Open-Source Implementierung von OPC UA PubSub für echtzeitfähige Kommunikation mit Time-Sensitive Networking

Julius Pfrommer, Thomas Usländer

Informationsmanagement und Leittechnik (ILT)
Fraunhofer IOSB
Fraunhoferstraße 1, 76131 Karlsruhe
julius.pfrommer@iosb.fraunhofer.de
thomas.uslaender@iosb.fraunhofer.de

Zusammenfassung. OPC UA hat sich in den letzten Jahren als Protokoll für industrielle nicht-echtzeitkritische Kommunikation durchgesetzt. Mit OPC UA PubSub wird nun OPC UA um das Publish / Subscribe Kommunikationsmuster erweitert. Es wird gezeigt, wie OPC UA PubSub in Kombination mit Time-Sensitive Networking (TSN) auch für echtzeitkritische Kommunikation genutzt werden kann. Der Ausgangspunkt ist die Erkenntnis, dass zur Einhaltung der Zeitschranken für Echtzeitkommunikation der Publisher in einem (von der Netwerkhardware getriggerten) Interrupt ausgeführt werden muss. Daraus folgern Anforderungen an die Synchronisation des PubSub Publishers mit dem nichtechtzeitkritischen OPC UA Server, dessen Informationsmodell auch die Grundlage für die im Publisher erzeugten Nachrichten ist.

1 Einführung

OPC UA ist weit verbreitet für flexible Kommunikation in der Automatisierungstechnik. Für viele Anwendungsfälle hat OPC UA seinen Vorgänger, das Microsoft DCOM-basierte OPC Classic, abgelöst. Die mit OPC UA verbundene Hoffnung ist die Durchsetzung einer einheitlichen Technologie für standardisierte und sichere Kommunikation über Herstellergrenzen und Anwendungsdomänen hinweg. Der derzeitige Grad der Durchsetzung von OPC UA in der Praxis unterstützt diesen Anspruch. Der neue Part 14 der OPC UA Spezifikation definiert eine Erweiterung von OPC UA basierend auf dem Publish / Subscribe Kommunikationsmuster [EFGK03]. Dies ermöglicht neue Einsatzszenarien, inklusive dem Nachrichtenaustausch zwischen vielen Teilnehmern (many-to-many). Zusätzlich ist die Verbindung von OPC UA PubSub mit Time-Senstive Networking (TSN) für echtzeitfähige Kommunikation vorgesehen. Es existiert jedoch derzeit eine Lücke zwischen den kommerziell erhältlichen Lösungen von OPC UA PubSub und dem Anspruch Echtzeitanforderungen mit hoher Flexibiliät zu verbinden. Diese Arbeit schließt diese Lücke durch die Darstellung einer Implementierung von OPC UA PubSub, die zur Laufzeit flexibel konfiguriert werden kann und dennoch Echtzeitanforderungen erfüllt.

J. Jasperneite, V. Lohweg (Hrsg.), *Kommunikation und Bildverarbeitung in der Automation*,
Technologien für die intelligente Automation 12, https://doi.org/10.1007/978-3-662-59895-5_6

Dieser Artikel ist eine Erweiterung von [PERK18]. Er ist wie folgt strukturiert: Dieser Abschnitt setzt sich mit einer Zusammenfassung von OPC UA und Time-Sensitive Networking (TSN) fort. Der Abschnitt 2 zeigt Wege auf wie nicht-echtzeitkritische OPC UA Server mit echtzeitfähigen OPC UA PubSub Publishern integriert werden können. Der Ansatz wurde auf Basis des open62541 SDKs implementiert. Die Details der Implementierung werden in Abschnitt 3 besprochen. Auf dieser Grundlage wurden Messreihen durchgeführt, die in Abschnitt 4 zur Evaluierung des Ansatzes verwendet werden. Der Abschnitt 5 beschließt den Artikel mit einer Zusammenfassung und einem Ausblick.

1.1 OPC Unified Architecture

OPC UA ist ein Client-Server Protokoll für industrielle Kommunikation basierend auf TCP/IP. Das Protokoll wurde in der IEC 62541 Reihe standardisiert. Ein OPC UA Server bietet Zugriff auf Daten und Funktionalitäten die in einem objekt-orientierten semantischen Informationsmodell repräsentiert werden. Clients interagieren mit dem Informationsmodell über eine Menge standardisierter Services. OPC UA folgt strikt dem Request / Response Kommunikationsmuster. Jeder Service definiert eine *Request*- und eine *Response*-Nachricht. Allerdings ist OPC UA ein asynchrones Protokoll bei dem Response-Nachrichten verzögert und in anderer Reihenfolge gesendet werden dürfen. Über diesen Mechanismus ist ein *Subscription*-Mechanismus für die Push-Benachrichtigung von Clients über Datenänderungen und Events definiert.

Wie bereits der Vorgänger OPC Classic hat sich OPC UA als wichtigstes Protokoll für herstellerübergreifende, nicht-echtzeitkritische industrielle Kommunikation durchgesetzt. Die OPC Foundation treibt die kontinuierliche Verbesserung des Standards voran und unterstützt die Entwicklung von Companion Spezifikationen, welche Informationsmodelle für die verschiedenen Anwendungsdomänen definieren. Weiterhin führt die OPC Foundation Zertifikationen von standardkonformen Implementierungen durch um Interoperabilität zu gewährleisten.

1.2 Time-Sensitive Networking (TSN)

Im Rahmen der IEEE 802.1 Standardisierungs-Serie wurden Ethernet-Erweiterungen für Echtzeitkommunikation zunächst unter dem Namen Audio Video Bridging (AVB) und in jüngeren Jahren als Time-Sensitive Networking (TSN) entwickelt. Einige der Erweiterungen aus der TSN-Reihe wurden bereits verabschiedet. Zum Beispiel die Synchronisierung der Uhren der Netzwerketeilnehmer mit IEEE 802.1AS [SJ07] und die Reservierung von Übertragungskapazität über Zeitschlitze in IEEE 802.1Qbv. Der IEEE 802.1Qbv Standard definiert Warteschlagen für Nachrichten in verschiedenen Klassen von Netzwerkverkehr. Jede Warteschlange wird von einer Übertragungs-Schranke (Transmission Gate) gesteuert. Wenn ein Paket nicht in dem zugeordneten Zeitschlitz gesendet werden kann, so muss auf den nächsten Zyklus gewartet werden. Für die Einhaltung von Echtzeit-Garantien die ein solches Verschieben der Übertragung nicht zulassen muss die Nachricht mit einem Offset vor der Öffnung des Zeitschlitzes vorbereitet

Tabelle 1. TSN Erweiterungen von Ethernet in der IEEE 802.1 Standardserie.

Standard	Titel
IEEE 802.1Qav	Forwarding and Queuing Enhancements for Time-Sensitive Streams
IEEE 802.1AS-Rev	Timing and Synchronization for Time-Sensitive Applications
IEEE 802.1Qbu	Frame preemption
IEEE 802.1Qbv	Enhancements for Scheduled Traffic
IEEE 802.1Qca	Path Control and Reservation
IEEE 802.1Qcc	Stream Reservation Protocol (SRP) Enhancements and Performance Improvements
IEEE 802.1Qci	Per-Stream Filtering and Policing
IEEE 802.1Qcr	Bridges and Bridged Networks Amendment: Asynchronous Traffic Shaping
IEEE 802.1CB	Frame Replication & Elimination for Reliability

und in der Warteschlange eingelastet werden. Tabelle 1 gibt einen Überblick über die weiteren TSN-Erweiterungen.

Die Autoren von [Ind18] beschreiben Traffic-Arten für typische Anwendungsfälle in der Automatisierung und ihre Beziehungen zu den TSN Standards. Das allgemeine Potential und Beschränkungen von Ethernet-basierten Echtzeitprotokollen wird in in [JSW07] beschrieben. Der Vorteil von TSN gegenüber klassischen Feldbussen liegt in der Herstellerneutralität, dem Wegfallen von Lizenzgebühren, höhere Datenraten [Sha18], der Möglichkeit der flexiblen Konfiguration echtzeitfähiger Netzwerkverbindungen über Bridges zwischen Teilnetzen hinweg [GAS+17] und die Skaleneffekte einer Technologie, welche auch im Consumer-Bereich breiten Einsatz finden soll.

1.3 OPC UA PubSub und OPC UA PubSub in Kombination mit TSN

Der neue Part 14 der OPC UA Spezifikation [OPC18] erweitert OPC UA um PubSub als zusätzliches Kommunikationsparadigma. Mehrere Subscriber können ein Thema abonnieren. Veröffentliche Nachrichten werden an alle Subscriber des Themas weitergeleitet. Da potentiell viele Subscriber eine Nachricht empfangen können kehrt OPC UA PubSub in gewisser Weise zur Definition von Telegramm-Nachrichten mit definiertem Aufbau zurück, ähnlich wie diese in klassischen Feldbussen verwendet werden. Allerdings kann der Nachrichtenaufbau zur Laufzeit flexibel konfiguriert werden. Der Nachrichtenaufbau wird in sogenannten *PublishedDataSets* definiert. Diese beinhalten eine Liste von Variablen und Event-Quellen aus dem Informationsmodell des OPC UA Servers für die gemeinsame Übertragung in einer PubSub Nachricht. Das PublishedDataSet ist ebenfalls im Informationsmodell repräsentiert und kann mit einem OPC UA Client ausgelesen werden um den Aufbau und die Semantik der PubSub Nachrichten zu erhalten.

Zum einen definiert der Part 14 die Verwendung existierender Protokolle, MQTT und AMQP, welche für das PubSub Kommunikationsparadigma ausgelegt sind. Diese Protokolle verwenden für die Verteilung von Nachrichten einen zentralen Broker. Sie sind heute weit verbreitet und können über das Internet verwendet werden. Zum anderen definiert der Standard ein eigenes UDP-basiertes Protokoll, UADP genannt, welches den Multicast-Mechanismus des IP-Standard für die Verteilung von Nachrichten an viele Empfänger verwendet. Die Subscriber registrieren sich in einer Multicast-Gruppe, die durch eine IP-Addresse repräsentiert ist. Pakete an diese Addresse werden an alle registrierten Teilnehmer der Multicast-Gruppe weiter geleitet. Der Aufwand der Nachrichten-Verteilung wird damit an die Netzwerkinfrastruktur ausgelagert. Das reduziert die Komplexität der Implementierung des Publishers. Allerdings ist ide Verfügbarkeit von Multicast auf lokale Netzwerke beschränkt. Im Internet wird dieser Teil des IP-Standards üblicherweise nicht unterstützt. Zuletzt definiert der Standard die Übertragung von PubSub über Ethernet-Frames auf Schicht 2 des OSI-Modells. Auf der Ebene von Ethernet ist die Einbindung von TSN als Ethernet-Erweiterung direkt möglich. Die OPC Foundation hat den Ethernet EtherType 0xB62C für TSN-basierten OPC UA PubSub Netzwerkverkehr registriert.[1]

Der Ursprung einer OPC UA PubSub Nachricht im Informationsmodell eines OPC UA Servers ist in dem tatsächlich versendeten Netzwerk-Paket nicht mehr erkennbar. Dies ermöglicht die Umsetzung sehr leichtgewichtiger OPC UA PubSub Implementierungen, welche einen festen Nachrichtenaufbau einkompiliert haben und die dynamische Zusammenstellung der Nachrichteninhalte aus der Konfiguration des PublishedDataSet übergehen. Eine solche Implementierung benötigt im Grunde gar keinen klassischen OPC UA Server im Hintergrund. Dadurch gehen aber die wichtigen Eigenschaften der Flexibilität und der Zugriff auf die Semantik der Nachrichteninhalte verloren.

2 Integration von OPC UA Servern mit echtzeitfähigem OPC UA PubSub

Es ist anzunehmen, dass ein TCP/IP basierter OPC UA Server keine harten Echtzeitgarantien einhalten kann. Langlaufende Funktionsaufrufe, aufwändige Verschlüsselung von Nachrichten und viele andere Operationen in einem OPC UA Server sorgen für potentielle Verzögerungen. Nun soll aber das Informationsmodell des OPC UA Servers als die Quelle von OPC UA PubSub Nachrichten verwendet werden. Auf einem System mit einem Prozessorkern kann der Kontrollfluss über einen (Hardware-getriggerten) Interrupt unterbrochen werden um die rechtzeitige Ausführung des PubSub Publishers sicher zu stellen.

Nebenläufige Programme mit mehreren Threads können Semaphoren verwenden um Race-Conditions durch den parallelen Zugriff auf geteilte Speicherbereiche zu unterbinden. Bei der Verwendung von Interrupts auf Systemen mit nur einem Prozessorkern ist der Ansatz aber nicht möglich. Den ein wartender Interrupt

[1] Siehe den Eintrag unter http://standards-oui.ieee.org/ethertype/eth.txt.

kann nicht durch das öffnen einer Semaphore aus einem parallelen Prozess „befreit“ werden. Stattdessen müssen alle vom Publisher verwendeten Methoden *reentrant* sein. Dies ist jedoch eine starke Einschränkung. Selbst die in POSIX definierte Schnittstelle zu `malloc` für die Verwendung von Speicher auf dem Heap ist nicht reeantrant [But97]. Noch wichtiger jedoch ist der Zugriff auf das OPC UA Informationsmodell welches zwischen dem OPC UA Server und dem Publisher geteilt wird. Dieser Zugriff muss ebenfalls reentrant sein.

Die erste Möglichkeit ist die Verwendung bekannter Speicheraddressen für Shared Memory und das Setzen individueller skalarer Werte mit atomischen Operationen.[2] Atomische Operationen sind notwendig um sicher zu stellen, das ein Interrupt ein Update des Informationsmodells nicht unterbrechen kann. Dies würde sonst zu einem inkonsistenten Informationsmodell führen. Allerdings können nur kleine Skalare (Integer, Gleitkommazahlen, usw.) mit atomischen Operationen gesetzt werden. Der Ansatz ist in der Abbildung 1a dargestellt.

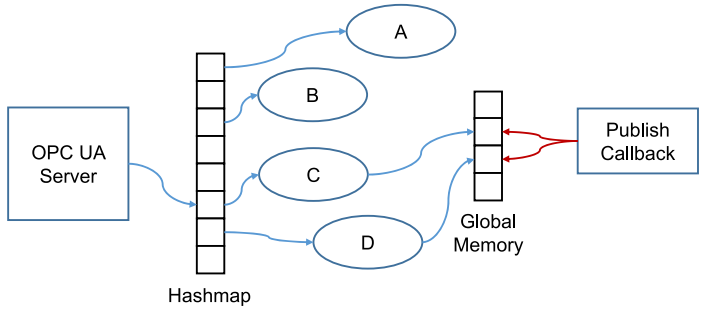
Die zweite Möglichkeit ist die Verwendung von Copy-on-Replace für Änderungen im Informationsmodell: Ein OPC UA Informationsmodell ist ein Graph von Knoten mit typisierten Referenzen als Kanten zwischen den Knoten. In open62541 sind alle Knoten über ihren Identifier aus einer Hashmap abrufbar. Einmal in der Hashmap referenziert ist ein Knoten unveränderlich. Er kann nur durch einen neuen Knoten, bzw. eine modifizierte Kopie seiner selbst, ersetzt werden. Das Update wird durch den Austausch eines Pointers mit einer atomischen Compare-and-Switch (CAS) Operation ausgeführt. Dadurch ist das Informationsmodell stets konsistent, auch wenn der Server während einer Schreib-Operation durch ein Interrupt unterbrochen wird. Der Ansatz ist in Abbildung 1b dargestellt. Der Vorteil des zweiten Ansatzes gegenüber dem Ersten ist, dass jede Art Wert (auch Strings) über PubSub veröffentlicht werden kann. Weiterhin kann die Konfiguration des Publishers im OPC UA Informationsmodell abgespeichert in jedem Publisher-Zyklus ausgelesen werden ohne Echtzeitgarantien zu verletzen.

3 Implementierung

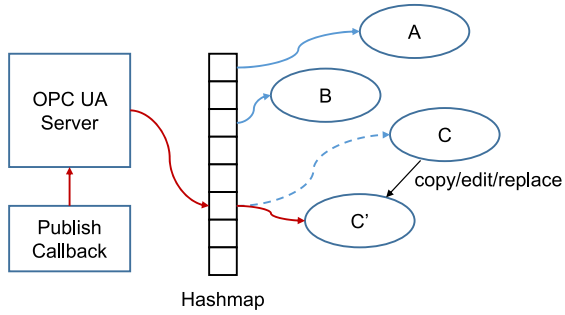
Das Fraunhofer IOSB hat gemeinsam mit Kalycito Infotech eine Implementierung von OPC UA PubSub über TSN umgesetzt und evaluiert. Die Open Source in Automation Development Labs (OSADL) eG hat den organisatorischen Rahmen für die Finanzierung dieser Aktivität durch ein Konsortium von Unternehmen aus der Automatisierungsindustrie bereitgestellt. Neben der Umsetzung von OPC UA PubSub realisierte das Projekt einen Demonstrator (siehe Abbildung 2) und Trainings für die unterstützenden Unternehmen.

Die Implementierung basiert auf dem open62541 SDK [PGP+15]. Das SDK ist in C99 geschrieben und unter der MPLv2 Lizenz veröffentlicht. Ein Abstraktions-Layer hält die Kern-Bibliothek frei von Platform-spezifischen Schnittstellen (auch POSIX). Das vereinfacht die Portierung auf neue Hardware-Architekturen und

[2] Im open62541 SDK ist dies über sogenannte *DataSources* umgesetzt. Eine DataSource ersetzt die normalen Lese- und Schreibzugriffe auf einen Knoten im Informationsmodell mit nutzerdefinierten Callbacks.



(a) Shared Memory bei dem der den Austausch kleiner Speicherbereiche (einzelne Integer) mit atomischen Operationen unterstützt wird.



(b) Copy-on-replace für unveränderliche Knoten im Informationsmodell. Der Knoten C wird erst zu C' kopiert, dann editier und zuletzt mit einer atomischen Operation mit C ausgetauscht.

Abb. 1. Möglichkeiten für die lock-freie Synchronisation zwischen einem OPC UA Server und einem OPC UA PubSub Publisher der in einem Interrupt ausgeführt wird. Die roten Pfeile zeigen Zugriffe aus dem Interrupt an die „reentrant" sein müssen.

Betriebssysteme. OPC UA Server auf der Basis des open62541 SDK benötigen, bei der Aktivierung eines minimalen Feature-Set, weniger als 100kB an ROM und RAM. Das Fraunhofer IOSB hat das open62541 SDK gemeinsam mit weiteren Partnern initiiert und begleitet die Weiterentwicklung des SDK gemeinsam mit Partnern aus der Forschung und der Industrie.

Die Integration von OPC UA PubSub in das Server-SDK basiert auf der Verwendung von drei verschiedenen Schnittstellen. Zunächst wird das bestehende Encoding und Decoding für das binäre OPC UA Protokoll für die PubSub-Nachrichten weiter verwendet. Zweitens werden die Laufzeitdaten für die PubSub Nachrichten aus dem Informationsmodell des OPC UA Servers ausgelesen. Drittens wird die Konfiguration des PubSub Publishers im Informationsmodell repräsentiert und können dort zur Laufzeit modifiziert werden.

Wie in Abschnitt 2 beschrieben wird der OPC UA PubSub Publisher zyklisch in einem Interrupt ausgeführt. Dabei werden die folgenden Schitte ausgeführt:

Abb. 2. OPC UA PubSub über TSN Demonstrator auf dem Stand der OPC Foundation auf der Hannover Messe 2018. Die folgenden Hardware-Plattformen wurden für den Demonstrator integriert: Intel Atom/i210 von TQ-Systems, Dual Chip Lösung mit ARM Cortex M4 + TSN Switch von Analog Devices, FPGA-TSN Cyclone V SoC von Intel PSG/Altera, Xilinx FPGA-TSN Zync SoC. Die Komponenten des Demonstrators verfügen über 2-Port Switches und sind zu einer Daisy Chain verbunden. Die verwendeten Betriebssysteme sind Linux mit der RT-Preempt Erweiterung [CB13] und FreeRTOS.

1. Austausch des Standard-`malloc` mit einer einfachen Puffer-basierten Implementierung für die Dauer des Interrupts
2. Erzeugen und Befüllen der Struktur für die PubSub Nachricht auf Basis der definierten PubishedDataSets
3. Kodierung der PubSub Nachricht in einem Speicherpuffer
4. Absenden der PubSub Nachricht (Einlasten des Speicherpuffers in der Warteschlange der TSN-fähigen Netzwerk-Hardware)
5. Wiederherstellung der Standard `malloc`-Implementierung

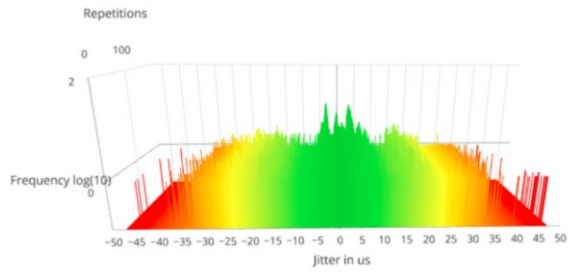
Neben dem binären UADP Protokoll für PubSub unterstützt open62541 mittlerweile auch JSON-enkodierung der PubSub-Nachrichten und den Transport über MQTT. Die Implementierung ist unter `https://github.com/open62541/open62541` frei unter der MPLv2 Lizenz zugänglich.

4 Evaluierung

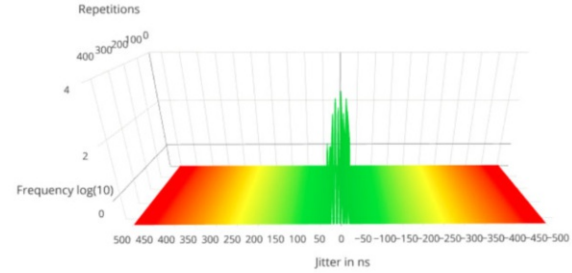
Die Messungen für die Evaluierung wurden auf zwei identischen PC mit Intel Core i5-6402P Prozessoren mit 2,80GHz Taktfrequenz durchgeführt. Der Netzwerkadapter basiert auf dem Intel i210 Chipsatz und ist über PCIe integriert.

Die Netzwerkadapter der beiden PCs sind mit einem Ethernet-Kabel Peer-to-Peer verbunden. Das verwendete Betriebssystem ist Linux 4.16.8-rt3 mit der RT-Preempt Erweiterung. Die Systemuhr der beiden PCs ist über die Netzwerkadapter auf der Basis von IEEE 802.1AS-Rev synchronisiert. Ein Patch-Set von Intel wurde für die Umsetzung von IEEE 802.1Qbv Funktionalität verwendet. Neben der Anbindung an die i210 Netzwerkhardware wird die SO_TXTIME Option für Netzwerk-Sockets, sowie Time-Based Scheduling (TBS) für RT-Linux bereit gestellt.

Die OPC UA PubSub Nachrichten werden mit einem Taktrate von 10kHz (alle 100µs) versendet. Die gesendete Nachricht basiert auf einem PublishedDataSet mit einem einzigen skalaren Zahlenwert. Die Konfiguration des PublishedDataSet wird in jedem Zyklus ausgelesen und auf dieser Basis die Inhalte für die Nachricht aus dem Informationsmodell ausgelesen und zusammen gestellt. Der Offset zwischen dem Beginn der Nachrichten-Erzeugung und dem Beginn des Übertragungs-Fensters ist auf 5µs eingestellt. Bei einer Taktfrequenz des Prozessors von 2,8GHz bleiben also 14.000 Rechenzyklen um die Nachricht fertig zu stellen und sie in eine Warteschlange in der Netzwerkhardware einzulasten.



(a) Beobachteter Jitter mit deaktiviertem TBS.



(b) Beobachteter Jitter mit aktiviertem TBS.

Abb. 3. Jitter der OPC UA PubSub Netzwerknachrichten.

Die Abbildung 3 zeigt den beobachteten Jitter von OPC UA PubSub mit und ohne Unterstützung von TBS. Aus der Abbildung 3a geht hervor, dass ohne TBS große Schwankungen im Zeitverhalten im mehrstelligen Mikrosekunden-Bereich

auftreten. Im Gegenzug dazu zeigt Abbildung 3b ein hochgradig deterministisches Verhalten mit aktiviertem TBS. Der beobachtete Jitter ist im Nanosekunden-Bereich. Da die Nachichten mit einem Offset vor dem Absenden eingelastet werden, rührt dieser Jitter also nur noch von der Netzwerkhardware her.

5 Zusammenfassung und Ausblick

Der Artikel hat sich mit den Time-Sensitive Networking (TSN) Erweiterungen für den Ethernet Standard befasst und wie diese für den echtzeitfähigen Transport von OPC UA PubSub eingesetzt werden können. Es wurde ein Ansatz vorgestellt, bei dem der OPC UA PubSub Publisher von einem Hardware-Interrupt angesteuert wird um Verzögerung und Jitter zu minimieren. Die Schwierigkeit liegt darin den echtzeitfähigen PubSub Publisher mit dem normalen OPC UA Server zu synchronisieren, da beide auf ein gemeinsames OPC UA Informationsmodell zugreifen. Die beschriebenen Ansätze wurden für das open62541 OPC UA SDK umgesetzt. Messungen wurden auf einem Testsystem mit Linux RT-Preempt Patches durchgeführt. Dabei wurden Wiederholungsraten bis zu 10kHz mit minimalem Jitter erreicht.

Zukünftige Aufgaben bezüglich der Verwendung von TSN für echtzeitfähiges OPC UA PubSub beinhalten die Portierung auf weitere TSN Hardware und Embedded-Plattformen, langfristige Latenz-Tests in einer QA-Farm, sowie die Umsetzung einer „Distribution“ für OPC UA PubSub über TSN, bestehend aus Hardware, Treibern, echtzeitfähigem Betriebssystem und dem OPC UA PubSub SDK. Diese Komponenten müssen aufeinander abgestimmt und getestet sein um auf der Basis bekannter Ende-zu-Ende Eigenschaften Lösungen entwickeln zu können. Die Entwicklung weiterer Features ist geplant. Mit dem Blick auf nicht-echtzeitkritische Anwendungen von OPC UA PubSub steht noch die Erweiterung um JSON als Format für Netzwerknachrichten, sowie Broker-basierte Nachrichtenübertragung auf der Basis von MQTT und AMQP aus. Ein weiteres Thema ist die Verschlüsselung von OPC UA PubSub Nachrichten mit symmetrischen Gruppenschlüsseln.

Literatur

[But97] David R Butenhof. *Programming with POSIX threads*. Addison-Wesley Professional, 1997.

[CB13] Felipe Cerqueira und Björn Brandenburg. A comparison of scheduling latency in linux, preempt-rt, and litmus rt. In *9th Annual Workshop on Operating Systems Platforms for Embedded Real-Time Applications*, Seiten 19–29. SYSGO AG, 2013.

[EFGK03] Patrick Th Eugster, Pascal A Felber, Rachid Guerraoui und Anne-Marie Kermarrec. The many faces of publish/subscribe. *ACM computing surveys (CSUR)*, 35(2):114–131, 2003.

[GAS+17] Marina Gutiérrez, Astrit Ademaj, Wilfried Steiner, Radu Dobrin und Sasikumar Punnekkat. Self-configuration of IEEE 802.1 TSN networks. In *Emerging Technologies and Factory Automation (ETFA), 2017 22nd IEEE International Conference on*, Seiten 1–8. IEEE, 2017.

[Ind18] Industrial Internet Consortium. Time Sensitive Networks for Flexible Manufacturing Testbed - Description of Converged Traffic Types. Bericht, 2018.

[JSW07] Juergen Jasperneite, Markus Schumacher und Karl Weber. Limits of increasing the performance of industrial ethernet protocols. In *Emerging Technologies and Factory Automation, 2007. ETFA. IEEE Conference on*, Seiten 17–24. IEEE, 2007.

[OPC18] OPC Foundation. OPC Unified Architecture Specification, Part 14: PubSub, 2018.

[PERK18] Julius Pfrommer, Andreas Ebner, Siddharth Ravikumar und Bhagath Karunakaran. Open source OPC UA PubSub over TSN for realtime industrial communication. In *2018 IEEE 23rd International Conference on Emerging Technologies and Factory Automation (ETFA)*, Jgg. 1, Seiten 1087–1090. IEEE, 2018.

[PGP+15] Florian Palm, Sten Grüner, Julius Pfrommer, Markus Graube und Leon Urbas. Open source as enabler for OPC UA in industrial automation. In *Emerging Technologies & Factory Automation (ETFA), 2015 IEEE 20th Conference on*, Seiten 1–6. IEEE, 2015.

[Sha18] Shaper Group. OPC UA TSN: A new Solution for Industrial Communication. Bericht, 2018.

[SJ07] Sebastian Schriegel und Jürgen Jasperneite. Investigation of industrial environmental influences on clock sources and their effect on the synchronization accuracy of IEEE 1588. In *Precision Clock Synchronization for Measurement, Control and Communication, 2007. ISPCS 2007. IEEE International Symposium on*, Seiten 50–55. IEEE, 2007.