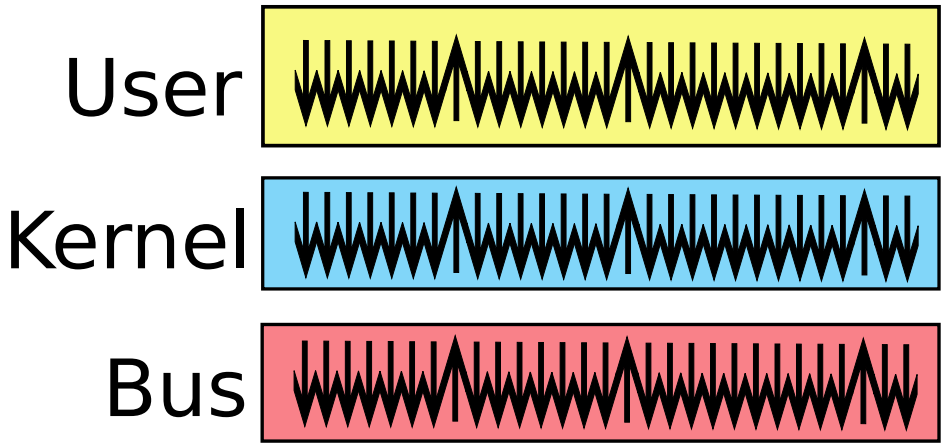
- Recommended practices and standards:
  - SAE J1939: for truck and bus applications, possible other..
    - ISO 11783: for tractors and machinery for agriculture and forestry
    - NMAE 2000: marine sensors and display units within ships and boats
    - MilCAN: military vehicles
- Reliable transfer for large amounts of data
- SAE J1939: Transport Protocol = 1785 Bytes

  (255 packets * 7 Bytes/packet)
- ISO 11783: Extended Transport Protocol ~ 112 MiB

  (16.777.215 packets * 7 Bytes/packet)

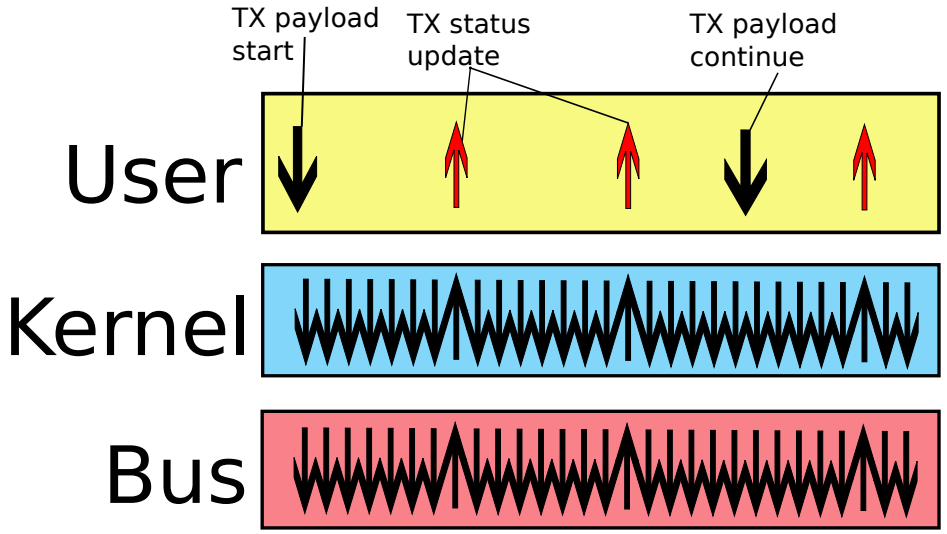
What if (E)TP is implemented in user space

- CAN_RAW is used
- 8Byte kernel<>user per message
- User space stack needs high prio to meet strict timing requirements in the protocol



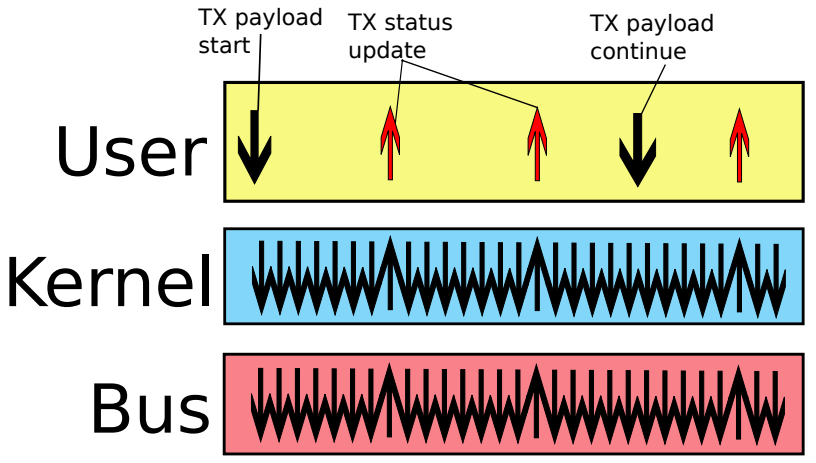
J1939: What current v5.13 can do? (TX path)

- challenges:
  - Length of the (E)TP transfer is send in first packet
  - Preserve transfer boundaries
  - Limited socket buffer size (usually < 112MiB)
- Implementation:
  - First send() with full length (112MIB if needed)
  - If MSG_DONTWAIT, loop send() with remaining data



J1939: TX status update

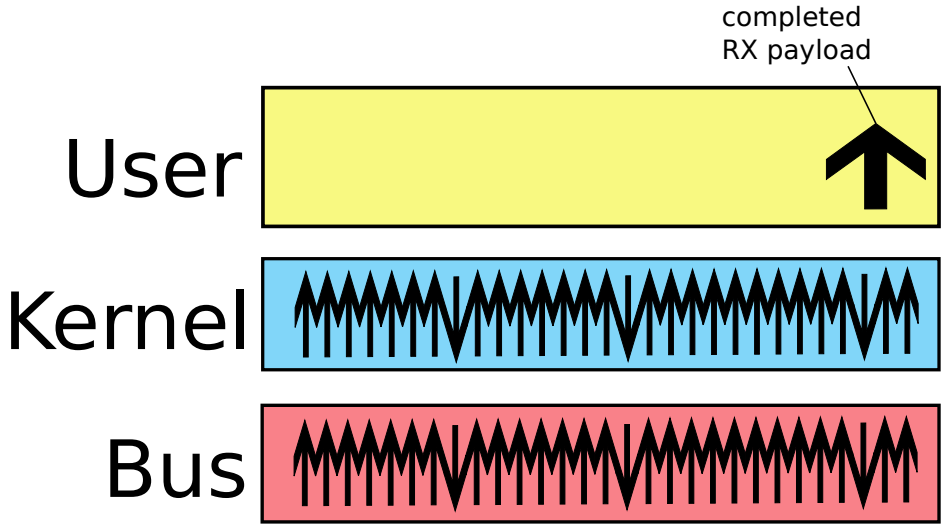
```c
static void j1939_tx_timestamp(void)
{
        int sock = socket(PF_CAN, SOCK_DGRAM, CAN_J1939);
        int value = 1;
        ret = setsockopt(sock, SOL_CAN_J1939, SO_J1939_ERRQUEUE, &value,
                         sizeof(value));

        sock_opt = SOF_TIMESTAMPING_SOFTWARE | SOF_TIMESTAMPING_OPT_CMSG |
                   SOF_TIMESTAMPING_OPT_STATS | SOF_TIMESTAMPING_OPT_TSONLY |
                   SOF_TIMESTAMPING_OPT_ID |
                   SOF_TIMESTAMPING_TX_ACK |
                   SOF_TIMESTAMPING_TX_SCHED;
        setsockopt(sock, SOL_SOCKET, SO_TIMESTAMPING,
                   sock_opt, sizeof(sock_opt));

        bind(sock, ...);
        send(sock, , MSG_DONTWAIT);
        struct pollfd fds = {.fd = sock, .events = POLLERR, };
        poll(&fds,...);
        if (fds.revents & POLLERR) {
                recvmsg(sock, &msg, MSG_ERRQUEUE);
                .....
        }
}
```



https://github.com/linux-can/can-utils/blob/master/j1939cat.c

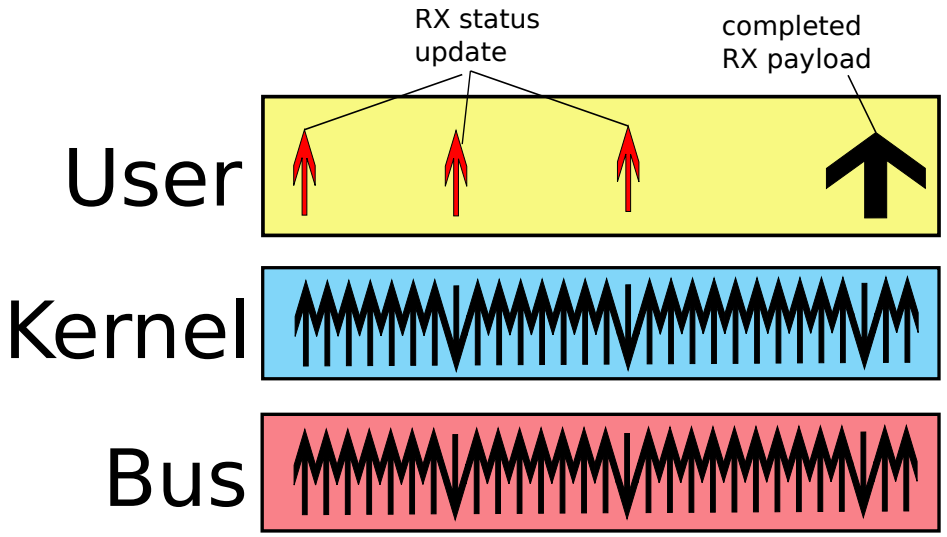
J1939: What current v5.13 can do? (RX path)

- Payload on the end of transfer
- No status notifications
- No way to tell user about started transfer
- No way to tell user about aborted transfer



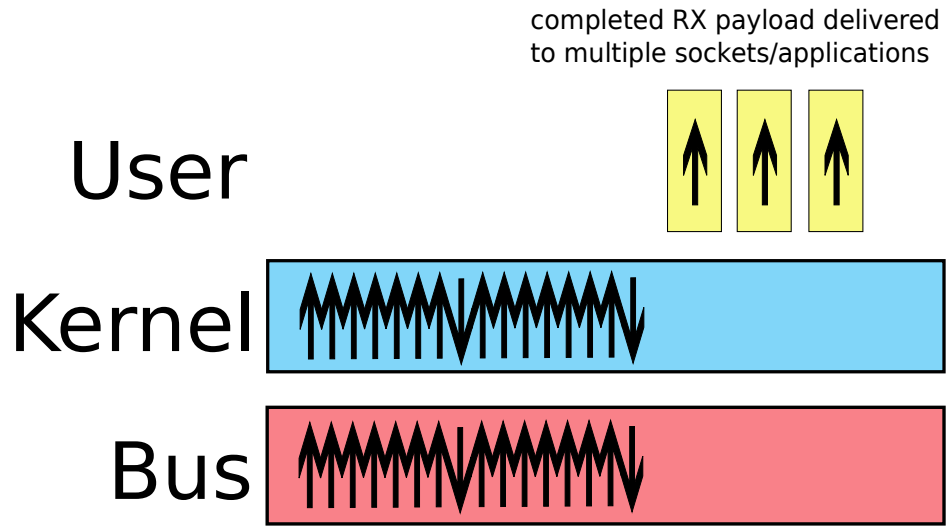
J1939: What new v5.xx will do? (RX path)

- Optional RX status updates. Compatible with old applications
- RX status updates for:
  - RTS – request to send
  - DPO – data packet offset
  - Abort
- On completion, payload instead of status update
- Notifications are delivered to sockets



J1939: One TX, multiple RX sockets

- On CAN bus is everything broadcast
- Traffic sniffing is common practice
- For example update of multiple MCUs with one transfer. Reducing bus load



CAN_J1939 socket

- Available since kernel 5.4 (Nov 2019)
- 25 bug reports provided by google syzkaller, 20 are fixed (09.07.2021)

TODO

- J1939
  - Proper way to export address claiming cache to the user space
  - Quirky buses
  - Test automation (follow osmocom testing experience?)
  - Incremental RX support
- Time Sensitive Networking (TSN)
  - More drivers supporting HW timestamps
  - TT-CAN (Time Triggered-CAN), for example in m-can (IP core on stm32-mp1)
- PHYs
  - following functionalities are currently expected on CAN PHYs:
  - switch PHY on/off
  - listen only support
  - wake-up support
  - cable testing
- HW filter support
- XDP support

Thank you!

Questions?

hwsim tutorial and RIOT-OS interfacing

Alexander Aring

hwsim driver

- Module “mac802154_hwsim.ko”
- Replaced “~~fakelb.ko~~”
- Idea “mac80211_hwsim.ko”
- Provides PHYs and PHY layer
- Defining a Mesh-Topology

Other solutions around...

- MiniNet[1] – high level hwsim
- RIOT-OS ZEP[2] – UDP POSIX
- openthread[3] – UDP POSIX

hwsim provides a PHY layer standard

?with more functionality?

1) http://mininet.org/
2) https://github.com/RIOT-OS/RIOT/pull/6121
3) https://openthread.io/codelabs/openthread-simulation-posix

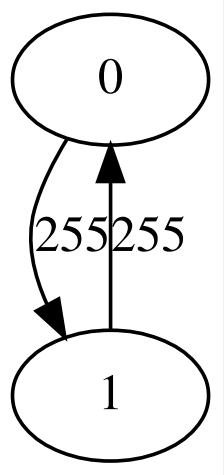
RIOT-OS

- The friendly Operating System for the Internet of Things[1)]
- ELF-Binary Sandbox “native”[2)]
- Provides 802.15.4/6LoWPAN

1) https://www.riot-os.org/
2) https://github.com/RIOT-OS/RIOT/wiki/Family%3A-native

PHY: Build a topology 1

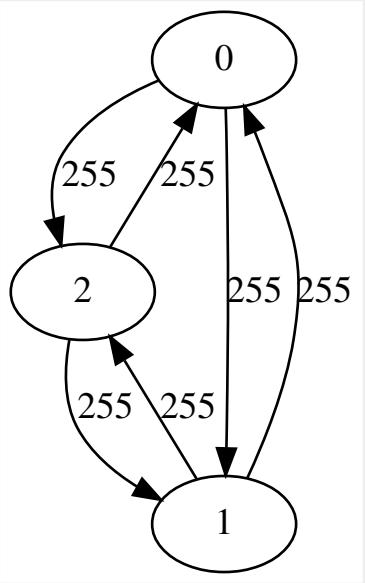
- “wpan-hwsim” - wpan-tools[1)]
- Default: Two PHYs registered
- `wpan-hwsim -d`
- Exampe: phy0, phy1
- Note: assymmetric possible



1) https://github.com/linux-wpan/wpan-tools

PHY: Build a topology 2

- `wpan-hwsim add` - new phy2
- `wpan-hswim edge add 0 2`
- `wpan-hswim edge add 2 0`
- `wpan-hswim edge add 1 2`
- `wpan-hswim edge add 2 1`



Red Hat

MAC: How to connect them?

Interfaces (MAC) on PHYs!

- Linux <-> Linux
- Linux <-> $USER_STACK
- $USER_STACK <-> $USER_STACK
- Monitor everything (Sniffer)

MAC: Linux

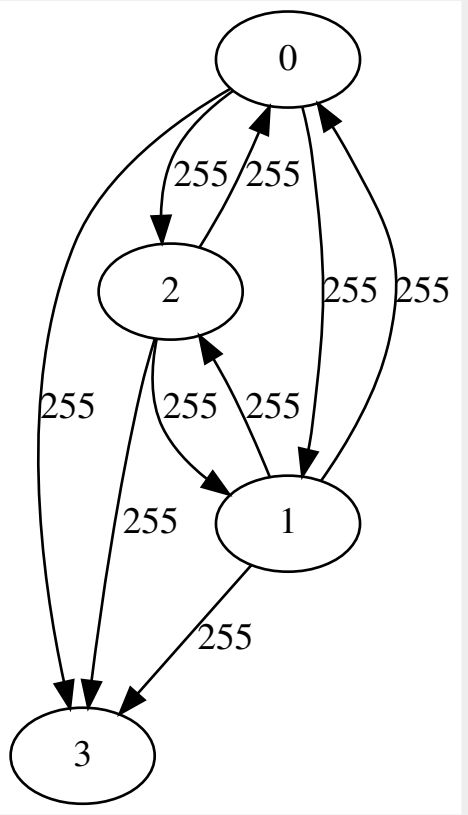
- Node type: node
- Somewhat required:
  - Put the phy into it’s own netns!
  - `ip netns add foo`
  - `iwpan phy2 set netns name foo`
  - Separate resources! E.g. routing

MAC: $USER_STACK

- e.g. RIOT-OS: native platform
- Node type: monitor!
- Required: Virtual Driver in APP
  - Send/Recv RAW frames
  - ~~af_802154~~ – AF_PACKET RAW!
- Maybe no namespace required

MAC: Sniffer

- Node type: monitor
- PHY3 receives everything!
- Run e.g. wireshark on monitor interface then
- `wpan -d -i 3` - ignore 3



Red Hat

MAC: Limitations

- No ARET/ACK handling!
- No Filtering - Address/CRC
- $USER_STACK okay for development
- $USER_STACK is an OS on top of OS - Sandbox

Future Work

- $USER_STACK on node type
- Upstream RIOT Virtual Driver
- Upstream openthread?
- Graphical User Interface
- LQI, Simulate 802.15.4 -> TC?

End

Thanks!

IoT Gateway Blueprint

- Brainstorming on ideas of a modern IoT smart home hub
- Ignoring branding and vendor lock-in topics
- Could the zoo of smart IoT hubs be reduced to a single one?
- My hope is that this discussion could drive direction and dicovery of pain points for further IoT networking

Seed Topics to Start the Discussion

- Aim for end-to-end solution without tranlating proxies on the hub e.g. IPv6 from device to device or cloud
- Native IPv6 solutions (OpenThread, Matter (former CHIP))
- Native IPv6 in SOHO e.g. prefix delegation from ISP router
- NAT64 for IPv4-only transit networks (tayga, jool, ebpf?)
- Sandboxes for vendor specific code (e.g. backend connection, updates, etc) to avoid an extra hub for just software

Updated IoT Devices

- Encouraged topic from program committee
- Mandatory update functionality by law is coming in some countries
- ASOS approach: direct OTA on powerful devices and assisted-OTA on devices that could not spare the resources
- Any network related topics, beside transport, for updates?

More IPv6 first/only Protocols

- 6lowpan paved the way for IPv6 on low-power standards e.g. ieee802154 in OpenThread
- IPv6 as default on WIP protocols like Matter as well
- We start to see IPv6-only subnets like we have seen IPv6-only data-centers before
- Some transition techniques could be re-used

Ieee802154 mlme

- Advanced functionality like scanning, joining, network coordinator, etc needs support for command and beacon frame processing
- Over the years various parties started work but it never found it way into mainline
- To make linux-wpan an option for real-world deployments this will be needed

Native ieee802154 Transport

- OpenThread runs natively on Linux for Border Router scenarios
- This still uses a radio co-processor or network co-processor architecture where Linux networking is only involved on top
- PHY, MAC, 6lowpan are all firmware or userspace driven
- Alex mentioned raw frames with AF_PACKET which would, again, cirumvent parts of the kernel stack
- More functionality needed if we want a linux-wpan OpenThread platform abstraction