The Road Towards a Linux TSN Infrastructure

Jesus Sanchez-Palencia
About me

• Software Engineer at Intel (~5 years)
  ○ Open Source Technology Center (OTC)

• Currently: drivers and kernel interfaces for TSN
  ○ Linux Network Stack

• Background
  ○ Intel Quark Microcontrollers SW stack (QMSI)
  ○ Embedded OSes: Zephyr and Contiki, Android, Maemo
  ○ Web Rendering Engines (WebKit, Crosswalk)
  ○ Qt Framework
Objectives

• Provide a (very) brief introduction to Time-Sensitive Networking
• Present the current upstream TSN SW architecture
• Discuss the challenges ahead
Objectives

• Provide a (very) brief introduction to Time-Sensitive Networking
• Present the current upstream TSN SW architecture
• Discuss the challenges ahead
LANs and the Internet

• Common model based on Internet Protocols and the IEEE 802 architecture.

• Mode of operation is **best-effort**
  - as in *quickest*
  - Metrics are all based on *average* (i.e. delay, speed)

• Not suitable for use cases that require high / known *availability*
  - like circuit switching networks
  - or Fieldbuses for control networks
    - operational network (OT) != information network (IT)
    - e.g.: CAN*, EtherCAT*, Profibus*, Profinet*, ...
      • lack of interoperability
What is Time-Sensitive Networking?

- Set of evolving standards developed by IEEE to allow for time-sensitive traffic on Ethernet based LANs.
  - started from Audio/Video Bridging (AVB)
  - allows for OT and IT traffic to co-exist
- Provides bounded worst-case latency
  - as in deterministic
  - determinism is prioritized over throughput
- Standards are mostly developed as extensions to 802.1Q
  - Virtual LANs (vlans) and QoS
- AVNU Alliance*
  - Interoperability
- Targets different segments
  - e.g.: Pro A/V, Industrial Control, Automotive systems
TSN: Example

- **Infotainment**
  - multiple screens
  - multiple speakers
    - video + audio synchronized
  - noise reduction?
    - multiple mics
- **Control**
  - multiple sensors and actuators
- **Why TSN?**
  - Ethernet is cheap.
  - Cabling is one of the most expensive components in a car.
- **Same network?**
  - Theoretically, yes.
TSN: Theory of Operation

Physicist

• Mechanisms:
  o Time Sync
    ■ 802.1AS
  o Traffic identification
    ■ VLAN tags
  o Resource allocation
  o Traffic shaping / scheduling

• Network Config:
  o 802.1Qcc
  o Dynamic or static
    ■ e.g.: SRP
  o Distributed or centralized
TSN: Traffic Shapers

• TSN applications have different requirements
  o Reserved Bandwidth
  o Strict cycles: scheduled Tx
• 802.1Qav: Credit-based shaper (CBS)
  o per-queue bounded bandwidth
  o “transmit all packets from this traffic class at X kbps”
• Time-based Scheduling (TBS)
  o per-packet Tx time
  o “transmit this packet at timestamp 152034537600000000 ns”
  o not earlier than or not later than?
• 802.1Qbv: Enhancements to Scheduled Traffic
  o per-port queues schedule
  o “execute the Tx algorithm on queue 0 every 100us for 20us, on 1 every 240us for 30us”
• 802.1Qbu, 802.1Qci, ...
“Talker” application
a. Enable Multiqueue
b. Configure Queues (shapers)
c. Classify traffic
   - steer to Tx queue
   - allow network to identify it
d. Transmit

“Listener” application
a. Optionally: setup Rx filters
   - i.e. VLAN priority, src and dst MAC
b. Receive
Objectives

• Provide a (very) brief introduction to Time-Sensitive Networking
• Present the current upstream TSN SW architecture
• Discuss the challenges ahead
TSN SW - Previous Attempts

- **OpenAVB - Eric Mann’s (Intel)**
  - bypasses kernel network stack
  - forked driver: igb_avb
  - config and data paths: libigb
  - Tx Queues exposed directly to the userspace

- **RFCs on netdev from Henrik Austad (CISCO*)**
  - media centric (AVB)
  - bundled up as a TSN driver
    - ConfigFS based interface
  - ALSA shim for audio streaming

- **Driver-specific interfaces on upstream**
  - stmmac* and (maybe) others: devicetree as a config interface for shapers

- **Downsides: kernel bypassing, hw-dependent, monolithic solutions**
Traffic Control on Linux

• Provides
  o Shaping / Scheduling (Tx)
  o Policing (Rx)
  o Dropping

• Queueing Disciplines, Classes and Filters

• Qdiscs
  o Kernel Packet buffer
    ■ Sits ‘between’ protocol families and netdevice driver
  o Control when / how packets are transmitted
  o Every interface has a default root qdisc attached
    ■ Qdiscs can expose classes
  o Qdiscs can “offload” work to hardware

$ tc -g qdisc show dev wlp58s0
qdisc mq 0: root
qdisc fq_codel 0: parent :4  (⋯)
qdisc fq_codel 0: parent :3  (⋯)
qdisc fq_codel 0: parent :2  (⋯)
qdisc fq_codel 0: parent :1  (⋯)

$ tc -g class show dev wlp58s0
+---(:4) mq
+---(:3) mq
+---(:2) mq
+---(:1) mq
Config interface: Multiqueue

- **mqprio qdisc: Multiqueue priority**
  - It "exposes" HW queues as classes, allowing for other inner qdiscs to be attached.
  - Maps priorities to traffic classes to HW queues.

- **Example: 3 traffic classes**
  - prio 3 -> tc 0 -> queue 0 (8001:1)
  - prio 2 -> tc 1 -> queue 1 (8001:2)
  - other -> tc 2 -> queues 2 (8001:3) and 3 (8001:4)

```
$ tc qdisc replace dev enp2s0 \
parent root mqprio num_tc 3 \nmap 2 2 1 0 2 2 2 2 (...) \nqueues 1@0 1@1 2@2 hw 0
```

```
$ tc -g class show dev enp2s0
+--(8001:ffe2) mqprio
 | -++-(8001:3) mqprio
 | -++-(8001:4) mqprio
 | +---(8001:ffe1) mqprio
 | | +---(8001:2) mqprio
 | | +---(8001:ffe0) mqprio
 | +---(8001:1) mqprio
```
Config interface: Credit-based shaper

• For credit-based shaping (802.1Qav) we developed the **cbs qdisc**.
  o Available from kernel 4.15.
    ■ debuted with Intel i210 support only, but more to follow.
  o Provides both HW offloading and SW fallback.
  o Config parameters derived directly from Annex L of IEEE 802.1Q.
  o Remember: CBS is bandwidth-centric.

• Example: configure CBS for traffic class 1 (priority 2)

```bash
$ tc qdisc replace dev enp2s0 parent 8001:2 cbs
  locredit -1470 hicredit 30
  sendslope -980000
  idleslope 20000 offload 1
$ tc -g qdisc show dev enp2s0
qdisc mqprio 8001: root tc 3 (...)
  queues:(0:0) (1:1) (2:3) (...)
qdisc fq_codel 0: parent 8001:1
  limit 10240p (...)
qdisc cbs 8002: parent 8001:2
  hicredit 30 locredit -1470
  sendslope -980000 idleslope 20000 offload 1
```
Config interface: Time-based Sched.

- For time-based scheduling, we are developing the tbs qdisc and the SO_TXTIME socket option.
  - Co-developing with Richard Cochran (linuxptp maintainer).
  - Provides both HW offloading and SW fallback.
  - Trending well, currently on its RFC v3
    - [https://patchwork.ozlabs.org/cover/882342/](https://patchwork.ozlabs.org/cover/882342/)
    - debuted with Intel i210 support only, but more to follow.

- tbs qdisc can:
  - hold packets until their TxTime minus a configurable delta factor
  - sort packets based on their TxTime
    - optional, and only before they are sent to the device queue

- tbs is time-centric
  - Requires a per-packet timestamp.

- Example: configure TBS for traffic class 0 (priority 3)

```bash
$ tc qdisc replace dev enp2s0 parent 8001:1 tbs clockid CLOCK_REALTIME delta 150000 sorting enable

$ tc -g qdisc show dev enp2s0
(...)
qdisc tbs 8003: parent 8001:1 clockid CLOCK_REALTIME delta 150000 offload on sorting on
qdisc cbs 8002: parent 8001:2 hicredit 30 (...)
```
Data path: Socket interface

• We use regular sockets for transmitting data.
• TBS
  o a new socket option (SO.TXTTIME) is used for enabling the feature for a given socket.
  o A cmsg header is used for setting a per-packet txtime, and a drop_if_late flag.
    ■ reference clockid_t will become a socket option argument

(...)
clock_gettime(CLOCK_REALTIME, &ts);
__u64 txtime = ts.tv_sec * 1000000000ULL + ts.tv_nsec;
cmsg = CMSG_FIRSTHDR(&msg);
cmsg->cmsg_level = SOL_SOCKET;
cmsg->cmsg_type = SCM_TXTIME;
cmsg->cmsg_len = CMSG_LEN(sizeof(__u64));
*((__u64 *) CMSG_DATA(cmsg)) = txtime;
cmsg = CMSG_NXTHDR(&msg, cmsg);
cmsg->cmsg_level = SOL_SOCKET;
cmsg->cmsg_type = SCM_DROP_IF_LATE;
cmsg->cmsg_len = CMSG_LEN(sizeof(uint8_t));
*((uint8_t *) CMSG_DATA(cmsg)) = 1;

(...)
const int on = 1;
setsockopt(fd, SOL_SOCKET, SO_TXTIME, &on, sizeof(on))
Data path: Socket interface

• Classifying traffic:
  o The socket option **SO_PRIORITY** is used to flag all packets with an specific priority.
    ■ Preferred method, but iptables or net_prio cgroup can be used.
  o The priority is later used as the PCP field of the VLAN tag of the ethernet header.
  o Steers all traffic from the socket into the correct HW Tx queue.
    ■ Remember: we have setup a mapping for that with the mqprio qdisc.
## Results - TxFTime Based Scheduling

### SW

<table>
<thead>
<tr>
<th>min (ns):</th>
<th>+4.820000e+02</th>
<th>max (ns):</th>
<th>+9.999300e+05</th>
<th>pk-pk:</th>
<th>+9.994480e+05</th>
<th>mean (ns):</th>
<th>+3.464421e+04</th>
<th>stddev:</th>
<th>+1.305947e+05</th>
<th>count:</th>
<th>600000</th>
</tr>
</thead>
</table>

### TBS

<table>
<thead>
<tr>
<th>min (ns):</th>
<th>+1.510000e+02</th>
<th>max (ns):</th>
<th>+9.977030e+05</th>
<th>pk-pk:</th>
<th>+9.975520e+05</th>
<th>mean (ns):</th>
<th>+1.416511e+04</th>
<th>stddev:</th>
<th>+5.750639e+04</th>
<th>count:</th>
<th>600000</th>
</tr>
</thead>
</table>

- **DUT:** i5-7600 CPU @ 3.50GHz, kernel 4.16.0-rc2+ with about 50 usec maximum latency under cyclic test.
- ptp4l + phc2sys
- packet size: 322 bytes all headers included
What about the userspace?

- **OpenAVNU**
  - Evolution of OpenAVB, maintained by the AVNU Alliance members
  - Provides daemons, libs, examples, frameworks
    - gPTPd: 802.1AS
    - MRPd: SRP daemon
  - Mostly focused on the Pro A/V domain
  - Recent contribution from Intel: **libavtp**
    - Provides packetization for applications that use AVTP as a transport

- **linuxptp**
  - ptp4l: Precision Time Protocol implementation for Linux
  - phc2sys: Synchronizes the PTP Hardware Clock to the System Clock
Objectives

• Provide a (very) brief introduction to Time-Sensitive Networking
• Present the current upstream TSN SW architecture
• Discuss the challenges ahead
Config interfaces: 802.1Qbv and 802.1Qbu

• Qbv: Enhancements to Scheduled Traffic
• Qbu: Frame Preemption
• We’ve shared ideas for a new qdisc-based interface before: ‘taprio’.
  o A time-aware version of mqprio.
  o Part of the CBS RFC v1: https://patchwork.ozlabs.org/cover/808504/
  o Push-back: there were no NICs for end stations with support for these standards.
  o Providing a SW fallback is required, so we may re-consider an ethtool based interface instead.
• TBS could be used, but that requires a scheduler for converting the per-port schedule from Qbv into a per-packet txtime.
Data path: Looking ahead

- Linux network stack is *very good* for throughput.
  - TSN will require more: **bounded low latency**

- XDP
  - eXpress Data Path
    - High performance data path for Rx.
    - Does not bypass the kernel, but avoids allocation of skbuffs.
    - [https://www.iovisor.org/technology/xdp](https://www.iovisor.org/technology/xdp)

- AF_PACKET_V4 -> **AF_XDP**
  - New socket family aiming to improve throughput / latency by reusing XDP hooks.
  - Zerocopy will be finally allowed, but only with driver support.
  - [https://lwn.net/Articles/737947/](https://lwn.net/Articles/737947/)
  - [https://patchwork.ozlabs.org/cover/867937/](https://patchwork.ozlabs.org/cover/867937/)
Wrap up

- TSN aims to provide bounded latency on Ethernet based LANs.
- SW interfaces for Linux are starting to become available upstream starting with the cbs and tbs qdiscs.
- Future work aims to address other traffic shapers (802.1Qbv / Qbu).
- Low latency is (probably) an issue. There are efforts trying to reduce the bounded worst-case latency of the Linux network stack: AF_XDP.
- Userspace building blocks are also gaining traction.
  - OpenAVNU is becoming the consolidator of TSN SW components for userspace.
- Zephyr will have TSN support soon!
Call to Action

• Enable support on your upstream drivers.
• Have use cases? Engage on the netdev discussions!
• Have TSN products? Help us testing by using the upstream interfaces.
• Contribute code and bug-fixes!
More References

• Mann’s Plumbers 2012 talk: https://linuxplumbers.ubicast.tv/videos/linux-network-enabling-requirements-for-audiovideo-bridging-avb/
• Austad’s TSN driver RFC v2: https://lkml.org/lkml/2016/12/16/453
• Austad’s ELC 2017.2 Presentation: https://www.youtube.com/watch?v=oxURD2rr4Y4
• CBS v9: https://patchwork.ozlabs.org/cover/826678/
• TBS RFC v2: https://patchwork.ozlabs.org/cover/862639/
• mqprio man page: http://man7.org/linux/man-pages/man8/tc-mqprio.8.html
• OpenAVNU: https://github.com/AVnu/OpenAvnu
Obrigado!

jesus.sanchez-palencia@intel.com