Real-time Open Source Solution for Industrial Communication Using OPC UA PubSub over TSN

Gopiga S K[1], Keerthivasan A S[2], Nikhil Vannan K[3], Selva Suba Jenifer J[4], Shriya Chaurasia[5], Suriya Narayanan P V[6], Thangavaila K T[7]
Kalycito Infotech Private Limited, Coimbatore, India.
[1]gopiga.sk@kalycito.com, [2]keerthivasan.as@kalycito.com, [3]nikhilvannan.k@kalycito.com, [4] selvasuba.jenifer@kalycito.com, [5]shriya.c@kalycito.com,
[6]suriyanarayanan.pv@kalycito.com, [7]thangavaila.k@kalycito.com

***Abstract*— The world is moving in a direction where any future Industry 4.0 device shall support Open Platform Communication Unified Architecture (OPC UA) - a platform independent, service-oriented modeling framework to achieve interoperability through a vendor neutral standard. There is general alignment among different industry players to leverage OPC UA along with emerging time aware Ethernet standard Time Sensitive Networking (TSN) for seamless connectivity between different Industry 4.0 devices (spanning from sensor to cloud). The OPC UA Publisher/Subscriber (PubSub) standard together with TSN standards will enable deterministic real-time networking capabilities that in so far needed proprietary extensions. TSN is expected to be established as a standard with wide availability of network interface chipsets in two years from now.**

In this whitepaper, the requirements for such Industry 4.0 devices are extracted from the use-cases described in the 'IEC/IEEE 60802 TSN Profile for Industrial Automation', implemented and tested using Open Source Software components. The performance measurements in this whitepaper focus on isochronous communication traffic as specified in the IEC/IEEE 60802 document, while other traffic types according to IEC/IEEE 60802 are still present in the network for the purpose of recreating load conditions similar to actual application use cases.

This whitepaper improves the results from the previous whitepaper (of the same name) and will help OEMs looking at technology readiness aspects for adoption into their products and services.

Keywords*— *Industry 4.0, M2M, M2C, PubSub, OPC UA, TSN

I. INTRODUCTION

Prior to Industry 4.0, manufacturers and users of automation components such as sensors, drives and PLCs were predominantly using a variety of different fieldbus/industrial Ethernet standards that are not interoperable (analog signaling before moving to legacy fieldbuses protocols such as Modbus, Profibus, DeviceNet over physical medium such as RS485, CAN, etc for the last 40 years).

Since the last 10 years, there are multiple Ethernet based communications protocols promoted by different PLC vendors in the market. This has created a fragmented ecosystem where device manufacturers constantly face additional costs to support as many protocols as possible in their automation products. This also prevents end-users like smart cities and smart factories from seamlessly interconnecting equipment from different manufacturers.

To fully exploit the potential of Industry 4.0 and Industrial IoT, it must be possible for data to flow securely to wherever in the system it can add value: from field devices like sensors and actuators to the control devices, between control devices, to edge gateway devices, data aggregation/ingestion devices, to cloud-based applications and enterprise systems for big data analytics, IT-OT and supply chain integration. In recent years, the industry has been demanding the development of a standard communications interface that will provide increased interoperability together with increased bandwidth, reduced wiring complexity and reduced costs while also guaranteeing deterministic performance.

While OPC UA can address this at higher layers, it cannot by itself guarantee deterministic real-time data transfer – a typical requirement for field level devices. In such a scenario, TSN is an IEEE 802.1 standard that is fast emerging as the upgrade to the IEEE 802.3 Ethernet standard capable of delivering deterministic real-time performance at the layer 2 – data link. Going up to the higher layers, IEC 62541 OPC UA is fast emerging as the application data modeling standard to enable interoperable data exchange between devices, systems and applications.

Table 1 lists the related TSN standards both released and under development for audio-video, industrial and automotive use cases. This paper is focusing on the time synchronization standard IEEE 802.1AS and scheduled traffic IEEE 802.1Qbv that are important for industrial use cases. The configuration standard IEEE 802.1Qcc is also an important topic for interoperability but is beyond the scope of this paper.

Table 1
TSN STANDARDS

Standard	Description
IEEE 802.1AS - a specific profile of IEEE 1588 (AS-rev which is more suited for industrial applications is under development)	Timing and Synchronization for Time-Sensitive Applications
IEEE 802.1Qav	Forwarding and Queuing Enhancements for Time-Sensitive Streams
IEEE 802.1Qbu and IEEE 802.3br	Frame Preemption
IEEE 802.1Qbv	Enhancements for Scheduled Traffic
IEEE 802.1Qca	Path Control and Reservation

IEEE 802.1Qcc	Enhancements and Performance Improvements
IEEE 802.1Qci	Per-stream Filtering and Policing
IEEE 802.1CB	Seamless Redundancy
More to come as the standards evolve	...

Coming to IEC/IEEE 60802, a device is defined as an end station, bridged end station, bridge or access point. For the sake of this whitepaper, we define an "Industry 4.0 Device" as either an end-station in the case of single port endpoints or a bridged-end-station in the case of endpoints that have at-least a two-port switch. This definition typically includes industrial devices such as Controlling Devices, IO Devices, Drives, HMIs, Gateways, IPC/Cloud/Virtual Servers (and excludes pure network infrastructure devices like bridge, router, access point etc.). Examples for such devices:

- Controlling Device – Programmable Logic Controller
- IO Devices – Sensors, Actuators
- Drives – Power electronic drive systems used in motion control applications and robotics
- HMI – Human Machine Interface
- MES – Manufacturing Execution System
- SCADA – Supervisory Control and Data Acquisition System
- IPC – Industrial PC used to run a PLC or HMI or SCADA System

II. Full Stack Perspective

We see that market defines Machine to Cloud (M2C) architectures based on security requirements and Machine to Machine (M2M) architectures based on deterministic communication requirements. While the focus of this paper is on OPC UA PubSub over TSN for M2M architectures, M2C is also considered when including traffic for secure Client/Server exchanges. This is the basis for approaching the implementation architecture from a full stack perspective to visualize all the layers that can impact the performance between the applications on different devices. Section IV of this paper contains the results of i) measuring the deterministic real-time capability of the test systems, ii) the quality of the time synchronization between systems using 802.1 AS Generic Precision Time Protocol (gPTP) and iii) the quality of the time synchronization between the network time and the system time. The subsequent sections V and VI describe the performance of the network synchronized applications that run over an OPC UA TSN network.

Figure 1 shows the full stack perspective of end-to-end real-time industrial systems according to OSI layers.

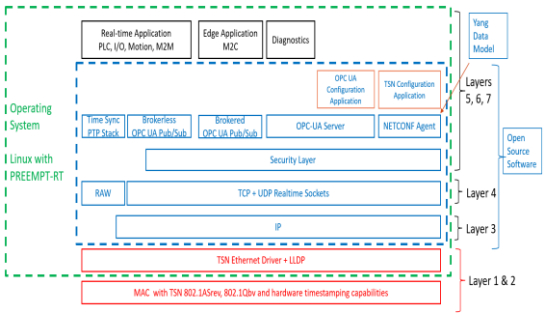

Figure 1: Full stack perspective for end-to-end real-time industrial systems

The performance measurements in this whitepaper focus on isochronous communication traffic while other traffic types according to IEC/IEEE 60802 are still present in the network for the purpose of recreating load conditions similar to application use cases that are relevant in actual end-user use cases. This is also the first version of the whitepaper prepared after the integration of fully featured implementation of the subscriber part. This is also the first time that the Fast Path Message (FPM) optimization is included in the publisher part of OPC UA PubSub.

III. Test Setup

A. Hardware setup

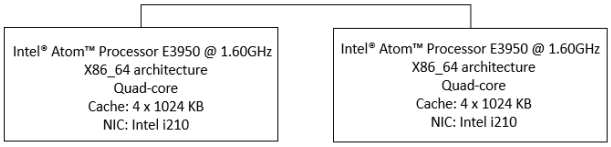

Figure 2: Hardware setup for performance measurements

The hardware setup consists of two quad core Atom processor systems with i210 network interface cards connected via PCIe. Both the systems are connected peer to peer with an Ethernet cable at a link speed of 1 Gb/s.

B. Software setup

All the software packages used in the system are open source and available to download from the respective developer forums. Intel's Earliest TxTime First (ETF) implementation in i210 igb driver is used for IEEE 802.1Qbv implementation whereas IEEE 802.1Qav is already present in the i210 solution. In order to increase the performance in the receiver side, XDP (eXpress Data Path) has been integrated with the Subscriber. The idea behind XDP is to add an early hook in the RX path of the kernel. To use XDP, bpf-next kernel tree is required.

Table 2
Software Package Information

Software package	**Versions**
Linux OS	lubuntu1-18.04.3, Kernel 4.19.37-rt19
Linux PTP	lptp v2.0
OPC UA stack	open62541 master
bpf-next	4.19

IV. Real-time Performance

The deterministic real-time performance (max latency or worst-case latency) of the system is measured using the *cyclictest* application. Briefly, *cyclictest* consists of a main task that initially starts a number of child tasks. The latter install cyclic alarms using a real-time capable timer at a given interval and repeatedly wait for the expiration of the timer. When a timer expires the current time is obtained and compared to the theoretically expected wake-up time. The difference that represents the inability of the system to exactly wake up in time, but a little later represents the so-called system latency. If the measurement is repeated frequently enough and under as many conditions as possible, the longest latency ever obtained may be used to describe a system's worst-case latency.

A. *Worst-case latency*

The longest acceptable worst-case latency of a deterministic real-time system depends on the system requirements such as cycle time and sample rate and may be as short as several tens of microseconds. Based on this observation and with the goal in mind that the results presented in this paper should be applicable to the vast majority of industrial use cases, it was decided that the real-time systems used for the tests described in this paper should have a worst-case latency of less than 70 µs.

The *cyclictest* application was run for a period of 12 hours in both the nodes without applying any additional load. While running the *cyclictest* application, the two nodes are time synchronized as described in the next section. The observed worst-case latency of the nodes is 46 µs.

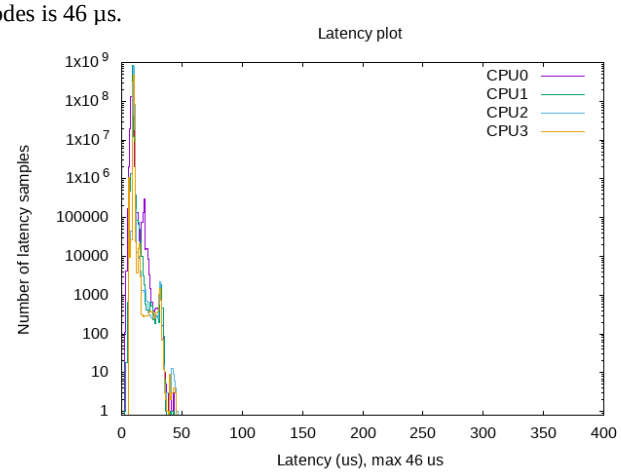

Figure 3: Worst-case latency of the node over 12 hours

B. *PTP synchronization*

Once *cyclictest* results were verified to be within the worst-case limit of 40 µs (as described in the previous section), we then looked at the PTP synchronization accuracy. In this case, Node 1 was configured to be the PTP master and Node 2 was configured to be the PTP slave. We used the gPTP configuration that is available in the linuxptp stack. The PTP hardware clock (PHC) of the PTP master serves as the clock for this network. The PTP slave adjusts its PHC to that of the PTP master. This ensures that the two nodes are synchronized.

Figure 4a shows the PTP clock offset between the PTP master and PTP slave that was measured using the 1PPS output from the i210 NIC that was observed in persistence mode using a DSO for a long run over multiple days.

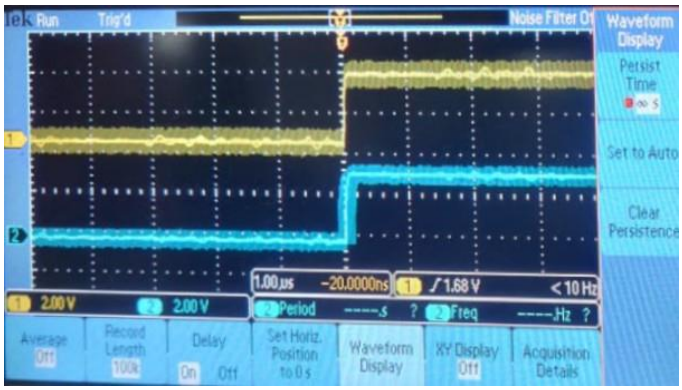

Figure 4a: Persistence view of PTP clock offset in DSO over multiple days

Figure 4b shows the results of the PTP accuracy measurement between the two nodes over a period of 12 hours; the observed clock offset is in the range of -1.2 µs to +1.2 µs and, thus, close to the performance that is theoretically achievable.

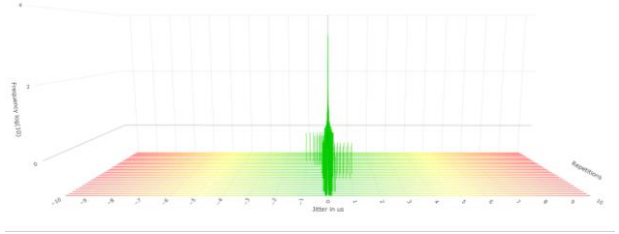

Figure 4b: PTP clock offset

C. *PTP hardware clock to system clock synchronization*

The system clock is used by most user-space applications; hence the system clock needs to be synchronized between nodes on the network. This can be achieved by synchronizing the system clock with the PHC by using the *phc2sys* utility.

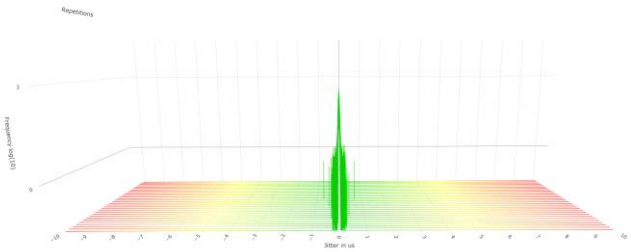

Figure 5: phc2sys accuracy on Master System (Node 1)

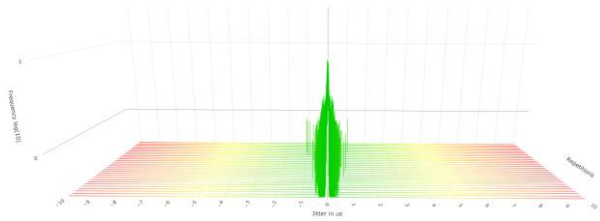

Figure 6: phc2sys accuracy on Slave System (Node 2)

Referring to figures 5 and 6, *phc2sys* jitter values of PTP master and slave nodes are at -5 µs to +3 µs and -140 ns to +124 ns respectively, it is evident that the system can synchronize the system clock to hardware clock at single-digit microsecond range which matches the requirements of most, if not all, industrial real-time applications.

V. PUBSUB NETWORK PERFORMANCE

Once the PTP synchronization was established and was verified to work accurately, the next step was to test the OPC UA PubSub application. In this case, OPC UA PubSub application was configured to publish Ethernet packets at 100 µs cycle time. One of the main factors that affect the performance is the jitter we experience to place the packet in the network. Figures 7 and 8 show the jitter performance of the OPC UA PubSub application for 1 million samples without ETF and with ETF respectively. It must be noted that the cycle time needs to be a multiple of 31.25 µs i.e. 125 µs. We

chose 100 µs cycle time for our measurements to show potential users that the 125 µs cycle time can be guaranteed.

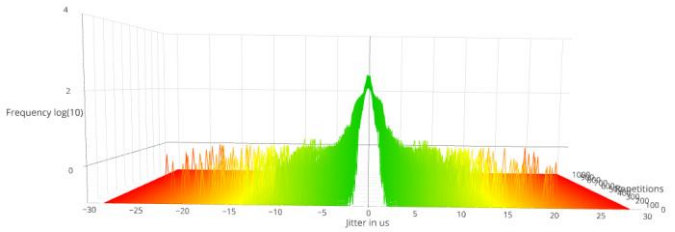

Figure 7: Jitter of OPC UA UADP publish traffic – without ETF

The packet to packet jitter of OPC UA UADP publish traffic without ETF functionality is found to be in the range of ±30 µs over 1 million samples. This places a severe limitation on the smallest network cycle time that can be achieved.

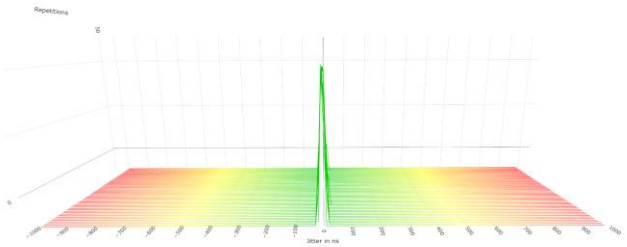

Figure 8: Jitter of OPC UA UADP publish traffic – with ETF

However, when we use the ETF feature, the packet to packet jitter of the OPC UA PubSub publish traffic with ETF functionality is improved drastically and is measured to be within ±48 ns over 1 million samples.

From the above, it is observed that using ETF is able to guarantee the deterministic transmit of OPC UA PubSub publish packet from Linux user-space with minimum jitter in nanosecond range even while publishing at very low network cycle times.

Figures 9a, 9b and 9c illustrate how ETF features enable placing the time critical isochronous OPC UA PubSub publish packets in the specified IEEE 802.1 Qbv windows.

Figure 9a shows a IEEE 802.1 Qbv base period of 100 µs split into four windows of 25 µs each. It also shows the eight different types of traffic present on the network and highlights that the time critical OPC UA PubSub traffic is always placed at the intended time in the beginning of the second window.

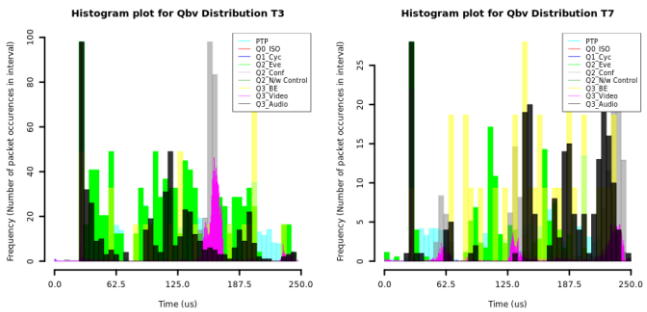

Figure 9a: Histogram plot for Qbv including all eight traffic types

Note: T3 denotes packets captured at Node 2 (for packets transmitted from Node 1). T7 denotes packets captured at Node 1 (for packets transmitted from Node 2).

Figure 9b is similar to 9a except that it shows only best effort ping traffic (leaving out the six other types of traffic) along with time critical OPC UA PubSub traffic that is always placed at the intended time in the beginning of the second window.

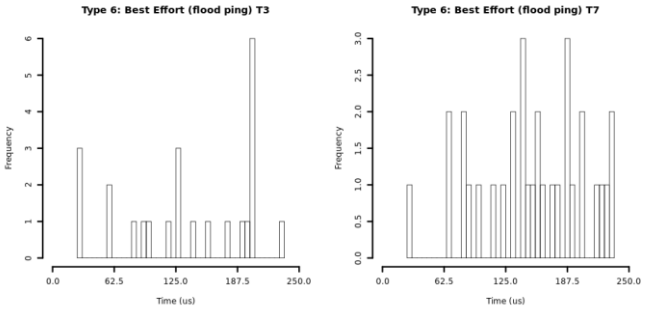

Figure 9b: Histogram plot for Qbv including best effort and PubSub

Figure 9c illustrates only the time critical OPC UA PubSub traffic for a clearer view when compared with 9a and 9b.

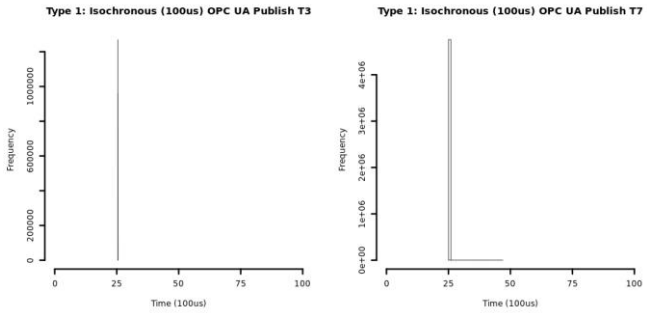

Figure 9c: Histogram plot for Qbv including PubSub only

ETF ensures that the transmission (Tx) of high priority traffic is prioritized through the high priority hardware queue. It also helps to buffer packets and make sure that the packets are sent out in the configured time before their deadline (Tx time). If ETF is not enabled, the network device will not be able to distinguish between priorities of different traffic. So, no guarantee for an end-to-end delivery time can be given when ETF is disabled.

VI. PubSub Round-Trip Time Measurements

The previous section looked at performance of an application in meeting the deadline to place a packet on the network. This section goes further and looks at the application round-trip time (RTT) measured using the OPC UA PubSub application over TSN. This is important for the communication latency from a PLC application to an I/O or motion control node application and back to the PLC.

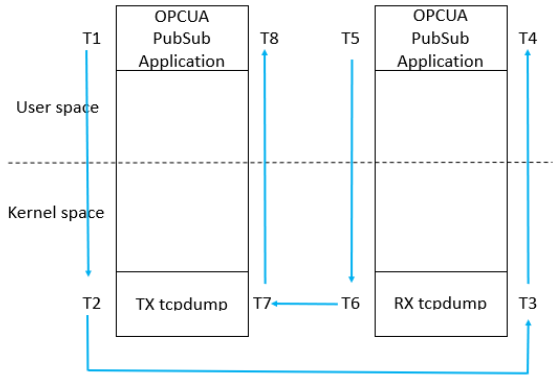

Figure 10: Trace point setup

T1 – Ethernet packet publish timestamp
T2 – tcpdump outgoing packet timestamp
T3 – tcpdump incoming packet timestamp
T4 – Ethernet packet subscribe timestamp
T5 – Ethernet loopback packet publish timestamp
T6 – tcpdump loopback packet outgoing timestamp
T7 – tcpdump loopback packet incoming timestamp
T8 – Ethernet loopback packet subscriber timestamp

The round-trip time is the time taken for the application in Node 1 to encode and publish a packet to Node 2 whose application decodes the received publish packet and loops back the same to make the published data finally available to the application on Node 1. From figure 10, T8-T1 provides us the round-trip time of the application. To keep track of the sequence in which the packets were sent and received at both ends, the payload in the OPC UA PubSub publish packet contains a counter variable that increments in every application cycle (in this test setup, the application cycle is same as the network cycle). Round-trip time = (2 * (Tx latency + Rx latency)) + (Loopback time on Node 2).

Tx latency – Time taken to encode and send data from one node to reach the network interface of another node (includes wire delay).

Rx latency – Time taken to receive and decode the subscribed packets from the network interface.

One of the main goals of this whitepaper is to recommend the best practice for where to place the application control loop. The approach followed in this whitepaper is similar to one of many options described in the IEC/IEEE 60802 Industrial Use cases document[4]. To guarantee low latency, we designed the application control loop to start at a fixed reference point in time after the subscriber thread has completed receiving data from the incoming packet. Once the application control loop thread completes its execution, the response value is then available for the packet preparation part of the publisher thread in the current cycle itself. This enables the actual publish to take place in the immediate next i.e. 2nd cycle (2 cycle reference model) to guarantee consistent application data update. This is illustrated in figure 11.

As shown in figure 11, transfer safety margin provides the safety margin to ensure that the subscriber thread has completed its job before the application control loop thread starts its execution. Similarly, the application safety margin provides the safety margin to ensure that application control loop thread has completed its job before the publisher thread starts its execution.

If we know the max latency of our system (from *cyclictest* results) and also the time spent in the locks inside each thread involved in publish, subscribe and application control loop as well as the priorities of each thread, it is possible to compute the application safety margin and the transfer safety margin. Then it will be possible to configure the maximum values for application safety margin and transfer safety margin and guarantee the deadline in the ideal scenario as shown in figure 11.

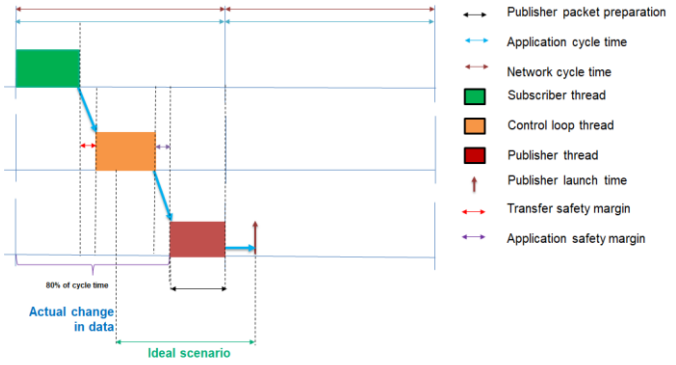

Figure 11: Ideal Scenario - Isochronous Publish Latency

However, it should be noted that this may mean the network cycle time is not the smallest possible value. Also, the final goal is to provide the lowest possible deterministic deadline for end-to-end transfer of data and not merely providing a very low number for the lowest possible network cycle time. In this paper, we are trying to achieve the lowest possible network cycle time and also comment on the worst-case deterministic deadline for end-to-end transfer of data in the lowest possible network cycle time as we see that this is the route most customers are expected to take. For example, figures 12 and 13 additionally show the practical and worst-case scenarios that can occur for an implementation that is based on the design in figure 11.

Figure 12 shows a scenario where there is sensor value change just after the application control loop scans the sensor inputs. In this case, the change will be available to the application control loop only after a one cycle delay i.e. until the application control loop executes in the next cycle.

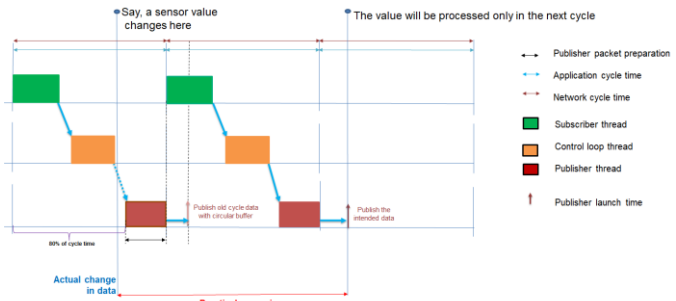

Figure 12: Practical Scenario – Isochronous Publish Latency

Figure 13 shows a scenario (in addition to the one cycle delay introduced in figure 12) where the application loop takes longer to complete execution (than the configured application safety margin) and introduces one more cycle of delay. This means the data finally

gets published in the 4th cycle. Depending on different configurations, this may mean a worst-case deadline of 3 cycles or 4 cycles.

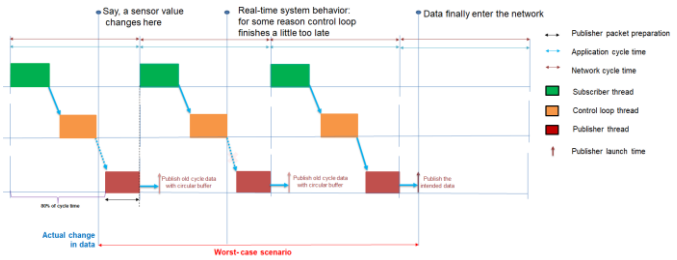

Figure 13: Worst-Case Scenario – Isochronous Publish Latency

Figures 11 till 13 illustrates the different scenarios that affect the isochronous publish latency of a node with respect to change in input data for the application control loop. We see that the publisher requires 2 cycles in the best-case scenario and 4 cycles in the worst-case scenario to reflect the change in input sensor value into its publish payload. Similarly, there will be 2 cycles in the best-case scenario and 4 cycles in the worst-case scenario that are consumed in the subscriber side.

When we measure round-trip time between two nodes, we have to consider the number of cycles required for a particular counter value to be sent from the application control loop of node 1 to the application control loop of node 2 and back to the application control loop of node 1.

As shown in figure 14, the counter value from the application control loop of node 1 takes one cycle to be published (as part of the payload in the OPC UA PubSub publish packet) to the network. It takes one more cycle for this published packet to be available at the subscriber side in node 2. Now, referring the 2 cycle reference model in figure 11, it will take two additional cycles for this counter value to be published by node 2 to the network. Finally, it will take one more cycle for this counter value to be available at the application control loop of node 1. Therefore, it will take 1+1+2+1 = 5 cycles for the best case round-trip of the counter value to complete.

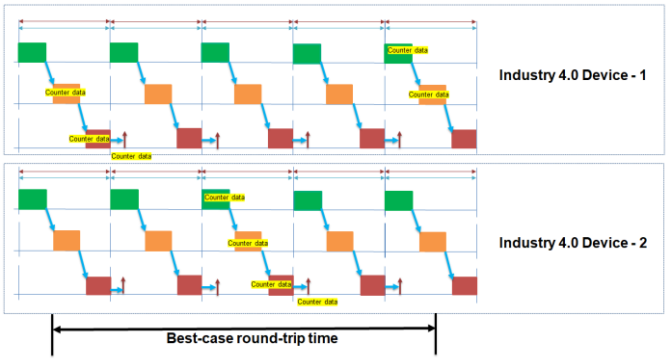

Figure 14: Best-case round-trip time between two nodes

According to the design shown above and referring the worst-case scenario according to figure 13, the counter value from the application control loop of node 1 will take three cycles to be published (as part of the payload in the OPC UA PubSub publish packet) to the network. It will take one more cycle for this published packet to be available at the subscriber side in node 2. Again, referring the worst-case scenario according to figure 13, it will take four additional cycles for this counter value to be published by node 2 to the network. Finally, it will take one more cycle for this counter value to be available at the application control loop of node 1. Therefore, it will take 3+1+4+1 = 9 cycles for the worst-case round-trip of the counter value to complete.

With the above design and the worst-case latency possible from it, we tried to configure the minimum possible application safety margin and transfer safety margin and a very low cycle time (say when the measured max latency was 34 µs, we were configuring the network cycle time to be 100 µs – a value that looks improbable to work). However, we measured the maximum RTT in 1 million samples for a payload size of 100 bytes when XDP was disabled in the receiver side, to be 940 µs as shown in figure 15. We found a packet miss of about 30% in a total of 1 million samples.

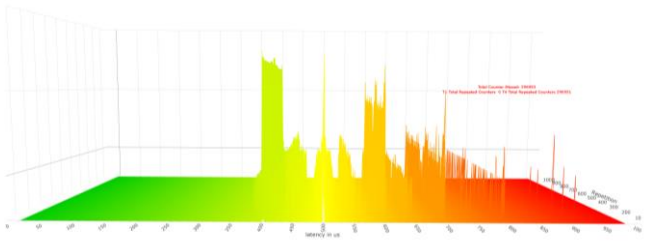

Figure 15: Round-trip time without XDP

When XDP was enabled in the OPC UA Subscriber, the maximum RTT in 1 million samples for a payload size of 100 bytes was measured to be 701 µs as shown in figure 16. The packet miss was 7% in a total of 1 million samples.

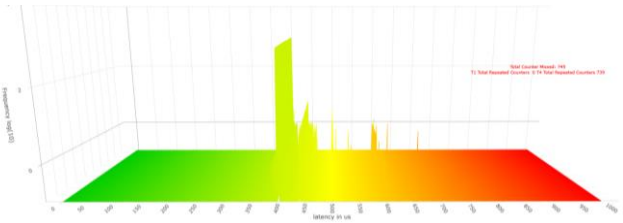

Figure 16: Round-trip time with XDP

This shows that when XDP is used with the OPC UA Subscriber, the performance was increased to about 25%. The above measurements were taken under nominal load conditions in which realistic CPU and network loads were applied to both the nodes. The CPU load was about 30% utilization on each of the four cores. Realistic network loads were created using Apache, sipp and OPC UA Client/Server traffic.

Note: On both nodes, Core 0 is made entirely available for OS and kernel. Core 1 is entirely reserved to run PTP to maintain time synchronization. Core 2 is used for running OPC UA PubSub application. Core 3 is used for user specific real-time application. Network loads (i.e. realistic load on Apache, sipp, OPC UA Client/Server) are run on Core 0 in this test setup which is a four-core system in which Cores 0, 1 and 2 are reserved as described above.

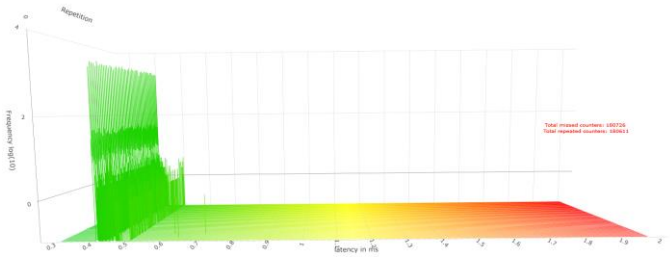

Figure 17: Round-trip time with XDP for 0.5 billion samples

Figure 17 shows that when the same XDP enabled Subscriber application was run for a longer period of time, the RTT obtained for half a billion samples was measured to be 700 µs which is within the expected worst-case latency of 9x cycle time with a packet miss of less than 10% of total samples.

Consistent data update is one of our prime objectives in this entire test scenario. It should be noted that throughout the experiment consistency of data exchanged within each cycle is checked. This is done by including 8 counter variables and check if all the 8 counters were successfully updated in a single cycle. The application is configured to stop if this consistency check fails.

Figure 18 shows the slope graph for missed counters running the OPC UA Publish packets in 100-byte payload size at various application cycle time and network cycle time under nominal load condition in a total of 1 million samples each. The entire measurement is taken at a 1 Gbps network link speed. The counters missed is 0 when the cycle time is greater than 250 µs. As the cycle time reduces below 250 µs the number of counters missed increases.

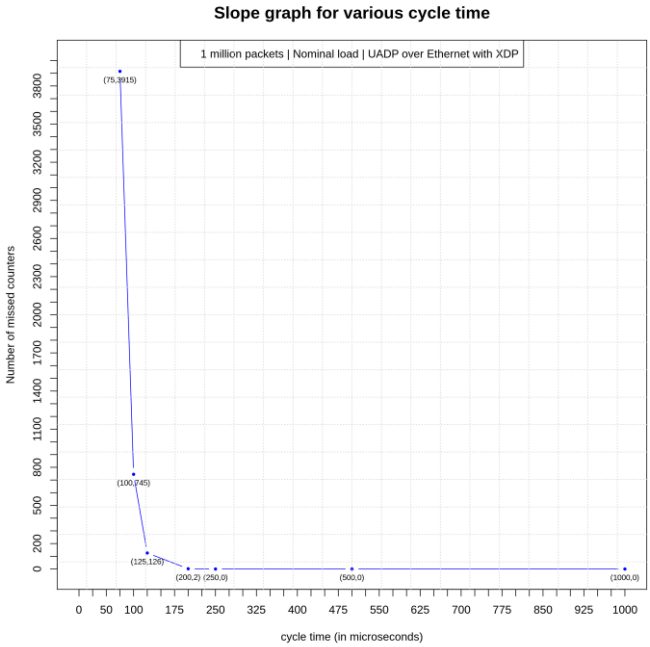

Figure 18: Slope graph for missed counters at various cycle time

Figure 19 shows that the Round-trip time measured with 100-byte payload size at 250 µs application cycle time and network cycle time in a total of 5 million samples shows only 1 occurrence of missed counters and repeated counters with a maximum round-trip time latency of 1.25 ms under nominal load condition. This is our recommended cycle time which can be used for motion control applications.

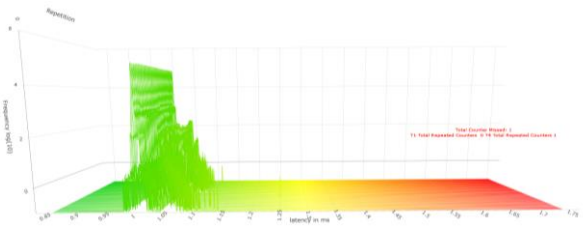

Figure 19: Round-trip time measurement in 250 µs

Our next steps involve improving the stability and reducing the network cycle time to achieve better performance at cycle time lesser than 250 µs.

VII. KNOWN CHALLENGES & NEXT STEPS

The above measurements were made in a test setup that used OPC UA PubSub packets with 100-byte payload size at 100 µs application and network cycle time. When we shortened the cycle time, the observed occurrences of data duplication or data loss (i.e. counter values were repeating or missed respectively) was minimum in XDP integrated Subscriber than that of the Subscriber without XDP.

Figure 20 shows the Round-trip time measured with 100-byte payload size at 31.25 µs application cycle time and network cycle time measured in 3.1GHz Intel i5 processor PCs shows a total missed counter occurrence of 2.5 million in a total of 5 million samples. By integrating Just in Time compilation (JIT) and XDP transmit in the publisher side along with the XDP ZC in the subscriber we can enhance this existing performance number thus achieving the industry standard 31.25 µs network cycle time in 1.6 GHz Intel Atom processor PCs thus enabling the solution to run in current loop applications.

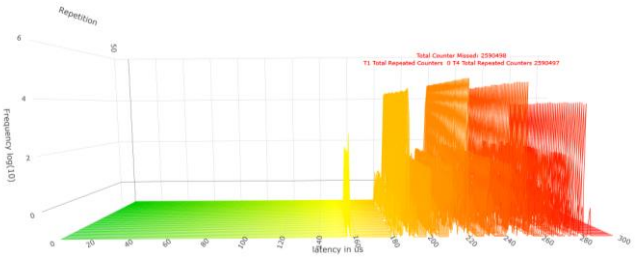

Figure 20: Round-trip time measurement in 31.25 µs

The market requirement expects the number of Industry 4.0 devices in the test network shall be increased and each of the devices shall be configured to publish more packets as well as subscribe to a larger number of publishers. The impact of OPC UA Client/Server exchanges on the values (i.e. need to update the same nodes in the information model for both Pub/Sub and Client/Server) needs to be considered. The performance impact introduced due to using security in both Client/Server and Pub/Sub shall also be explored. There is also the possibility to achieve more real-time performance in Linux to reduce the network cycle time further. The design calculation used to arrive at the number '9' can be promised as the deterministic

boundary of the system and if needed, possible design alternatives can be envisaged. Also, in the event of the system ever breaching the deterministic boundary, the user may have to implement the required safe shutdown/error handling and recovery mechanisms according to their application use case.

There are additional performance improvement topics that the authors are currently working on but have not yet been included in the scope of this whitepaper. These topics include:

- Multithreading (multiple publish & multiple subscribe)
- Just in time compilation (JIT) for Fast Path Message
- PTP packets are presently in the main interface and are not using a particular VLAN interface (intended traffic type as per IEEE 60802), *linuxptp* working branch has the support to run PTP in the VLAN interface and shall be incorporated in our test environment once it is available as a major release
- On the OPC UA PubSub publisher side, significant reduction in RTT has been observed after the implementation of FPM and it has been noted that the same improvement will also be possible with FPM at the subscriber side
- XDP has showed significant improvement in the OPC UA Subscriber side, XDP_transmit can be incorporated into the OPC UA Publisher to achieve more realistic performance
- Moving to Linux kernel version to 5.4
    - TA PRIO to use the time aware shaper for Qbv implementation
    - Receive side XDP ZC optimization. XDP ZC can be used in OPC UA Subscriber to further optimize the PubSub communication. However, XDP ZC support is not available for igb drivers on i210 NIC.
- Testing in a larger network with more subscriptions per Industry 4.0 device

VIII. SUMMARY

From the series of measurements done in the Embedded Optimized OPC UA stack, along with TSN drivers and a real-time Linux system, it is seen that the open source solution is able to achieve cutting edge performance demanded by modern Industry 4.0 devices.

As the project progresses with testing in larger networks, with more subscribers and publishers, we expect additional challenges with respect to computing the worst-case deterministic boundary. In such cases, we plan to leverage OSADL expertise in capturing long term data to prove deterministic behavior of the system. As and when we have new feature implementation pull requests merged to the project, we will be updating and publishing this whitepaper with improved results.

Those interested to know more about the above topics can get in touch with the open source community project and its members with their queries and contributions to make further improvements and achieve better numbers. The online forums at https://github.com/open62541/open62541/issues can be used for this purpose.

ACKNOWLEDGMENT

The authors would like to thank the industry consortium supporting the joint project for developing the PubSub extension for the open62541 OPC UA SDK. Special thanks go to Carsten Emde of the Open Source Automation Development Lab (OSADL) for assistance to setup the real-time systems and to Julius Pfrommer of Fraunhofer IOSB for providing continuous support on the open62541 stack development and maintenance.

REFERENCES

[1] "Hardware Assisted IEEE 1588 Clock Synchronization Under Linux" By Bálint Ferencz, Budapest University of Technology and Economics

[2] "The Road Towards a Linux TSN Infrastructure", Jesus Sanchez-Palencia

[3] "OPC UA TSN A new Solution for Industrial Communication" by D. Bruckner, R. Blair, M-P. Stanica, A. Ademajz, W. Skeffingtonx, D. Kutscher, S. Schriegel, R. Wilmes, K. Wachswender, L. Leurs, M. Seewald, R. Hummen, E-C. Liux and S. Ravikumar

[4] "IEC/IEE 60802 Use cases for Industrial automation v1.3" by Rudy Belliaridi, Josef Dorr, Thomas Enzinger, Florian Essler, János Farkas, Mark Hantel, Maximilan Riegel, Marius-Petru Stanica, Guented Steindl, Reiner Wamßer, Karl Weber, Karl Weber and Steven A.Zuponcic.