Deterministic Networking for Real-Time Systems

(Using TSN and DetNet)

Henrik Austad
haustad@cisco.com

Cisco Systems

Prague, Oct 25, 2017

about:henrik

- Cisco Collaboration, Audio group at Lysaker, Norway
- All things Linux
- Real-time tweaking and tuning
- Staring at traces
- AVB/TSN and DetNet
- *"I have a script for that somewhere.."*

https://projectworkplace.cisco.com

SX80 Codec backplane

Real-time systems

- Correctness of system not only depends on the logical result, but also on *time of arrival*
- When an rt-system fails, **bad** things typically happen
- Not a trivial problem for simple systems
- Quite difficult for multicore (this subject is an entire talk by itself..)
- *Heterogenous* multicore, because apparently pain is temporary..

"I want to make a distributed real-time system using a packet-switched network with off-the-shelf hardware!"

Challenge 1 - Time

Real-time systems have *very* strict requirements. Adding *distributed* to the mix;

- No 2 clocks ever run at the same rate
- Complex SW introduces latencies, giving rise to more uncertainties
- Unpredictable network delays add insult to injury
- NTP can du sub-ms accuracy *if* LAN, low traffic, full moon, Saturn and Venus in phase etc
- PTP has already fixed this, enabling sub μs accuracy
- HW Support in both the network and in the end-stations is **strongly** preferred

We are not going to cover PTP in more detail in this talk.

Challenge 2 - Reliable[1] Packet Switched Networks (PSN)

- Acceptable for a PSN to drop a frame on collision
- Bridges have finite buffers (can lead to framedrops)
- Next frame out on a port is probably FIFO, *perhaps* it supports VLAN PCP
- Jumboframes will block all others until tx completes

- Cannot express "arrive no later than" to the network
- ... nor indicate "send frame at time X" to the NIC
- And not "give me Y kbps of bandwidth and *never*, **ever** drop a frame"

[1] "Reliable" as in "*real-time systems*"-reliable

Challenge 3+n (where n $\in \mathbb{N}^0$)

Your application will probably have some other application-specific issues that are amplified by being distributed.

These are left as an exercise for the reader ;)

AVB/TSN

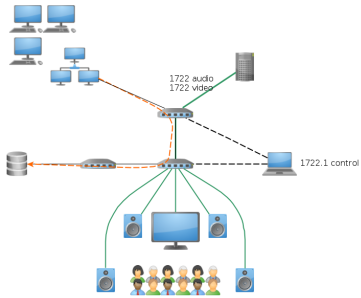
Source: https://goo.gl/images/13846A

Audio/Video Bridging - AVB

This started out as a set of open standards that aimed to solve the network challenge for media over PSN.

- Idea: AD/DAs are cheap. Ethernet MACs are cheap...
- Why not handle A/V *digitally* as early as possible?
  $\Rightarrow$ Started out as Pro Audio/Video only.

- Media must have guaranteed delivery, best-effort not acceptable
- Allows for very flexible setups (can easily reroute and duplicate streams)
- High (audio-)capacity in a single cable
- As units grow smaller with more processing power, being restricted by the physical dimensions of the backplane is not otimal
- Can co-exist with BE traffic

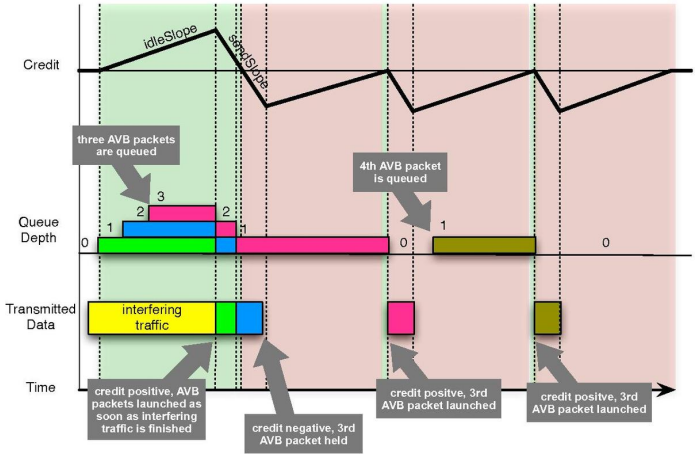


Office network combined with movie streaming

AVB

- Initial target was simple systems (mic, speakers, DSPs)
- Started out with etherframes (L2) only
- Specified PTPv2 profile (gPTP)
- Uses Stream Reservation Protocol (SRP) to express requirements to the network
- Reliable, low-jitter streams with guaranteed BW most important
- An easy way to connect end-stations (IEEE 1722.1)
- Security not that important (subnet only)

The Credit Based Shaper (CBS)



https://en.wikipedia.org/wiki/File:Traffic-shaping.pdf

TSN: Time Sensitive Networking

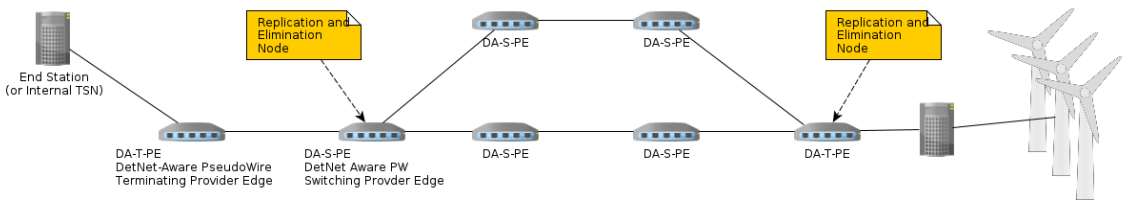
People started to use AVB for all sorts of crazy things, so "AVB" proved to be a misnomer. Renamed to TSN in Nov. 2012

- Pro-AV
- Consumer AV
- Infotainment systems (cars, messaging boards, theme-parks, ...)
- Automotive (ABS breaks, control systems, monitoring, etc)
- Industrial applications (e.g. Control systems/Robotics, IIoT - "Industry 4.0")
- Combine Operation Technology (OT) networks with IT networks

Focus no longer on just to provide reliabe, jitter-free streams, but also on time of transmission, frame preemption, larger networks, path redundancy.

IETF and Deterministic Networking (DetNet) WG

- Large network oriented (WAN support important)
- Pseudo-wire encapsulation (invisible to the end-station)
- Set of guidelines to acheive deterministic behavior (rather than hard requirements)
- Multipath routing (replication and elimination)
- Requires Central controller (not like 1722.1 where 'anyone' can configure)
- Brings a somewhat larger security concern to the table

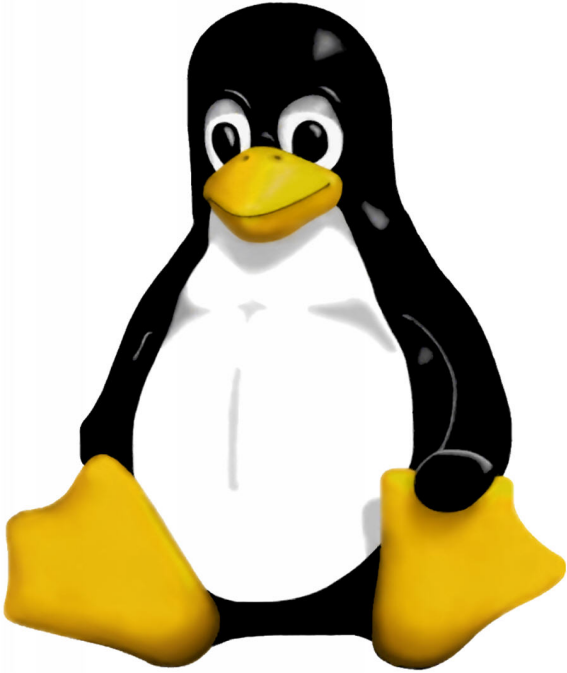


https://datatracker.ietf.org/wg/detnet/about/

DetNet motivation

- Broadcasting (digital TV, PA at large venues)
- Electrical utilities (SmartGrid, coordinate producers, grid frequency)
- Building Automation Systems (sensors, HVAC)
- When end-to-end latency is important

- Replacing proprietary deterministic networks
- Same network for both critical and Best-Effort traffic



TSN in the Linux kernel

- Previous approach was media centric
- A *lot* of central pieces missing (hacked into network and media)
- Mostly done via SW inside kernel (timing and best-effort tx of frames)
- Made it possible to use *whatever* media-app to play audio over the network, which was *fun*.
- TSN is more than stream reservation.
- Currently, only Intel's i210 is available in a PCI-e formfactor, so this is the NIC used for kernel testing.

But things are happening!

Time-triggered TSN driver

- Author: Richard Cochrane (linutronix.de)
- `https://lkml.org/lkml/2017/9/18/76`
- A Time-triggered transmit approach (not bandwith centric)
- Uses i210's LaunchTime (32ns granularity time-triggered tx)
- Can specify "send frame at time X" with great accuracy

```c
/*
 * Credits: Richard Cochran <rcochran@linutronix.de>
 *
 * Lots left out, only the essentials are back
 */

int setup(void)
{
        int fd = socket(PF_INET, SOCK_DGRAM, IPPROTO_UDP);

        /* more init, and tag with SO_TXTIME */
        setsockopt(fd, SOL_SOCKET, SO_TXTIME, &on, sizeof(on));
        return fd;
}
int send(int fd, void *buf, int len, __u64 txtime)
{
        /* ... */

        /* set txtime in a cmsg, part of the message */
        cmsg = CMSG_FIRSTHDR(&msg);
        cmsg->cmsg_level = SOL_SOCKET;
        cmsg->cmsg_type = SO_TXTIME;
        cmsg->cmsg_len = CMSG_LEN(sizeof(__u64));
        *((__u64 *) CMSG_DATA(cmsg)) = txtime;

        /* finally, send it, will be sent at txtime */
        sendmsg(fd, &msg, 0);
}
```

SO_TXTIME test

- 2 machines, time synchronized using PTP, connected via crossover cat5)
- DUT running PREEMPT_RT 4.9.40-rt30
- Using i210 NIC
- Look at time of arrival compared to expected arrival
- Compares 2 modes, sw triggered tx-time (set a timer, send a frame) and HW triggered tx.

SO_TXTIME results

	plain preempt_rt @1ms	so_txtime @1ms	txtime @ 250 us
min:	19 408 ns	472 ns	472 ns
max:	75 560 ns	568 ns	576 ns
pk-pk:	56 152 ns	96 ns	104 ns
mean:	32 928 ns	507.23 ns	507.36 ns
stddev:	6 514.709	13.10849	15.07144
count:	600 000	600 000	2400000

Results from R. Cochrane. 10min testrun, times are delta from expected arrival time for frames, all values positive (e.g. no frames arrived earlier than specified) obtained from `https://lkml.org/lkml/2017/9/18/76`

Note: a 2m Cat5-cable has a 10ns propagation delay.

Credit Based Shaper

- Authors: Andre Guedes, Ivan Briano, Jesus Sanchez-Palencia and Vinicius Gomes (Intel)
- Currently on v9 (based on netdev-next)
  `https://www.spinics.net/lists/netdev/msg460869.html`
- Solving constant bandwidth ("classic AVB")
- Implemented as a Qdisc scheduler and an update to i210 driver (via `.ndo_setup_tc`)
- Uses mqprio as root qdisc
- Use tc to assign a PCP to a hw-queue
- Tie `sch_cbs` to each queue afterwards and specify CBS parameters
- All frames with a given priority will be handled by this scheduler.

cbs hw offload

```
# Create 4 separate queues
tc qdisc replace dev eth2 parent root mqprio num_tc 4 \
   map 3 3 1 0 2 2 2 2 2 2 2 2 2 2 2 2 queues 1@0 1@1 1@2 1@3 hw 0

tc -g class show dev eth2
+---(8008:ffe3) mqprio
|     +---(8008:4) mqprio # Tx-3
|
+---(8008:ffe2) mqprio
|     +---(8008:3) mqprio # Tx-2
|
+---(8008:ffe1) mqprio
|     +---(8008:2) mqprio # Tx-1, have CBS and time-triggered launch
|
+---(8008:ffe0) mqprio
      +---(8008:1) mqprio # Tx-0, have CBS and time-triggered launch

tc qdisc add dev eth2 parent 8008:1 cbs idleslope 20000 sendslope -980000 \
   hicredit 30 locredit -1470 offload 1
```

```c
/*
 * excerpt from tsn_talker
 * Published by Guedes et. al, intel 2017
 *
 * https://www.spinics.net/lists/netdev/msg460869.html
 */
int send_cbs(void)
{
        int fd, prio=3;

        /* open socket and init, removed for brevity */
        fd = socket(AF_PACKET, SOCK_DGRAM, htons(ETH_P_TSN));
        setsockopt(fd, SOL_SOCKET, SO_PRIORITY, &priority,
                sizeof(priority));

        sendto(fd, data, size, 0, size, &addr, sizeof(addr));

        close(fd);
}
```

Testing sch_cbs (not up to scientific standards!)

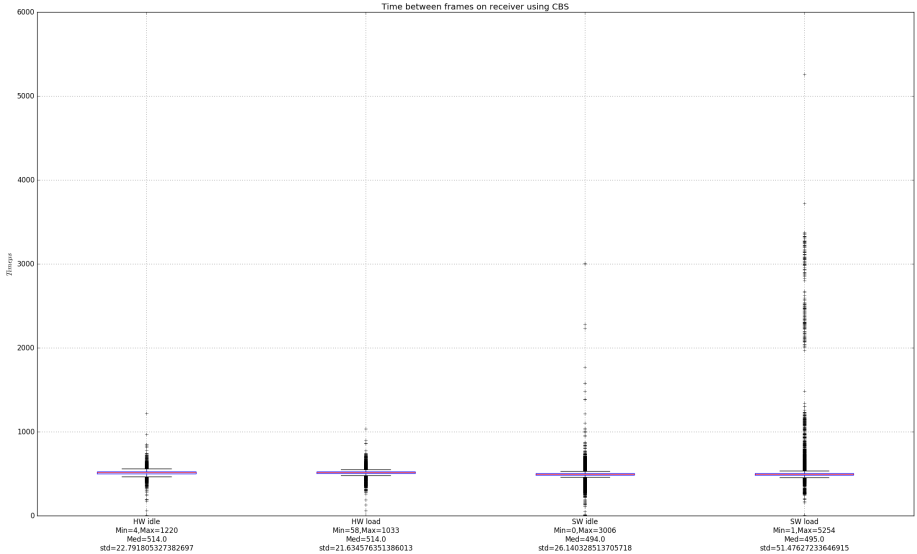
Guedes et. al provided a set of scripts to test scheduler (tsn_listener and tsn_talker), that has been slightly modified for the tests.

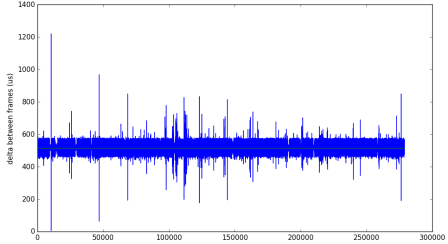
Device under Test:

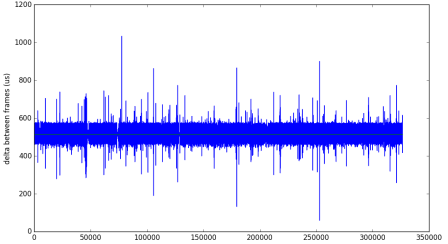
- old core2duo, 1.8GHz, 8GB, 7200rpm disk, running Linux v4.14-rc4 (netdev-next)
- Intel i210 PCI-e NIC
- `chrt --fifo 50 ./tsn_talker -i eth2 -d 14:da:e9:2b:0a:c1 -s1250 -p3`
- Generated load using "`make -j16 all`" of a Linux kernel

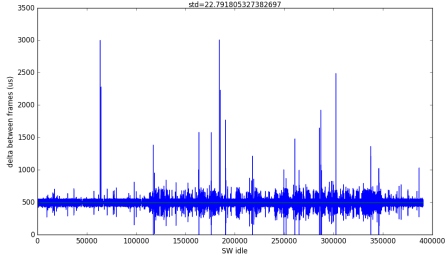
Receiver:

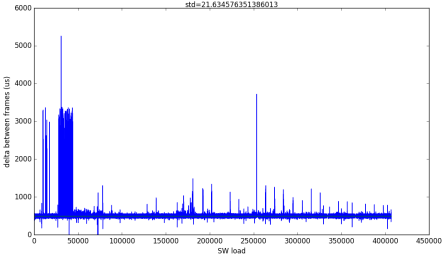
- Intel i7 2700k, 16GB, Intel 82579V (e1000e), ssd, debian stable (linux v 3.16.0)
- 2 switches between (no vlan)
- added tagging of trace_marker to tsn_listener
- `time sudo taskset -c 1 chrt --fifo 50 ./tsn_listener -i eth2 -s 1250 -t`
- irq for eth2 bound to core 0


Time between frames on receiver using CBS
6000
5000
4000
3000
2000
1000
0
HW idle
Min=4,Max=1220
Med=514.0
std=22.791805327382697
HW load
Min=58,Max=1033
Med=514.0
std=21.634576351386013
SW idle
Min=0,Max=3006
Med=494.0
std=26.140328513705718
SW load
Min=1,Max=5254
Med=495.0
std=51.47627233646915


delta between frames (us)
HW idle
Min=4,Max=1220
Med=514.0
std=22.791805327382697


delta between frames (us)
HW load
Min=58,Max=1033
Med=514.0
std=21.634576351386013


delta between frames (us)
SW idle
Min=0,Max=3006
Med=494.0
std=26.140328513705718


delta between frames (us)
SW load
Min=1,Max=5254
Med=495.0
std=51.476272336469l5

Thank you!

TSN - IEEE (for reference)

- 802.1BA - AVB Systems
- 802.1AS-2011 - gPTP
- 802.1Q-2014 (Sec 34: FQTSS - .1Qav, 35: Stream Reservation Protocol - .1Qat)
- 1722 / 1722a d16 AVTP
- 1733 (AVTP over RTP)
- 1722.1 Discovery and enumeration
- 802.1Qbu Frame preemption
- 802.1Qbv Time triggered transmission
- 802.1QCB Frame replication and elimination
- 802.1Qch Cyclic Queueing and forwarding
- 802.1Qcp Yang modelling
- ...