

TSN mit Linux

Embedded Software Engineering Kongress 2019

Kurt Kanzenbach

Linutronix GmbH

03.12.2019

Agenda

- 1 Motivation
- 2 Zeitsynchronisation
- 3 Traffic Scheduling
 - Earliest Tx Time
 - Credit Based Shaper
 - Time Aware Shaper
 - Switches
- 4 Konfiguration
- 5 Fazit

Feldbusse

Feldbusse

- ☞ System zur Vernetzung von Geräten
- ☞ *Echtzeitfähig und deterministisch*
- ☞ Industrieanlagen, Automotive, Luftfahrt, . . .



Ethernet

Vorteile von Ethernet Feldbussen

- + Standard Hardware
- + Überall verfügbar
- + Geschwindigkeit
- + Interoperabilität

Ether**CAT**[®] 

PROFI[®]
NET

sercos
the automation bus

Time Sensitive Networking

Was ist TSN?

- ☞ Time Sensitive Networking
- ☞ *Deterministisches Ethernet*
- ☞ Definiert über IEEE 802.1Q
- ☞ *Offener und gemeinsamer Standard*

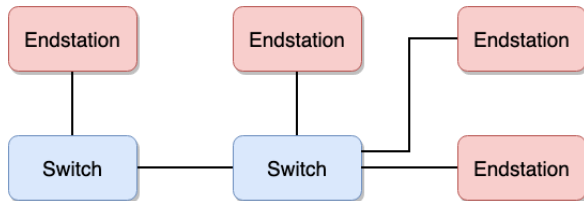
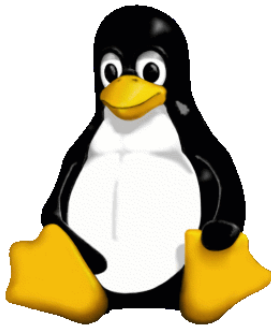


Abbildung: TSN Netzwerk

Linux

Warum Linux?

- ☞ **Offen**
- ☞ **Hardware Unterstützung**
- ☞ **Große Community**
- ☞ ***Echtzeitfähig* mit PREEMPT_RT [1]**



Linux TSN Lösungen I

Bisherige TSN Implementierungen

- 📄 OpenIL [2]
- 📄 OpenAVNU [3]
- 📄 Acontis [4]
- 📄 ...

Gemeinsamkeiten

- 📄 Modifikationen am Kernel
- 📄 Erstellung eigener Interfaces und Tools

Linux TSN Lösungen II

Vorteile

+ Funktional?

Nachteile

- Wartbarkeit
- Abhängigkeit vom Hersteller
- Keine Standard Tools und Interfaces
- Interoperabilität

⇒ Lösung: **Standard Linux!**

Agenda

- ① Motivation
- ② Zeitsynchronisation
- ③ Traffic Scheduling
 - Earliest Tx Time
 - Credit Based Shaper
 - Time Aware Shaper
 - Switches
- ④ Konfiguration
- ⑤ Fazit

Precision Time Protocol

Precision Time Protocol

- ☞ Synchronisation von Knoten via Ethernet
- ☞ Analog zu NTP
- ☞ Genauigkeit im Bereich $< 1 \mu\text{s}$ möglich
- ☞ Standards: IEEE 1588, 802.1AS, 802.1ASRev
- ☞ Profile: Default, Telecom, gPTP, ...

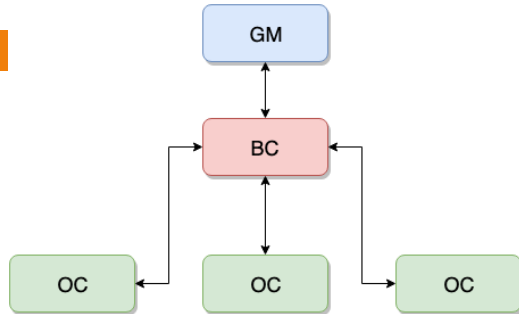


Abbildung: PTP Domain

Linux und PTP

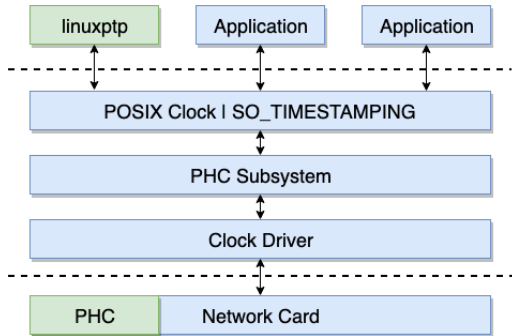





Abbildung: Linux PTP Architektur

PTP Hardware Clocks

Linux PTP Hardware Clocks [5]

-  **Darstellung: Character Devices**
-  **API: Posix Clock**
-  **Kontrolle: ioctl()**

Kernel API

```

1 struct ptp_clock_info {
2     int (*adjfine)(struct ptp_clock_info *ptp, long scaled_ppm);
3     int (*adjfreq)(struct ptp_clock_info *ptp, s32 delta);
4     int (*_gettime64)(struct ptp_clock_info *ptp, struct timespec64 *ts);
5     int (*settime64)(struct ptp_clock_info *p, const struct timespec64 *ts);
6 };
    
```

Timestamping

Linux Zeitstempel [6]

- ☒ **Interface:** SO_TIMESTAMPING
- ☒ **Erzeugung:**
 - SOF_TIMESTAMPING_TX/RX_HARDWARE
 - SOF_TIMESTAMPING_TX/RX_SOFTWARE
- ☒ **Receive:** Über Control Messages (CMSG)
- ☒ **Transmit:** Über die Error Queue (MSG_ERRQUEUE)

linuxptp

Linuxptp [7]

- Linux User Space PTP Stack
- Unterstützt:
 - IEEE 1588: Ordinary Clock, Boundary Clock und Transparent Clock
 - 802.1AS als End Device
- Open Source, Lizenz: GPL
- Nutzt Linux PHC und SO_TIMESTAMPING Interface

Anwendung

System- und Netzwerkzeit

- 📄 **ptp4l: Regelt die PTP Hardware Clocks**
- 📄 **phc2sys: Synchronisiert die Netzwerkzeit auf das System**

Anwendung

- 📄 **Posix API: `clock_nanosleep()`**
- 📄 **Clock ID: `CLOCK_TAI`**

Agenda

- ① Motivation
- ② Zeitsynchronisation
- ③ **Traffic Scheduling**
 - Earliest Tx Time
 - Credit Based Shaper
 - Time Aware Shaper
 - Switches
- ④ Konfiguration
- ⑤ Fazit

Überblick

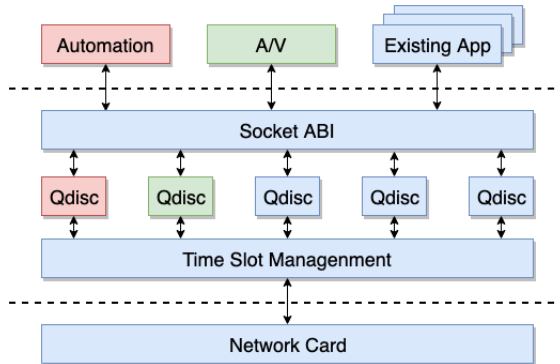


Abbildung: Linux TSN Integration [8]

Agenda

- 1 Motivation
- 2 Zeitsynchronisation
- 3 Traffic Scheduling
 - Earliest Tx Time
 - Credit Based Shaper
 - Time Aware Shaper
 - Switches
- 4 Konfiguration
- 5 Fazit

Earliest Tx Time

Earliest Tx Time [9]

- ☒ **Sende Paket an Zeitpunkt X**
- ☒ **Deadline Mode verfügbar**
- ☒ **Implementierung Kernel: Qdisc**
- ☒ **Implementierung Userspace: Socket Option SO_TXTIME**

Beispiel

ETF Beispiel

```

1 # Setup multi queue
2 tc qdisc add dev eth0 handle 100: parent root mqprio num_tc 3 \
3     map 2 2 1 0 2 2 2 2 2 2 2 2 2 2 2 2 \
4     queues 1@0 1@1 2@2 \
5     hw 0
6
7 # Configure ETF for TC 0
8 tc qdisc replace dev eth0 parent 100:1 etf \
9     clockid CLOCK_TAI \
10    delta 300000

```

Userspace

ETF Beispiel

```

1  /* Prepare message */
2  msg.msg_control = control;
3  msg.msg_controllen = sizeof(control);
4
5  /* Add tx time */
6  cmsg = CMSG_FIRSTHDR(&msg);
7  cmsg->cmsg_level = SOL_SOCKET;
8  cmsg->cmsg_type = SO_TXTIME;
9  cmsg->cmsg_len = CMSG_LEN(sizeof(__u64));
10 *((__u64 *)CMSG_DATA(cmsg)) = txtime;
11
12 ret = sendmsg(fd, &msg, 0);

```

Hardware Offloading

TAPRIO Hardware Offloading

- ☞ **TC Parameter:** offload
- ☞ **Treiber Callback:** `ndo_setup_tc()`; **Typ:** `TC_SETUP_QDISC_ETF`
- ☞ **Treiber Offload Parameter:** **struct** `tc_ETF_qopt_offload`;

Treiber




- ☞ Intel IGB, i210

Agenda

- 1 Motivation
- 2 Zeitsynchronisation
- 3 Traffic Scheduling
 - Earliest Tx Time
 - Credit Based Shaper**
 - Time Aware Shaper
 - Switches
- 4 Konfiguration
- 5 Fazit

Credit Based Shaper

CBS [10]

-  Bandbreitenlimitierung
-  Qdisc
-  Hardware Offloading verfügbar

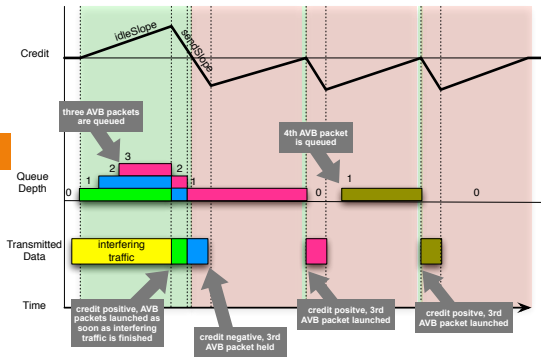


Abbildung: Credit based shaper [11]

Beispiel

CBS Beispiel

```

1 # Setup multi queue
2 tc qdisc add dev eth0 handle 100: parent root mqprio num_tc 3 \
3     map 2 2 1 0 2 2 2 2 2 2 2 2 2 2 2 2 \
4     queues 1@0 1@1 2@2 \
5     hw 0
6
7 # Configure CBS for reserving 20Mbit/s of a 1Gbit/s connection
8 tc qdisc replace dev eth0 parent 100:1 cbs \
9     locredit -1470 hicredit 30 \
10    sendslope -980000 idleslope 20000

```

Hardware Offloading

CBS Hardware Offloading

- ☞ **TC Parameter:** offload
- ☞ **Treiber Callback:** `ndo_setup_tc()`; **Typ:** `TC_SETUP_QDISC_CBS`
- ☞ **Treiber Offload Parameter:** `struct tc_cbs_qopt_offload;`

Treiber

- ☞ Intel IGB, TI CPSW, STMMAC, NXP ENETC

Agenda

- 1 Motivation
- 2 Zeitsynchronisation
- 3 Traffic Scheduling
 - Earliest Tx Time
 - Credit Based Shaper
 - Time Aware Shaper**
 - Switches
- 4 Konfiguration
- 5 Fazit

TAPRIO

TAPRIO [12]

- ☞ Time slot management
- ☞ Qdisc
- ☞ Pro port

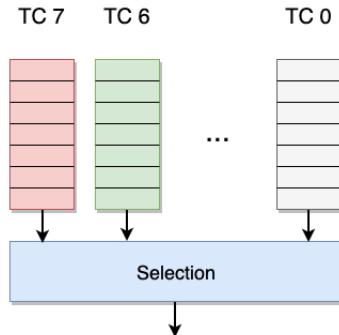


Abbildung: Traffic Klassen

Beispiel I

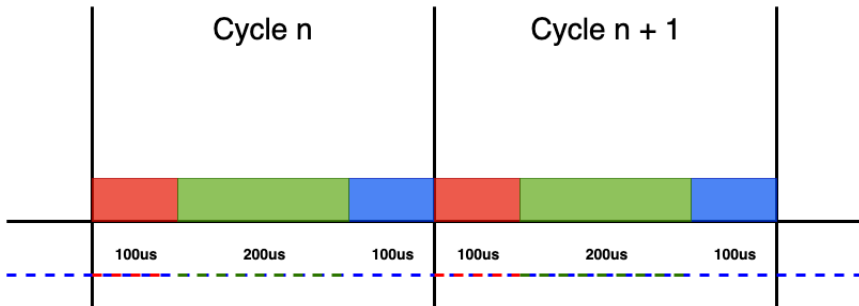


Abbildung: Beispiel Zyklus

Beispiel II

```

_____ TAPRIO Beispiel _____
1 # Setup cycle: TC 0: 200us; TC 1: 400us; TC 2: 200us
2 tc qdisc replace dev eth0 parent root handle 100 taprio \
3     num_tc 3 \
4     map 2 2 1 0 2 2 2 2 2 2 2 2 2 2 2 2 \
5     queues 1@0 1@1 2@2 \
6     base-time 1570615200123456789 \
7     sched-entry S 01 200000 \
8     sched-entry S 02 400000 \
9     sched-entry S 04 200000 \
10    clockid CLOCK_TAI \
11    flags 0x02

```

Hardware Offloading

TAPRIO Hardware Offloading

- ☞ **TC Parameter:** flags
- ☞ **Treiber Callback:** `ndo_setup_tc()`; **Typ:** `TC_SETUP_QDISC_TAPRIO`
- ☞ **Treiber Offload Parameter:** **struct** `tc_taprio_qopt_offload`;

Treiber

- ☞ **sjal105, NXP ENETC**




Agenda

- 1 Motivation
- 2 Zeitsynchronisation
- 3 Traffic Scheduling
 - Earliest Tx Time
 - Credit Based Shaper
 - Time Aware Shaper
- 4 Konfiguration
- 5 Fazit



Switches

Switch Konfiguration

Wie?

-  Bootloader?
-  Linux Kernel?
-  Spezielle Applikationen?

Lösung

-  Linux enthält Switch Frameworks
-  Darstellung von Switch Ports als reguläre **Netzwerkinterfaces**

Distributed Switch Architecture

Distributed Switch Architecture [13]

- ☞ Switch Framework
- ☞ Treiber Modell
- ☞ Kaskadierte Aufbauten möglich
- ☞ Voraussetzung: CPU Port

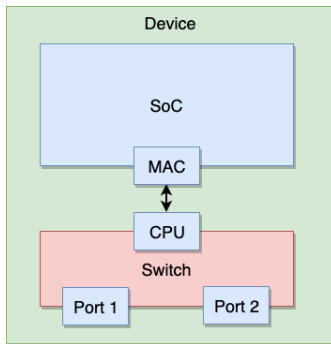


Abbildung: Struktur

Switchdev

Switchdev [14]

- ❏ Switch Framework
- ❏ Zustandslos
- ❏ Ziel: Hardware Offloading
- ❏ DSA nutzt switchdev

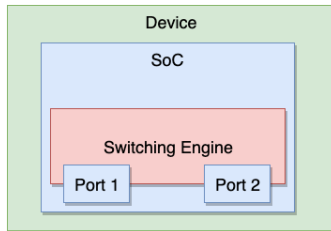


Abbildung: Struktur

Linux view

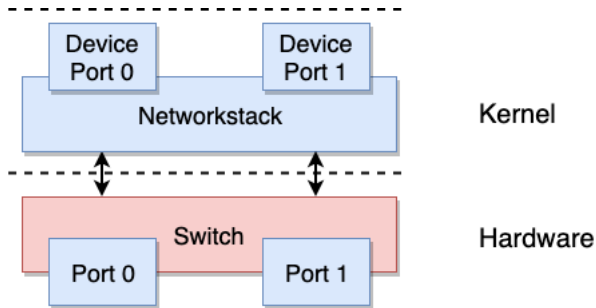


Abbildung: Switch Linux view

Agenda

- 1 Motivation
- 2 Zeitsynchronisation
- 3 Traffic Scheduling
 - Earliest Tx Time
 - Credit Based Shaper
 - Time Aware Shaper
 - Switches
- 4 Konfiguration
- 5 Fazit

Überblick

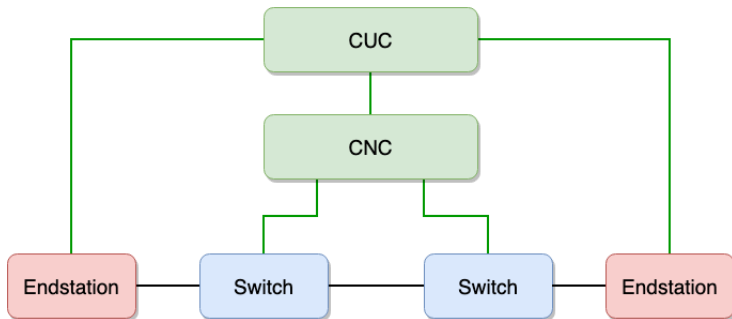


Abbildung: TSN Konfiguration

Konfiguration

Komponenten

- ☞ **Centralized User Configuration (CUC):**
 - Topologie Discovery
 - Kommunikation mit den End Devices
 - Übergabe der TSN *Streams* an die CNC
- ☞ **Centralized Network Configuration (CNC):**
 - Kommunikation mit den Switchen
 - Berechnung und Deployment des Schedules

Streams

Stream

- StreamID
- Destination MAC
- VLAN ID
- Priority (PCP)
- Intervall
- QoS Werte

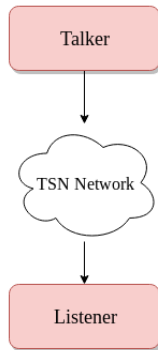


Abbildung: TSN Stream

Linux

Konfiguration

- ☒ Konfiguration ist größtenteils Software:
 - CNC
 - CUC
 - Gegenpart auf den End Devices

Implementierungen

- ☒ Viele Hersteller haben eigene Implementierungen
- ☒ Open Source Lösungen?

Agenda

- ① Motivation
- ② Zeitsynchronisation
- ③ Traffic Scheduling
 - Earliest Tx Time
 - Credit Based Shaper
 - Time Aware Shaper
 - Switches
- ④ Konfiguration
- ⑤ Fazit

TSN Linux Status | Zeitsynchronisation

Linux PTP

- ☒ **Linux PTP Unterstützung ist hervorragend:**
 - Hardware Timestamping
 - linuxptp
- ☒ **Standards: IEEE 1588, 802.1AS**

Was fehlt?

- ☒ **Unterstützung für 802.1AS Time Aware Bridges ⇒ WIP**
- ☒ **802.1ASRev**

TSN Linux Status | Traffic Scheduling

Linux Traffic Scheduling

- Linux Netzwerkstack ist für TSN geeignet:
 - Verschiedene Shaper, Scheduler, Filter, . . .
 - Tooling: iproute2, tcpdump, ethtool, eBPF, . . .
 - Hardware Offloading
 - Support für Switche

Was fehlt?

- Unterstützung für Streams

TSN Linux Status | Konfiguration

Konfiguration

- ❏ Konfiguration von TSN Netzwerken ist **nicht** trivial
- ❏ Offene Lösungen fehlen
- ❏ Diverse Standards
- ❏ ⇒ **Baustelle**

AccessTSN

AccessTSN Projekt

- 📁 Research Projekt
- 📁 Entwicklung einer generischen TSN Lösung
- 📁 Einsatz von Open Source Komponenten
- 📁 Partner: ISW, Hochschule Offenburg, Hirschmann und Linutronix



Abbildung: AccessTSN

Outlook

Zukunft

- ☞ Offenen Punkte für PTP und Traffic Scheduling
- ☞ Konfiguration

Erwägungen





- ☞ PREEMPT_RT
- ☞ eXpress Data Path
- ☞ devlink

Fazit



Fazit

- 📄 **Standard** Linux für TSN Netzwerke möglich
- 📄 Verwendung der bestehenden Linux Interfaces und Tools
- 📄 Konfiguration ist offener Punkt
- 📄 *Contribute to Upstream, Influence the Interfaces*





Referenzen I

-  Thomas Gleixner. (2019, Dec.) PREEMPT_RT Wiki. [Online]. Available: <https://wiki.linuxfoundation.org/realtime/start>
-  openil.org. (2019, Oct.) Open Industrial Linux. [Online]. Available: <https://www.openil.org/>
-  OpenAvnu. (2019, Oct.) OpenAvnu. [Online]. Available: <https://github.com/AVnu/OpenAvnu>
-  acontis technologies GmbH. (2019, Oct.) TSN (Time Sensitive Networking) Software Stack. [Online]. Available: <https://www.acontis.com/en/tsn.html>



Referenzen II

-  **Richard Cochran. (2019, Oct.) PTP (Precision Time Protocol) Documentation. [Online]. Available: <https://www.kernel.org/doc/Documentation/ptp/ptp.txt>**
-  **Patrick Ohly. (2019, Oct.) The Linux Kernel Archives. [Online]. Available: <https://www.kernel.org/doc/Documentation/networking/timestamping.txt>**
-  **Richard Cochran. (2019, Oct.) The Linux PTP Project. [Online]. Available: <http://linuxptp.sourceforge.net/>**
-  **Thomas Gleixner, “Evolution and current status of TSN in Linux,” Nov. 2018.**

Referenzen III

-  **Jesus Sanchez-Palencia, Vinicius Costa Gomes. (2019, Oct.) ETF (Earliest TxTime First) Documentation. [Online]. Available: <http://man7.org/linux/man-pages/man8/tc-etf.8.html>**
-  **Vinicius Costa Gomes. (2019, Oct.) CBS (Credit Based Shaper) Documentation. [Online]. Available: <http://man7.org/linux/man-pages/man8/tc-cbs.8.html>**
-  **Wikipedia. (2019, Aug.) Credit Based Shaper Algorithm. [Online]. Available: <https://upload.wikimedia.org/wikipedia/en/d/d3/Traffic-shaping.pdf>**
-  **Vinicius Costa Gomes. (2019, Oct.) TAPRIO (Time Aware Priority Shaper) Documentation. [Online]. Available: <http://man7.org/linux/man-pages/man8/tc-taprio.8.html>**

Referenzen IV

-  Florian Fainelli. (2019, Aug.) Distributed Switch Architecture. [Online]. Available: <https://www.kernel.org/doc/Documentation/networking/dsa/dsa.txt>
-  Jiri Pirko, Scott Feldman. (2019, Aug.) Ethernet switch device driver model. [Online]. Available: <https://www.kernel.org/doc/Documentation/networking/switchdev.txt>