LINUTRONIX

This project has received funding from the Eurostars-2 joint programme with co-funding from the European Union Horizon 2020 research and innovation programme

Schlussbericht zum Vorhaben

„Software basiertes TSN (Time Sensitive Network, ein deterministisches Layer 3 Netzwerk) für Linux“

im Rahmen des Eurostars Projekts
E! 114051 OSIE
„Open Source Industrial Edge“

Heinz Egger, Kurt Kanzenbach

Linutronix GmbH
Bahnhofstrasse 3
88690 Uhldingen-Mühlhofen

Das diesem Bericht zugrunde liegende Vorhaben wurde mit Mitteln des Bundesministeriums für Bildung und Forschung gefördert. Die Verantwortung für den Inhalt dieser Veröffentlichung liegt bei der Autorin / beim Autor.

Förderkennzeichen: 01QE2022B
Projektlaufzeit: 01.05.2020 – 30.04.2023

Kurzbericht

Dieser Kurzbericht enthält die ursprüngliche Aufgabenstellung, den damals aktuellen Stand der Technik sowie den Ablauf des Vorhabens und die erzielten Ergebnisse.

Im Rahmen des „Open Source Industrial Edge“ Projekts hat sich die Linutronix GmbH mit dem Thema Software basiertes Time Sensitive Networking (TSN) beschäftigt. Es galt einen Ansatz und eine Implementierung für eine deterministische Ethernet Kommunikation auf Layer 3 und reiner Software Basis zu entwickeln.

TSN ist ein Satz von offiziellen IEEE Standards, um Ethernet echtzeitfähig zu gestalten. Das zugrunde liegende Verfahren ist Time Division Multiplex (TDM). Die Ethernet Pakete werden in unterschiedliche Traffic Klassen, wie z.B. Isochron, Management und Best Effort, unterteilt. Jede Traffic Klasse erhält einen bestimmten Zeitschlitz, indem nur diese gesendet werden darf. Wenn dieses Verfahren richtig angewendet wird, können *Garantien* im Bereich Jitter und Latenz für alle Pakete im gesamten Netzwerk aufgestellt werden. Das ist eine Eigenschaft, die herkömmliche Ethernet basierte Netzwerke nicht leisten. Der zugrunde liegende IEEE Standard ist 802.1Qbv. Die Basis für die Funktionsfähigkeit eines solchen Verfahrens stellt die Zeitsynchronisation dar. Alle Teilnehmer im Netzwerk müssen sich zur selben Zeit an die konfigurierten Zeitschlitze halten. Die Zeitsynchronisation in TSN Netzwerken erfolgt anhand dem Precision Time Protokoll (PTP). PTP funktioniert ähnlich wie NTP erlaubt allerdings eine Genauigkeit im Micro- bzw. Nanosekunden Bereich. Dieses Protokoll ist für TSN nach IEEE 802.1AS-2011 und IEEE 802.1AS-2020 standardisiert.

TSN funktioniert am besten mit Hardware Unterstützung. D.h. TSN fähige Netzwerkkarten und Bridges implementieren das TDM Verfahren direkt in der Hardware und sorgen damit für eine sehr genaue Präzision. Das Ziel von OSIE ist es auf Standard Hardware ohne TSN Unterstützung zurückzugreifen. Deswegen hat sich Linutronix zunächst mit der Ausarbeitung eines Konzeptes und der Abschätzung der erwarteten Präzision von einem rein Software basierten Ansatz beschäftigt. Die Idee ist, sich soweit wie möglich an bereits bestehenden Standards zu orientieren und die Verfahren, wenn möglich, wiederzuverwenden.

Zu dem Startzeitpunkt des OSIE Projektes im Jahre 2020 hat der Linux Kernel bereits Software Implementierungen für den Time Aware Shaper (vgl. IEEE 802.1Qbv) und die Zeitsynchronisation mittels PTP enthalten. Eine frühe Evaluierung dessen hat ergeben, dass diese Software Implementierungen insbesondere im Vergleich zu TSN fähiger Hardware *nicht* für einen produktiven Einsatz geeignet sind. Die Evaluierung wurde auf TSN fähiger Hardware von Intel und NXP durchgeführt. Als Betriebssystem kommt das offene Linux mit dem PREEMPT_RT Patch zum Einsatz. PREEMPT_RT ist eine Erweiterung für den Linux Kernel, um diesen echtzeitfähig zu modellieren und damit für den industriellen Einsatz vorzubereiten. Dieser Ansatz sorgt für deterministische Task Abarbeitungen und limitierte Worst Case Execution Times. Da die vorhandene Linux Software TSN Implementierung nicht ausreichend für echte industrielle Use Cases ist, hat Linutronix diese Implementierung angepasst und erweitert. Die Erweiterungen beinhalten hauptsächlich Modifikationen der verwendeten Interfaces in Richtung Interrupt Handling, Locking und Timer. Diese Modifikation erlauben eine echtzeitfähige Software basierte Time Aware Shaper Implementierung im Rahmen der üblichen und erwarteten Latenzen von PREEMPT_RT. Damit lassen sich Zykluszeiten bis zu 1kHz stabil erreichen. Als Hardware für die Evaluierung der Implementierung kommen STM32MP1 basierte Boards vom Projektpartner Olimex und auf der anderen Seite Standard x86 Server von Intel zum Einsatz. Linutronix hat für die Evaluierungen, Messungen von Latenz und Jitter sowie Langzeittests mehrere Testaufbauten und Demonstratoren entwickelt. Zum Teil mit Low Cost Ethernet Switches und zum anderen Teil mit direkten 1:1 Verbindungen.

Es gilt zu beachten, dass TSN nur den Layer 2 (Ethernet-Ebene) betrachtet. Die Identifikation der verschiedenen Traffic Klassen basiert auf dem Prioritätsfeld eines VLAN getaggten Ethernet Frames. Folglich gibt es acht unterschiedliche Traffic Klassen. Der Ansatz im OSIE Projekt basiert allerdings auf Layer 3 (IP-Ebene) und konzentriert sich nur auf IPv6. Die Linux 802.1Qbv Implementierung ordnet die verschiedenen Traffic Klassen und Zeitschlitze über sogenannte Socket Prioritäten zu. Sockets sind die

seit jeher verwendeten Kernel Interfaces für Netzwerkkommunikation. Als IPv6 Routing Protokoll kommt „Babel“ vom Projektpartner Nexedi zum Einsatz. Anhand der IPv6 Adressen wird der richtige Zeitslot usw. bestimmt. Resultierend, muss eine Zuordnung von IPv6 Adressen zu den Linux Kernel internen Socket Prioritäten existieren, um den TSN Ansatz von Layer 2 und Layer 3 zu übertragen. Diese Zuordnung kann mit den bestehenden Mechanismen des Linux Netzwerkstacks wie z.B. dem „Traffic Control Subsystem“ vorgenommen werden. Durch eine geschickte Analyse der IPv6 Adressen kann die richtige Socket Priorität bestimmt und gesetzt werden.

Als Kommunikationsprotokoll zwischen allen beteiligten Netzwerkpartnern kommt unter anderem OPC/UA PubSub zum Einsatz. Linutronix hat sich ebenso teilweise diesem Thema angenommen. Als Open Source Implementierung wird der offene *open62541* Stack eingesetzt. Da die gesamte TSN Kommunikation und Routing auf Layer 3 auf IPv6 Ebene angelegt ist, wurde der offene Stack um die IPv6 PubSub Fähigkeit über UDP erweitertet. Ebenso wurde ein Konzept für eine verteilte PubSub Kommunikation auf Basis von IPv6 Multicast dargelegt und getestet.

Da die Linutronix GmbH im Februar 2022 durch die Intel Deutschland GmbH übernommen wurde, ist der bis dahin gültige KMU Status nicht mehr valide. Daher wird die Nexedi GmbH Linutronix im Projekt ersetzen und die Plattform Integration des entwickelten TSN und OPC/UA Ansatzes übernehmen. Trotzdem wird Linutronix die Abschlussarbeiten an der Entwicklung (vgl. Arbeitspunkt 3) ohne weitere Förderung fertigstellen. Im Wesentlichen wurden die Arbeitspakete bis Februar 2022 wie geplant abgearbeitet. Es kam zu wenigen Verzögerungen durch Covid 19 und den resultierenden Chip Mangel.

Die Zusammenarbeit des Konsortiums konnte aufgrund der Reisebeschränkungen durch die Corona Pandemie nicht wie geplant stattfinden. Stattdessen wurden die notwendigen Abstimmungen virtuell durchgeführt. Die Zusammenarbeit hat trotz der widrigen Umstände gut funktioniert.

Zusammenfassend hat sich die Linutronix GmbH mit der Ausarbeitung, Implementierung und Evaluierung von einem rein Software basierten TSN Ansatzes für Layer 3 beschäftigt. Die Ergebnisse erlauben eine stabile und echtzeitfähige Ethernet Kommunikation und Zykluszeiten bis zu 1kHz. Ein Teil der Ergebnisse wurde in die jeweiligen Open Source Projekte beispielsweise Linux Kernel und open62541 zurück gepflegt.

LINUTRONIX

This project has received funding from the Eurostars-2 joint programme with co-funding from the European Union Horizon 2020 research and innovation programme

Schlussbericht zum Vorhaben

„Software basiertes TSN (Time Sensitive Network, ein deterministisches Layer 3 Netzwerk) für Linux“

im Rahmen des Eurostars Projekts

E! 114051 OSIE

„Open Source Industrial Edge“

Heinz Egger, Kurt Kanzenbach

Linutronix GmbH
Bahnhofstraße 3
88690 Uhldingen-Mühlhofen

Das diesem Bericht zugrunde liegende Vorhaben wurde mit Mitteln des Bundesministeriums für Bildung und Forschung gefördert. Die Verantwortung für den Inhalt dieser Veröffentlichung liegt bei der Autorin / beim Autor.

Förderkennzeichen: 01QE2022B
Projektlaufzeit: 01.05.2020 – 30.04.2023

Inhalt

Liste der Abbildungen

Liste der Tabellen

Sachbericht

Vorbemerkung

Dieser Abschlussbericht gibt einen Überblick über die im Forschungsvorhaben „Open Source Industrial Edge“ bearbeiteten Aufgaben und beschreibt die erreichten Ergebnisse. Ziel des Verbundprojektes ist es eine Referenz Edge Computing Lösung für industrielle Automatisierung und Robotik zu liefern, die *vollständig* aus freier Software und offener Hardware besteht. Das Ziel des Teilprojektes „Software basiertes TSN (Time Sensitive Network, ein deterministisches Layer 3 Netzwerk) für Linux“ ist die Spezifikation und Entwicklung des dazugehörenden deterministischen Netzwerkes basierend auf Standard Ethernet und Standard Netzwerkhardware.

Ergebnisse des Projekts

Arbeitspunkt 2: Anforderungen und Spezifikation

Time Sensitive Networking (TSN) ist ein Satz von IEEE Standards um reguläres Ethernet echtzeitfähig zu machen. TSN stellt Mechanismen für garantierte Latenzen, Jitter und Bandbreite zur Verfügung. Es werden insbesondere drei Aspekte abgedeckt: Zeitsynchronisation, Traffic Scheduling und das Netzwerk Management.

TSN ist eine Transport Technologie, welche auf Layer 2 (Ethernet-Ebene) arbeitet. Die Daten, welche durch das Netzwerk übertragen werden, sind applikationsspezifisch. Industrielle Protokolle beinhalten u.a. OPC/UA, PROFINET, Audio/Video und viele mehr. Alle Transportgarantien hinsichtlich Latenz und Jitter im Netzwerk werden im Speziellen durch die Netzwerkteilnehmer (die Endgeräte) selbst, als auch die Bridges gewährleistet. In einem TSN Netzwerk haben die Bridges Wissen über die *Traffic Flows* der Anwendungen und deren Anforderungen hinsichtlich Echtzeit und Ressourcen im gesamten Netzwerk. Dadurch kann Determinismus erreicht werden. Dahingehend unterscheidet sich Standard Ethernet von TSN.

Im Besonderen werden alle Echtzeitgarantien durch verschiedene Traffic Scheduling Algorithmen realisiert. Hervorzuheben ist der Time Aware Shaper aus dem IEEE 802.1Qbv Standard. Alle Ethernet Frames werden in verschiedene Traffic Klassen wie z.B. *Isochron*, *Management* oder *Best Effort* unterteilt. Die Zuteilung funktioniert anhand des Prioritätsfelds (PCP) eines VLAN getaggten Frames. Das Traffic Scheduling unterteilt die Netzwerkressourcen in Zeitschlitze. Jeder Zeitschlitz ist für eine bestimmte Zeit lang geöffnet. Innerhalb dieser Slots dürfen nur die konfigurierten Traffic Klassen versendet werden. Dadurch wird das Aufstellen von Garantien möglich. Das zugrundeliegende Verfahren ist Time Division Multiplex (TDM). Das Scheduling im gesamten Netzwerk basiert auf einer *gemeinsamen Zeitbasis*. Die Synchronisation zwischen allen beteiligten Geräten, inklusive der Bridges, wird durch das Precision Time Protokoll (PTP) erledigt. PTP ist für TSN nach 802.1AS-2011 bzw. 802.1AS-2020 spezifiziert. Das generelle Prinzip wird in Abbildung 1 verdeutlicht. Es wird beispielhaft ein Traffic Schedule dargestellt. Es gibt drei Traffic-Klassen (Isochronous, Management, Best Effort), die mit unterschiedlichen Farben dargestellt werden. Jede Klasse erhält einen eigenes Zeitschlitz und kann nur in diesem versendet werden. Nach einer Millisekunde beginnt der Schedule von vorne.

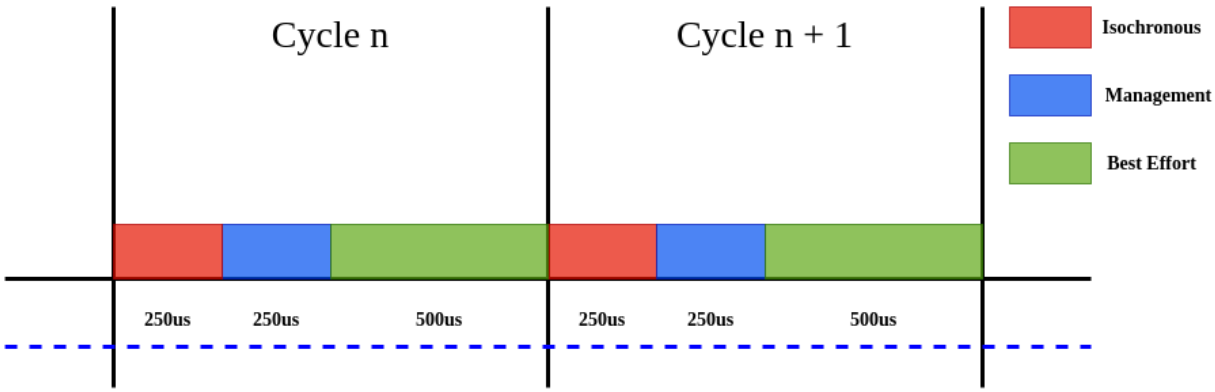

Abbildung 1: Beispiel Traffic Schedule

TSN Netzwerke funktionieren am Besten mit Hardware Unterstützung; d.h. TSN fähige Netzwerkkarten und Bridges implementieren das TDM Verfahren und zeitgesteuertes Versenden von Netzwerkpaketen direkt in der Hardware für eine sehr genaue Präzision. Das Ziel von OSIE ist es, auf Standard Hardware ohne TSN Unterstützung zurückzugreifen. Deswegen hat sich Linutronix im Rahmen der Anforderungsanalyse zunächst mit der Ausarbeitung eines Konzeptes und der Abschätzung der erwarteten Präzision von einem rein Software- basierten Ansatz beschäftigt. Die grundlegende Idee ist es, sich soweit wie möglich an bereits bestehenden Standards zu orientieren und die Verfahren, wenn möglich, wiederzuverwenden.

Im ersten Teil des OSIE Projektes hat Linutronix zusammen mit dem Projektpartner Nexedi einen Report der zu erwarteten Performance eines rein Software-basierten TSNs ausgearbeitet. Dazu kommt SlapOS als Betriebssystem zum Einsatz. Die Basis von *SlapOS* ist Linux als Open Source Betriebssystem in Kombination mit dem PREEMPT_RT Patch. PREEMPT_RT ist eine Entwicklung von u.a. Linutronix und sorgt dafür, dass der Linux Kernel echtzeitfähig wird. Als Hardware wurden x86 Server Maschinen und ARM basierte Boards von dem Projektpartner Olimex verwendet. Erste Messungen in Bezug auf Scheduling, Latenz, Zeitsynchronisierung mittels PTP, sowie Versand und Empfang von Netzwerkpaketen und GPIO Toggling wurden durchgeführt. Ausgehend von den erzielten Ergebnissen (siehe P-OSIE-Report.Software.Defined.Time.Sensitive.Networking-002-en.pdf) wurden die Anforderungen und Use Cases abgeglichen. Ausgehend von diesen Ergebnissen hat Linutronix sich entschieden den Software- basierten TSN Ansatz weiter zu verfolgen.

Arbeitspunkt 3: Entwicklung der Hauptkomponenten für die Plattform

Zum Startzeitpunkt des OSIE Projektes im Jahre 2020 hat der Linux Kernel bereits Software-implementierungen für den Time Aware Shaper (vgl. IEEE 802.1Qbv) und die Zeitsynchronisation mittels PTP enthalten. Die Implementierungen im Linux Kernel basieren auf sogenannten *Queing Disciplines* (Qdiscs). Eine Qdisc ist nichts anderes als ein Paket Scheduler als Teil vom Linux Netzwerkstack. Der Paket Scheduler entscheidet, wann ein Paket gesendet wird und wann nicht. Die verschiedenen TSN Scheduling Algorithmen lassen sich passend als Qdiscs implementieren. Das Prinzip ist in Abbildung 2 zu sehen. Alle Applikationen verwenden das normale Linux Socket Interface zur Netzwerkkommunikation. Anhand der Qdiscs wird entschieden, wann welches Paket von welcher Applikation gesendet wird. Folglich kann der Time Aware Shaper direkt im Linux Kernel als Software Scheduler implementiert werden. Es werden die Konfiguration des Ablaufplans und eine sehr genaue Zeitsynchronisation benötigt.

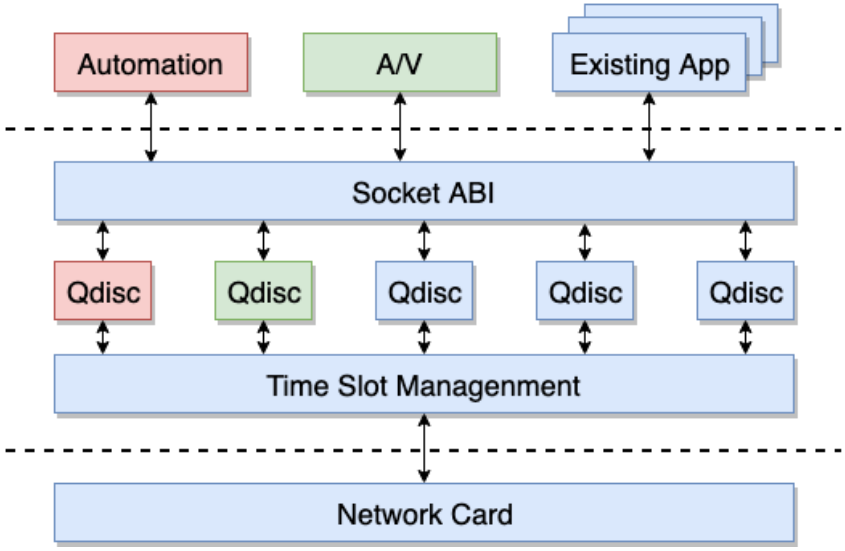

Abbildung 2: Linux Netzwerkstack

Vergleich Hardware und Software TSN

Im ersten Schritt der Entwicklung wurden die bereits bestehenden Linux Software Implementierungen für den Time Aware Shaper und PTP ausführlich im Vergleich zu korrespondierenden Hardware Implementierungen getestet. Ebenso wurden alle weiteren Latenzen im System ermittelt und gemessen. Das Ergebnis soll aufzeigen, ob und inwiefern die bereits bestehende Linux Implementierung für industrielle Anwendungen eingesetzt werden kann oder ob Optimierungen notwendig sind.

Für die Evaluierung wurde TSN-fähige Hardware ausgesucht. Zunächst wurden zwei verschiedene Boards verwendet: Ein SBC-LS1028A von der Firma Microsys [1] und ein Industrie PC von der Firma Congatec [2] ausgestattet mit Intel TSN Netzwerkkarten [3]. Die Hardware Spezifikation und Linux Kernel Version zum Zeitpunkt der Evaluierung ist in Tabelle 1 zu sehen.

	Industrie PC	Embedded Gerät
Device	PC	mpxls1028
CPU	4 x Intel i5	2x Cortex A72
RAM	4 GiB	4 GiB
NIC	Intel i210/i225	NXP enetc
NIC Features	IEEE 802.1AS, 802.1Qbv	IEEE 802.1AS, 802.1Qbv
Kernel	V5.11-rt	V5.11-rt

Tabelle 1: Test Geräte

Beide Geräte wurden direkt über einen einfachen Link miteinander verbunden. Die Hardware ist in der Lage sowohl das PTP Timestamping als auch den Time Aware Shaper direkt in den Netzwerkkarten auszuführen. Der Testaufbau ist in Abbildung 3 dargestellt. Anhand von diesem gezeigten Testaufbau wurden exakt vier Messungen durchgeführt und Größenordnungen von Software vs. Hardware Lösungen ermittelt:

1. Applikationsscheduling Latenz von PREEMPT_RT
2. Genauigkeit der Zeitsynchronisation
3. Latenz von dem Time Aware Shaper
4. Latenz von dem Versand bzw. Empfang von Netzwerkpaketen

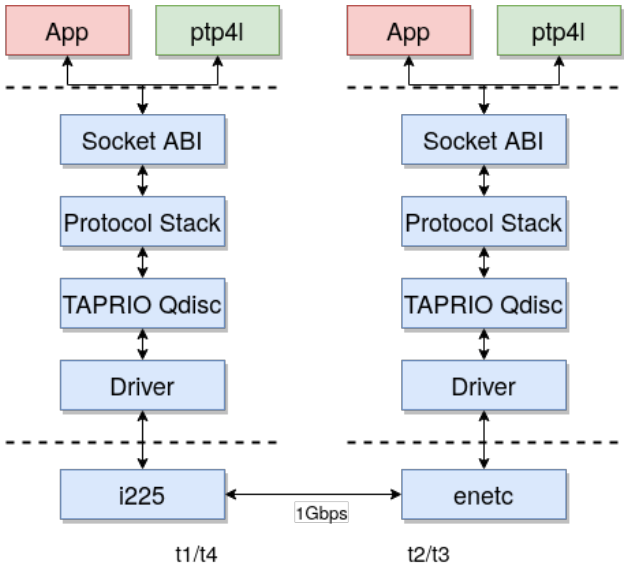

Abbildung 3: Hardware TSN Testaufbau

Um das Echtzeitverhalten und damit die Reaktionsfähigkeit des Linux Kernels mit PREEMPT_RT zu bestimmen, wurden die *rt-tests* [4] eingesetzt. Insbesondere ist das Werkzeug *cyclictest* verwendet worden. Cyclictest stellt periodische Timer auf vorbestimmte Zeitpunkte und überprüft, wann die Applikation tatsächlich auf diese Timer Events reagiert. Die Differenz zwischen dem eigentlich Aufwachzeitpunkt und dem tatsächlichen Zeitpunkt wird kontinuierlich aufgezeichnet. Die Werte lassen einen Rückschluss auf die Echtzeitfähigkeit und Reaktionsfreudigkeit des PREEMPT_RT Systems zu. Die Tests wurden auf beiden Hardware Plattformen für 24 Stunden ausgeführt. Parallel dazu wurden die Systeme unter Volllast gesetzt, um die Worst Case Werte zu erhalten. Die Ergebnisse sind in Abbildung 4 dargestellt.

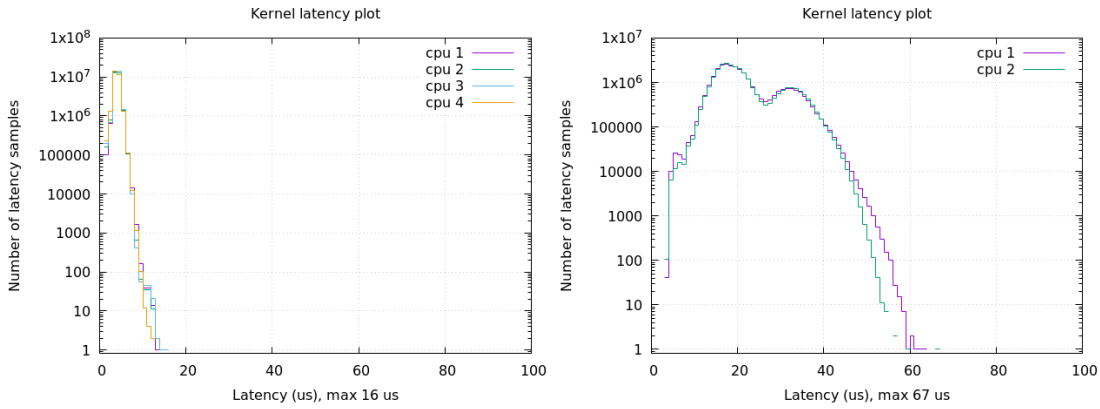

Abbildung 4: Industrie PC und Layerscape cyclictest Plots

Die maximale Antwortzeit für den Industrie PC liegt bei 16 µs und beim Layerscape bei 67 µs. Die Werte liegen für die eingesetzten CPUs im erwarteten Bereich. Ohne PREEMPT_RT wären die maximalen Werte deutlich höher (>= 500 µs).

Beide Testmaschinen sind über das PTP Protokoll miteinander synchronisiert. Als Software kommt der offene Stack *linuxptp* [5] zum Einsatz. Der Linux Kernel stellt für PTP lediglich den Zugriff auf die PTP Hardware Clock (PHC) und ein Socket Interface zum Ermitteln der Zeitstempel der PTP Pakete zur Verfügung. Das Protokoll an sich ist komplett im User Space als Anwendung (*ptp4l*) implementiert. Es gilt zu beachten, dass ptp4l lediglich die PHC kontrolliert. Die PHC ist i.d.R. Teil der Netzwerkkarte. Diese Zeitbasis ist zunächst unabhängig von der Linux Systemzeit. Für ein funktionierendes Applikations- und Paketscheduling auf Basis der Netzwerkzeit bedarf es folglich einer weiteren Synchronisation. Das

betrifft die Synchronisation der PHC zur Linux Systemzeit. Dazu bringt linuxptp ein zusätzliches Tool namens *phc2sys* mit. Dieses wurde parallel zu ptp4l eingesetzt.

Das PTP Protokoll basiert intern auf Berechnungen, wie lange Nachrichten von einem Netzwerkteilnehmer zu anderen Netzwerkteilnehmern im Netzwerk benötigen. Dazu werden PTP Nachrichten mit Zeitstempeln versehen. Diese können von TSN fähiger Hardware direkt beim Versand bzw. Empfang von Paketen generiert werden. Das kann im PHY oder in der MAC geschehen. Falls das nicht der Fall ist, kann der Linux Kernel diese Zeitstempel ebenso in Software generieren, z.B. direkt im zuständigen Linux NIC Treiber. Je nachdem welches Verfahren genutzt wird, ist die erreichbare Synchronisation eine andere. Die gemessenen Ergebnisse sind in Tabelle 2 dargestellt.

Methode	Minimum	Maximum
Hardware PHY	10 ns	50 ns
Hardware MAC	10 ns	100 ns
Software	10 µs	100 µs

Tabelle 2: Zeitsynchronisation Präzision

Die Größenordnung ist klar ersichtlich. Mit einer Software Implementierung liegt die Genauigkeit im Bereich von Mikrosekunden, mit Hardware Unterstützung im Nanosekundenbereich. Es gilt folgendes zu beachten: Auch nicht TSN-fähige industrielle Netzwerkhardware hat i.d.R. Unterstützung für PTP auf Hardwareebene. Folglich ist eine valide Annahme für die Implementierung von Software basiertem TSN, dass die Zeitsynchronisation sehr genau ist.

Für eine Umsetzung von einem TSN Netzwerk ist u.a. die Präzision des Time Aware Shapers ausschlaggebend. Dieser kann von TSN Netzwerkkarten und Switches direkt in Hardware implementiert werden. Ebenso existiert eine Linux Software Implementierung in Form der *TAPRIO* Qdisc [6].

Die ausgesuchte Hardware für die Evaluierung kann einen vergebenen Traffic Schedule direkt in der Hardware ausführen. Der Traffic wird in verschiedene Klassen eingeteilt und entsprechend in unterschiedliche Versandschlangen verschoben. Die Netzwerkkarte selbst überwacht die PTP Zeit und öffnet bzw. schließt die Traffic Klassen anhand der Konfiguration. In Linux kann die TAPRIO Qdisc Konfiguration anhand des *tc* Werkzeugs aus dem *iproute2* Paket erfolgen. Für die Messungen werden drei Traffic Klassen angelegt: Management, Isochron und Best Effort. Der Management Zeitschlitz beinhaltet die PTP Nachrichten. Der Isochron Slot beinhaltet die Echtzeit Pakete, welche zur Messung herangezogen werden. Jeglicher andere Traffic im Netzwerk wie z.B. HTTP oder SSH Datenverkehr wird in Best Effort versendet. Die Aufteilung ist folgendermaßen:

- 500 µs Best Effort
- 250 µs Management
- 250 µs Isochron

Wobei die Zykluszeit insgesamt eine Millisekunde beträgt.

Zur Messung der 802.1Qbv Latenz, sowohl in Hardware als auch in Software, wird eine zyklische Echtzeit Applikation verwendet. Diese generiert Pakete in einem Zyklus von ebenfalls einer Millisekunde. Die Applikation wird so geplant, dass der Isochron Zeitslot nicht offen ist, wenn die Pakete versendet werden. D.h. die Pakete müssen gepuffert und beim Öffnen der Isochron Traffic Klasse sofort versendet werden. Das Tool welches zum Einsatz kommt, sind die TSN Scripts [7] von Vladimir Oltean. Diese wurden u.a. für die Qbv Latenz Messungen entwickelt. Da die eingesetzte Hardware nicht nur PTP Nachrichten, sondern genauso alle anderen Ethernet Pakete zeitstempeln kann, wird diese Fähigkeit für die Messungen eingesetzt. Auf einer der beiden Maschinen wird der Versendezeitstempel jedes Echtzeit Pakets genommen und auf der anderen Seite der Empfangszeitstempel. Da beide Geräte

über PTP synchronisiert sind, können die Zeitstempel korreliert werden. Ebenso kann die Qbv Latenz abgeleitet werden:

$$expectedtime = (cycle * period) + basetime + pathdelay$$

Die erwartete Ankunftszeit eines jeden Pakets ist durch den Einsatz des Time Aware Shapers vorberechenbar. Die obige Formel zeigt dies. Die Basiszeit ist der Startzeitpunkt des konfigurierten Ablaufplans. Cycle gibt den aktuellen Zyklus an. Die Periode ist eine Millisekunde. Das Pathdelay ist die reine Übertragungszeit und konstant. Aus der Differenz zwischen der erwarteten Ankunfts- und tatsächlichen Ankunftszeit lässt sich die Qbv Latenz herleiten. Die Ergebnisse sind in Tabelle 3 zu sehen.

Methode	Minimum [µs]	Maximum [µs]
Hardware	1	5
Software	23	unbegrenzt

Tabelle 3: Latenz TAPRIO Implementierungen

Die Testläufe wurden über mehrere Stunden ausgeführt. Parallel wurde auf dem Netzwerk Last mit *iperf3* erzeugt, um die Worst Case Latenz zu bestimmen. Die Worst Case Latenz im Fall der Software Implementierung ist nach oben offen („unbounded"*). Folglich, ist die Software Implementierung nicht echtzeitfähig*. Es bedarf einer genauen Analyse, warum die Implementierung nicht ordnungsgemäß funktioniert. Die Ausführungen dazu sind im nächsten Kapitel beschrieben.

Neben der 802.1Qbv Latenz ist ebenso die Applikationslatenz entscheidend. Der Time Aware Shaper stellt sicher, dass die Netzwerkressourcen für gegeben Traffic Klassen im gesamten Netzwerk zur Verfügung stehen. Für Applikationen hingegen ist ebenso die Zeit der Paketverarbeitung im Linux Kernel und User Space von Bedeutung. Um diese zu messen hat Linutronix ein eigenes User Space Tool entwickelt. Dieses fungiert als eine Art „cyclictest" über das Netzwerk. Die *Talker* Anwendung sendet periodisch Ethernet Frames an einen *Listener*. Der Talker inkludiert den gewünschten Versandzeitstempel in das jeweilige Paket. Der Listener nimmt ebenfalls einen Zeitstempel, sobald das Paket im User Space angekommen ist. Die Differenz der beiden Zeitstempel ergibt die komplette Latenz von User Space zu User-Space-Applikation. Die Applikationen können auf Versandseite ein TSN-Hardware-Feature namens „Lauch Time" einsetzen. Das erlaubt der Applikation einen vorher berechneten Sendezeitstempel an die Hardware zu übergeben. Die Netzwerkkarte wird das Paket exakt zu diesem Zeitpunkt versenden. Ebenso implementieren die Applikationen UDP (Layer 4) und Ethernet (Layer 2) als Transport. Zusätzlich wird optional auf Empfangsseite der eXpress Data Path (XDP) [8] eingesetzt um die Latenz zu optimieren. Tabelle 4 zeigt die Latenz mit dem TSN-Hardware-Feature „Tx Launch Time".

	UDP	UDP XDP	Ethernet	Ethernet XDP
Minimum [µs]	9.299	4.821	8.207	4.599
Maximum [µs]	54.545	53.892	53.752	53.090
Average [µs]	9.796	8.331	8.634	8.248

Tabelle 4: Applikations- Ende zu Ende Latenz mit Tx Launch Time

Die gleiche Messung ohne die Verwendung von TSN Hardware Unterstützung ist in Tabelle 5 dargestellt.

	UDP	UDP XDP	Ethernet	Ethernet XDP
Minimum [µs]	29.959	28.350	21.775	21.239
Maximum [µs]	107.817	115.355	100.423	102.707
Average [µs]	31.034	29.376	22.573	22.049

Tabelle 5: Applikations- Ende zu Ende Latenz ohne Tx Launch Time

Diese Testläufe wurden mehrere Stunden auf den Testplattformen ausgeführt. Es ist zu erkennen, dass das „Tx Launch Time“-Feature die Latenz deutlich verbessert. Ebenso ist hervorzuheben, dass XDP die Latenz nicht spürbar verbessert. Andere Messungen, insbesondere auf langsamen CPUs wie z.B. ARM32-basierten haben hingegen gezeigt, dass die Verwendung von XDP im Kontrast zu klassischen RAW Sockets die Verarbeitungszeit um bis zu 50% reduziert. Die Testmaschinen in dem hier gezeigten Setup sind allerdings leistungsstark und damit ist die CPU-Zeitersparnis gering.

Software TSN Testumgebungen

Die durchgeführten Tests aus Schritt 1 haben gezeigt, dass die Zeitsynchronisation als Voraussetzung für Software-definiertes TSN gegeben ist. Ebenso können Linux Echtzeitapplikationen hinsichtlich deterministischer Ethernet Kommunikation entwickelt werden. Auf der anderen Seite wurde deutlich, dass insbesondere die Software-TDM-Implementierung des Linux Kernels nicht für einen produktiven Einsatz geeignet ist. Folglich muss diese Implementierung erweitert werden. Dazu bedarf es neben der Softwareentwicklung geeignete Testumgebungen.

Im Laufe des Projekts hat der Partner Olimex die Embedded Devices auf Basis des STM32MP1 von STMicroelectronics [9] fertiggestellt. Linutronix hat für diese Hardware Plattformen ein komplettes Board Support Package („BSP“) entwickelt. Dieses basiert auf Debian Stable („Bullseye“) als Distribution inklusive dem Linux Kernel in Version 5.15-LTS mit PREEMPT_RT Patch. Als Buildsystem kommt E.L.B.E. [10] von Linutronix zum Einsatz. Des Weiteren ist eine Update Funktionalität basierend auf dem Industriestandard „swupdate“ [11] vorgesehen.

Ausgehend von dieser Hardware Plattform hat Linutronix zwei Testumgebungen aufgebaut. Eine, bei der ein Industrie PC direkt mit den Olimex Boards verbunden ist und eine, wo einen Ethernet Switch von HP (Enterprise Procurve 1810G-8) verbaut ist. Dieser Switch hat keine TSN Funktionalitäten oder PTP Unterstützung integriert und stellt somit einen typischen Switch für Enterprise Netzwerke dar. Die Testaufbauten sind in Abbildung 5 abgebildet.

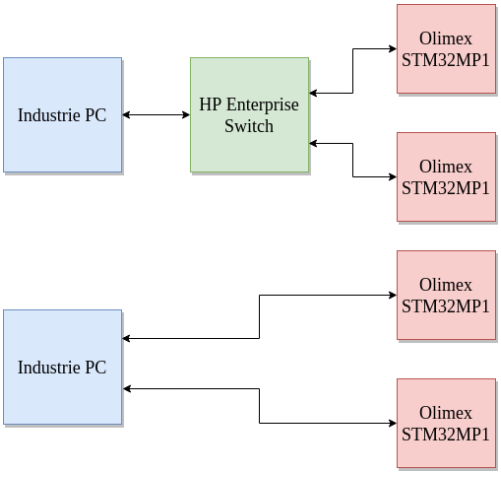

Abbildung 5: Software TSN Testaufbauten

Diese Testumgebungen dienen dazu die Software-basierte TSN Implementierung zu testen und bei Versionsupdates, z.B. des Linux Kernels, Regressiontests durchzuführen. Als Testsoftware kommen die eigens entwickelten Tools aus Schritt 1 zum Einsatz. Diese Tools wurden ebenso erweitert, dass nicht nur die Latenz in eine Richtung bestimmt werden kann, sondern ebenfalls die *Round-Trip-Zeiten* berechnet werden können. Das hat folgenden Hintergrund: Wenn der Ablaufplan im Netzwerk ordnungsgemäß funktioniert, kann für jeden Ethernet Frame die Ankunftszeit im nächsten Zyklus vorberechnet werden. D.h. anhand der Round-Trip-Zeiten ist sehr einfach abzulesen, ob das TDM Verfahren funktioniert oder nicht.

Beim Aufsetzen der STM32MP1 Boards haben sich mehrere Probleme gezeigt. Zunächst wurde ein RT Test (vgl. vorheriges Kapitel) durchgeführt. Die Worst Case Latenz lag bei 209 µs, was für eine ARM32 CPU zu hoch ist. Die Erwartung liegt bei maximal 150 µs. Die CPU-Takt Frequenz wurde im Bootloader zu gering eingestellt. Anschließend lagen die Worst Case Reaktionszeiten bei 139 µs. Das Histogramm ist in Abbildung 6 dargestellt.

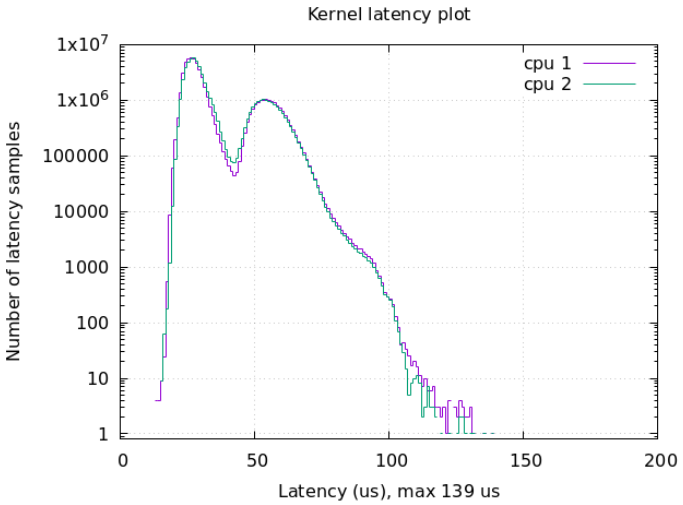

Abbildung 6: STM32MP1 cyclictest Plot

Die Ausführung der Netzwerk-Latenz-Evaluierung-Tools hat ebenfalls diverse Problem mit dem Netzwerk Treiber der Ethernet Schnittstelle aufgezeigt. Der STM32MP1 verwendet intern einen *stmmac*. Dieser hat keine TSN Hardware Unterstützung integriert. Allerdings wird PTP in Hardware implementiert. Diese Implementierung hat nicht funktioniert und wurde von Linutronix repariert. Die Fehlerbehebungen dazu sind in den Linux Kernel zurückgeflossen. Weiterhin gab es mit diesem Treiber Probleme in Richtung *Flow Steering* und der XDP Implementierung. Alle bekannt gewordenen Punkte sind zusammen mit den Linux Maintainern und anderen Firmen gelöst worden. Auszug der behobenen Probleme:

- Aktivierung der PTP Clock im Olimex U-Boot Bootloader [1]
- Korrekte Berechnung der PTP Zeitstempel im stmmac für ARM32 [2] [3]
- Korrektes EtherType Steering für PTP Pakete im stmmac [4]
- Korrekte Behandlung von TCP SYN Paketen im Time Aware Shaper (TAPRIO) [5]
- Korrekte Behandlung von Segmentierung im Time Aware Shaper (TAPRIO) [6]

1 https://github.com/OLIMEX/u-boot-olinuxino/commit/3e98e6fde3d324610e1625f60277ecf5a9ec9e4a

2 https://git.kernel.org/pub/scm/linux/kernel/git/netdev/net.git/commit/?id=3751c3d34cd5

3 https://git.kernel.org/pub/scm/linux/kernel/git/netdev/net-next.git/commit/?id=c6d5f1933085

4 https://git.kernel.org/pub/scm/linux/kernel/git/netdev/net-next.git/commit/?id=e48cb313fde3

5 https://git.kernel.org/pub/scm/linux/kernel/git/netdev/net.git/commit/?id=e8a64bbaaad1

6 https://git.kernel.org/pub/scm/linux/kernel/git/netdev/net-next.git/commit/?id=497cc00224cf

- Korrekte Implementierung von XDP im Intel i210 Treiber in Kombination mit PTP [7]

Schlussendlich konnten die beiden Testumgebungen erfolgreich mit der Hardware von Olimex aufgebaut und für Regression Tests eingesetzt werden.

Software TSN Implementierung

Der nächste Schritt ist die Analyse, warum die Time Aware Shaper Software Implementierung (TAPRIO) des Linux Kernels nicht funktioniert. Für diese Analyse wurde das Linux Kernel interne Tracing Framework *ftrace* herangezogen. Dieses ermöglicht die Aufnahmen von relevanten Ereignissen im Linux Kernel ohne die Laufzeit stark zu beeinflussen. Die Auswertung der Trace Daten hat ergeben, dass die Soft-Interrupt-Verwendung im Linux Netzwerk Stack nicht optimal für PREEMPT_RT Systeme ist.

Die Software Implementierung von TAPRIO verwendet intern *High Resolution Timer*, um die Umschaltung zwischen einzelnen Einträgen des konfigurierten Ablaufplans vorzunehmen. Problematisch ist, dass dieser Timer vom *ksoftirqd* Prozess abgearbeitet wird. Der Timer wiederum weckt den Netzwerkstack auf, um ggf. gepufferte Pakete zu versenden. Da der Netzwerkstack selbst ebenso Soft Interrupts (vgl. *NET_TX* und *NET_RX*) verwendet, werden diese im Falle von TAPRIO auch über den *ksoftirqd* abgearbeitet. Unter normalen Umständen ist dies kein Problem. Für Echtzeitsysteme ist diese Konstellation allerdings nicht in Ordnung. Das liegt daran, dass der *ksoftirqd* keine Echtzeitpriorität gesetzt hat und damit mit allen anderen Prozessen im System um die CPU-Zeit *konkurriert*. Folglich ist TAPRIO nicht echtzeitfähig. Der *ksoftirqd* läuft aus gutem Grund mit SCHED_OTHER, um z.B. Überlast (Denial of Service) Situationen zu handhaben. Die Hoch Priorisierung dieses Deamons ist demnach keine valide Option.

Die von Linutronix entwickelte Lösung sieht vor:

1. Die verwendeten High-Resolution-Timer im harten Interrupt Kontext zu behandeln. Das ist möglich, da diese Timer nur die Zeitschlitz Umschaltung des Ablaufplans und Triggern des Netzwerkstacks übernehmen. Dadurch, dass die geschützten atomaren Regionen sehr klein und die Zeiten im harten Interrupt Kontext somit kurz sind, ist dieser Ansatz vertretbar.

2. Das Aufwecken des Netzwerkstacks in einem dedizierten Thread anstatt im Transmit-Soft-Interrupt zu erledigen. Dadurch kann dieser TAPRIO Thread eine Echtzeitpriorität gemäß der Systemkonfiguration erhalten, ohne alle anderen Soft Interrupts zu beeinträchtigen.

Durch die beiden vorgestellten Modifikation werden im Versandpfad alle nicht echtzeitfähigen Komponenten umgangen. Anhand der vorgeschlagenen Implementierung wird Qbv Latenz im Worst Case beschränkt. Resultierend ist diese Lösung im industriellen Umfeld einsetzbar.

Im weiteren Verlauf hat sich gezeigt, dass dieses Problem mit den Soft Interrupts nicht nur in Versandrichtung, sondern ebenso in Empfangsrichtung auftreten kann. Beim Empfang eines Netzwerkpakets wird die Übergabe an den Netzwerkstack ebenfalls im Soft-Interrupt-Kontexts abgearbeitet. Unter PREEMPT_RT wird der Soft Interrupt im Normalfall am Ende und im Kontext des zugehörigen Interrupt Threads ausgeführt. Es kann allerdings Situationen geben, wo dies nicht der Fall ist. Beispielsweise wenn der *ksoftird* schon auf der CPU aktiv ist, wird der Empfang der Netzwerkpakete in diesen Daemon ausgelagert, was wiederum die Echtzeit erheblich beeinflusst. Nach langer Recherche hat Linutronix eine Lösung des Problems gefunden, ohne Code Modifikationen vornehmen zu müssen. Seit Linux Kernel Version 5.12 gibt es eine generische Optimierung im Netzwerkstack namens *Threaded NAPI*. Das erlaubt die Bearbeitung der empfangenen Pakete in dedizierten Threads anstatt in Soft Interrupts. Dieser Mechanismus wurde ursprünglich entwickelt, um den Empfang von Paketen auf mehrere CPUs zu verteilen und damit den Durchsatz zu erhöhen. Aktive Benutzer im Kernel sind u.a. WiFi Treiber. Werden diese dedizierten Threads allerdings mit Echtzeitprioritäten eingeplant, kann dieser Mechanismus ebenso für den deterministischen Empfang von Ethernet

7 https://lkml.kernel.org/netdev/20210503072800.79936-1-kurt@linutronix.de/

Paketen eingesetzt werden. Ausführliche Tests haben ergeben, dass dieser Mechanismus den Determinismus in Empfangsrichtung gewährleistet.

Werden die TAPRIO Erweiterungen mit dem Threaded NAPI kombiniert, wird die Linux Software TSN Implementierung für industrielle Use Cases nutzbar. Die Genauigkeit liegt im Bereich der gemessenen *cyclictest* Latenzen und ist damit wie ursprünglich erwartet. Standard Industrie Szenarien mit Zykluszeiten bis zu 1kHz (1ms) sind umsetzbar. Bei Linutronix stehen zu diesem Zeitpunkt allerdings noch Langzeittests aus. Ebenso müssen die Einflüsse von den herkömmlichen Ethernet Switches weiter betrachtet werden. PTP Tests haben gezeigt, dass sich die Präzision der Zeitsynchronisation im Worst Case von Nanosekunden auf einstellige bzw. sehr geringe zweistellige Mikrosekunden verringert. Das liegt am Jitter im Forwarding Delay der Switches. Je konstanter die Verzögerung, desto besser für PTP als auch die Echtzeitkommunikation im Allgemeinen. Da das OSIE Projekt Datacenter-taugliche Switche einsetzen wird, können diese ggf. besser z.B. anhand 802.1Q Priorisierung auf die Use Cases abgestimmt werden. Nichtsdestotrotz müssen die Switches bei Planung der Netzwerkzyklen berücksichtigt werden.

Der OSIE Projektplan sieht vor, dass die Software-TSN-Implementierung genauso auf Mikrocontrollern z.B. von Olimex eingesetzt werden soll. Linutronix hat hierzu erste Tests auf einem STM32-basierten Nucleo Board mit integrierter Netzwerkschnittstelle vorgenommen. Dadurch, dass die Kommunikation im OSIE Projekt über OPC/UA PubSub abgehandelt wird, wurde der offene Stack *open62541* eingesetzt. Als Echtzeit-Betriebssystem hat sich Linutronix für das Open Source Projekt Zephyr [12] entschieden. Die OPC/UA Kommunikation auf dem Mikrocontroller ist bis dato funktional. Die Portierung des TDM Ansatzes inklusive Zeitsynchronisation ist zum jetzigen Zeitpunkt des Projekts nicht abgeschlossen.

Im Rahmen der externen Mikrocontroller Anbindung, hat Linutronix ebenfalls den M4 Mikrocontroller der STM32MP1 Boards von Olimex in Betrieb genommen. Dieser SoC hat neben den beiden ARM32 CPUs einen M4 Mikrocontroller für Echtzeitaufgaben direkt im Chip integriert. Hierzu wurde ebenfalls Zephyr als RTOS eingesetzt. Das Linux *RemoteProc* Framework erlaubt die Anbindung und Kontrolle dieses Mikrocontrollers über ein standardisiertes Interface. Der Hintergrund ist folgender: Da diese Embedded Devices für Echtzeit Aufgaben wie z.B. Steuerung von Robotern oder Motoren eingesetzt werden sollen, sind diese Co-Prozessoren für sehr spezielle harte Echtzeitanforderungen hervorragend geeignet. Insbesondere dann, wenn die Reaktionsfähigkeit vom Linux mit PREEMPT_RT nicht ausreichend ist. Der Datenaustausch zwischen den Linux Applikationen und dem RTOS auf dem M4 Core laufen über OpenAMP [13]. OpenAMP ist ein offenes Framework zum Life Cycle Management des Mikrokontrollers und zum Datenaustausch. Eine Demo mit einem Sensor, welcher am Mikrokontroller angebunden ist, hat die Validität des Ansatzes gezeigt und dargelegt. Das Prinzip ist in Abbildung 7 dargestellt.

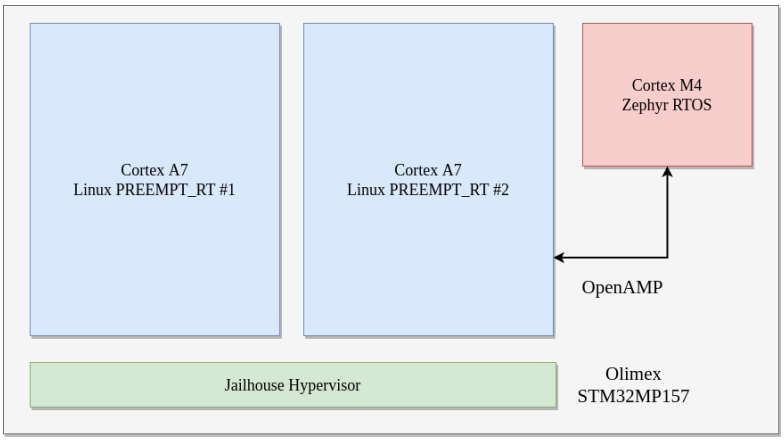

Abbildung 7: Olimex STM32MP1 Mixed Criticality

Der Cortex M4 ist mittels OpenAMP an das Linux PREEMPT_RT System angebunden. Ebenso ist dieser Ansatz mit Echtzeit Hypervisorn wie z.B. Jailhouse [14] als Open Source Lösung kombinierbar. Dazu hat Linutronix eine entsprechende Konfiguration ausgearbeitet.

Weiterhin sieht der Projektplan vor, den TDM Ansatz des Software-definierten TSNs von Layer 2 auf Layer 3 zu übertragen. Dazu kommt ein IPv6 basiertes Routing Protokoll namens „Babel“ von dem Projektpartner Nexedi zum Einsatz. Die IPv6 Route bzw. Adresse soll festlegen, welcher zugehörige Zeitschlitz im Netzwerk verwendet werden soll. Die TAPRIO Implementierung des Linux Kernels verwendet intern Socket Prioritäten für diese Zuordnung. Resultierend muss es eine Übersetzung von IPv6 Adressen zu den Socket Prioritäten geben. Der Ansatz von Linutronix sieht vor dieses Mapping transparent durch geschickte Konfiguration des Netzwerkstacks vorzunehmen. Linux hat ein Subsystem namens *Traffic Control*, welches diverse Paketverarbeitung erlaubt. Insbesondere die *skbedit* Aktion ermöglicht die Vergabe bzw. Modifikationen von Socket Prioritäten für einzelne Pakete. Anhand von *tc filter* können Pakete auf IPv6 Source bzw. Destination Adressen zugeordnet werden. D.h. die Kombination von *tc filter* mit der *skbedit* Aktion kann die Zuordnung von IPv6 Adressen auf Socket Prioritäten vollständig transparent übernehmen. Dieser Ansatz ist vorgesehen, wurde allerdings noch nicht getestet.

OPC/UA PubSub

Als Middleware im OSIE Projekt kommt u.a. OPC/UA PubSub zum Einsatz. OPC/UA ist ein Industriestandard zum Datenaustausch über Plattformen und Hersteller hinweg. Informationsmodelle beschreiben die verfügbaren Funktionen und Daten von einzelnen Geräten wie beispielsweise Sensoren und Aktoren. Insbesondere der Publish und Subscribe Mechanismus sieht echtzeitfähige Kommunikation entweder direkt über Layer 2 (Ethernet-Ebene) oder über Layer 4 (Transport-Ebene) via UDP/IP vor. Als Implementierung des OPC/UA Standards kommt der offene und freie Stack *open62541* zum Einsatz.

Linutronix hat diesen Stack in den vorgestellten Testumgebungen im Einsatz. Es hat sich gezeigt, dass die PubSub Implementierung über UDP / IPv6 nicht funktioniert. Da das OSIE Projekt allerdings nur IPv6 einsetzt, musste dieses Problem gelöst werden. Dazu hat Linutronix die IPv6 Implementierung hinsichtlich PubSub über UDP in open62541 repariert und alle notwendigen Änderungen zurück in den offiziellen Stack integriert [15].

Des Weiteren soll es Netzwerkteilnehmer geben, welche als IPv6 Router fungieren. Die PubSub Kommunikation muss weiterhin gewährleistet sein. Die OPC/UA-PubSub-Kommunikation basiert intern auf Multicast, was erlaubt Daten an mehrere Geräte gleichzeitig zu versenden. Die Switche im Netzwerk erkennen Multicast Kommunikation (vgl. IGMP Protokoll) und leiten Multicast Pakete nur an Ports weiter, dessen angeschlossene Geräte an den jeweiligen Nachrichten interessiert sind. Bei Layer 3 Routern gibt es die Möglichkeit zwischen verschiedenen IPv6 Netzen Multicast Routing zu betreiben. Da Nexedi’s Babel Ansatz auf diesem Prinzip basiert, hat Linutronix evaluiert, ob Linux Maschinen als IPv6 Multicast Router eingesetzt werden können. Die Ergebnisse haben gezeigt, dass dies möglich ist. Die technische Herausforderung besteht in der Konfiguration der Routen. Linutronix hat zur Evaluierung statische Routen verwendet. Nexedi hat diesen Ansatz aufgegriffen und ihr eigenes Routing Protokoll um die dynamische Konfiguration erweitert.

Arbeitspunkt 4: Plattform Integration

Arbeitspunkt 4 beinhaltet die Umsetzung der Erkenntnisse und Ergebnisse aus dem vorherigen Arbeitspunkt in einem Pilotprojekt (vgl. Schweißroboter).

Dadurch dass die Linutronix GmbH im Februar 2022 von der Intel Deutschland GmbH übernommen wurde, ist der bis dahin bestehende KMU Status nicht mehr gültig. Deswegen wird der Arbeitspunkt 4 von der Nexedi GmbH für Linutronix übernommen. Das OSIE Projekt läuft weiter bis Mitte 2023.

Die Ergebnisse des Arbeitspunkts 4 werden folglich beim Abschlussbericht des neuen Projektpartners Nexedi GmbH vorliegen.

Vergleich zur Vorhabenbeschreibung

Durch die Corona Pandemie konnten physikalische Abstimmungen und Besuche der Projektpartner nicht wie geplant stattfinden. Außerdem hat der dadurch einhergehende Chip Mangel für Verzögerungen im Projektablauf gesorgt. Dennoch wurden alle notwendigen Meetings virtuell durchgeführt und die Zusammenarbeit des Konsortiums hat insgesamt gut funktioniert.

Der Arbeitspunkt 2 wurde wie geplant am Anfang des Projekts abgearbeitet. Ebenso wurde die Implementierung des Software basierten Time Sensitive Networking wie geplant bis Februar 2022 durchgeführt. Zu diesem Zeitpunkt wurde die Linutronix GmbH von der Intel Deutschland GmbH übernommen. Das führte zum Verlust des KMU Status. Arbeitspunkt 3 ist noch nicht abgeschlossen. Linutronix wird diesen Arbeitspunkt ohne Förderung im Rahmen des OSIE Projekts beenden. Der folgende Arbeitspunkt 4 wird vom Nachfolger, der Nexedi GmbH, übernommen.

Verwertbarkeit des Ergebnisses

Die erreichten Ergebnisse weisen nach, dass Software based TSN für bestimmte industrielle Anwendungsbereiche geeignet sind. Davon lassen sich Verwertungsmöglichkeiten im Bereich Consulting, Integration und Support ableiten. Da das Projekt noch nicht abgeschlossen ist, und Linutronix durch den Verlust des KMU Status WP4 an die Nexedi GmbH übergeben hat, ist der Aufwand der zur vollständigen Verwertbarkeit führt, momentan nur schwer abzuschätzen.

Veröffentlichungen

An folgenden Veranstaltungen wurde von Linutronix aus über die Fortschritte und Ergebnisse aus dem OSIE Projekt berichtet:

- Kurt Kanzenbach, Vortrag „Software Time Sensitive Networking with Linux“ TSN/A Konferenz, 29. September 2021, virtuell

Literaturverzeichnis

[1] MicroSys, „System on Module based on NXP QorIQ LS1028A,“ 2022. [Online]. Available: https://microsys.de/fileadmin//user_upload/Images/News/miriac_MPX-LS1028A-DSC0274_small.jpg. [Zugriff am 25 05 2021].

[2] Congatec, „conga-TS170,“ 2022. [Online]. Available: https://www.congatec.com/de/produkte/com-express-type-6/conga-ts170/. [Zugriff am 25 05 2021].

[3] Intel, „Intel Ethernet Controller I225-V,“ 2022. [Online]. Available: https://ark.intel.com/content/www/de/de/ark/products/184676/intel-ethernet-controller-i225-v.html. [Zugriff am 25 05 2021].

[4] J. Kacur und C. Williams, „Suite of real-time tests,“ 2022. [Online]. Available: https://git.kernel.org/pub/scm/utils/rt-tests/rt-tests.git. [Zugriff am 25 05 2021].

[5] R. Cochran, "The Linux PTP Project," 2022. [Online]. Available: http://linuxptp.sourceforge.net/. [Accessed 26 12 2020].

[6] V. C. Gomes, "TAPRIO (Time Aware Priority Shaper) Documentation," 2022. [Online]. Available: http://man7.org/linux/man-pages/man8/tc-taprio.8.html. [Accessed 25 05 2021].

[7] V. Oltean, „TSN Test Scripts/Tools,“ 2022. [Online]. Available: https://github.com/vladimiroltean/tsn-scripts. [Zugriff am 25 05 2021].

[8] "eXpress Data Path," 2022. [Online]. Available:

https://www.kernel.org/doc/html/latest/networking/af_xdp.html. [Accessed 25 05 2021].

[9] Olimex, „Olimex Blog," 2022. [Online]. Available: https://olimex.wordpress.com/tag/stm32mp1/.

[10] Linutronix, „Debian based E.mbedded L.inux B.uild E.nvironment," 2022. [Online]. Available: https://elbe-rfs.org/.

[11] S. Babic, „SWUpdate: software update for embedded system," [Online]. Available: https://sbabic.github.io/swupdate/swupdate.html. [Zugriff am 09 07 2022].

[12] Z. Project, „Zephyr Project," [Online]. Available: https://zephyrproject.org/. [Zugriff am 09 07 2022].

[13] Linaro, „Open Asymmetric Multi-Processing," [Online]. Available: https://www.openampproject.org/. [Zugriff am 09 07 2022].

[14] Siemens, „Jailhouse," [Online]. Available: https://github.com/siemens/jailhouse. [Zugriff am 09 07 2022].

[15] K. Kanzenbach, „Open62541 PubSub IPv6," 15 06 2021. [Online]. Available: https://github.com/open62541/open62541/pull/4473. [Zugriff am 09 07 2022].