Enabling the Scalability of Industrial Networks by Independent Scheduling Domains

Konstantinos Christodoulopoulos[1], Wolfram Lautenschlaeger[1], Florian Frick[3], Nihel Benzaoui[2], Torben Henke[3], Ulrich Gebhard[1], Lars Dembeck[1], Armin Lechler[3], Yvan Pointurier[2], Sébastien Bigo[2]
[1] Nokia Bell Labs, Stuttgart, Germany
[2] Nokia Bell Labs, Paris, France
[3] ISW, University of Stuttgart, Stuttgart, Germany
konstantinos.1.christodoulopoulos@nokia-bell-labs.com

Abstract: We propose to extend the scalability of Time Sensitive industrial Networks, by partitioning them into time/scheduling domains and interconnect domain-devices through an optical backbone acting asynchronously to them. We show drastic scalability improvements and a proof of concept. © 2020 The Author(s)
OCIS codes: 060.4259, 060.4256, 060.1810

1. Introduction

The convergence of information technology (IT) and operation technology (OT) into a single infrastructure over factory floors is catching growing interest by Industry 4.0 [1]. Previously independent machines are expected to cooperate in larger areas, augmented/virtual reality applications to proliferate and control systems to be virtualized in enterprise data centers (DC). Industrial field busses like Profinet, Sercos, or EtherCAT have been efficiently fulfilling the stringent performance requirements (determinism) of OT, but they do not allow for converged networks. Time Sensitive Networking (TSN) has been developed for that purpose. It extends the Ethernet standard (IEEE 802.1) with real-time capabilities [2]. Key functions are time synchronization, traffic shaping including gating, prioritization, preemption, and configuration mechanisms of involved switches and devices. While prioritization and preemption are well suited for protection of a time critical class against greedy (best effort, BE) IT traffic, they do not help against collisions within that critical traffic class itself. With increasing number and diversity of end-devices, only reservations and cyclic scheduled gating (IEEE 802.1qbv) could provide reliable isolation of time-sensitive flows among themselves. However, scaling a flat TSN Qbv network (Fig. 1a) is hard. The interconnection of thousands of devices and multiple applications with strict and diverse (e.g. scaling over several orders of magnitude) timing requirements leads to synchronization problems, unsolvable scheduling, and efficiency issues on critical links.

In this paper, we considerably alleviate these limitations by proposing a new hierarchical architecture. We partition the TSN network into independent time/scheduling domains and use an optical backbone that acts asynchronously to them to tunnel the traffic of the multiple independent TSN domains simultaneously, along with possibly heavy IT traffic (Fig. 1b). A TSN domain preferably covers all devices and controllers which can spread across locations that require strict scheduling and a common clock base by their combined application. The optical backbone interconnects the multiple location of TSN domains while removing the need for synchronization among the domains, among logically unrelated devices. We show the improved scalability by applying to big factory halls a scheduling algorithm we developed. We also prove experimentally the feasibility of our approach: we show independent time synchronization of TSN domains across the shared, yet asynchronous backbone, and report transparent transmission of tightly scheduled traffic over a tunnel.

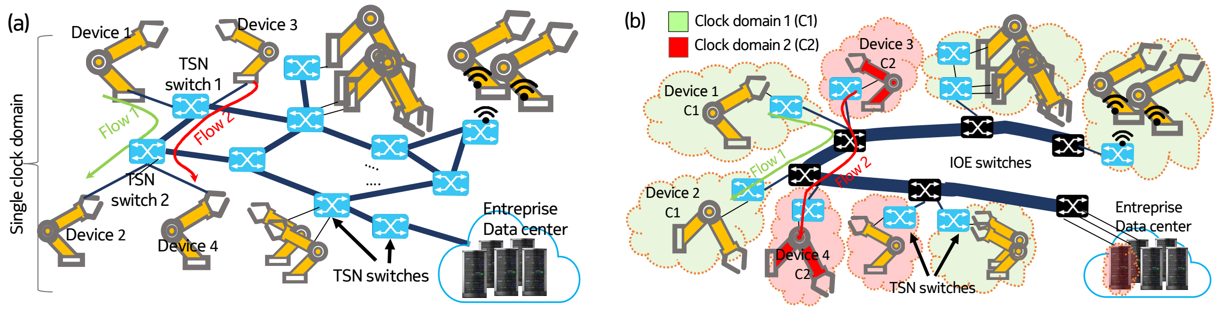

Fig. 1: (a) Reference flat TSN Qbv network, and (b) proposed architecture with independent clock/scheduling domains.

2. Managing the TSN scalability by domains

Gated network operation requires a precise common network time base. The gates are operated according to a globally coordinated schedule for each device [3]. Drifts and measurement inaccuracies of travel time between pairs of end-devices (e.g. by Precision Time Protocol, PTP) create timing mismatch. Synchronization error forces the use of guardbands so that packets do not miss their allocated gates, but at the expense of efficiency (Fig. 2a). As the network size and path lengths increase, synchronization error increases [5][6], longer guardbands are needed and efficiency drops. Moreover, higher data rates require tighter gate timing and equivalent improvement in synchronization to achieve high efficiency. Finally, flows have diverging sizes and cycles over several orders of magnitude. Scheduling cyclic flows is NP-complete [3][7], time gets fragmented and allocation efficiency drops as the network grows.

The proposed backbone is Industrial Optical Ethernet (IOE), an upgraded version of [4] for industrial networks. Each IOE bit stream over a wavelength is partitioned into container slots. While BE client packets are aggregated into containers that have opportunistic access to free slots, TSN client packets leverage *isochronous* reservations of slots for dedicated port-to-port connections. For that, IOE defines a cycle of, e.g. 1000 slots on a 100 Gb/s link, and reserves slots according to the requested capacity. The inevitable reservation jitter at ingress due to IOE asynchronous operation to clients is compensated at egress. Isochronous reservations provide guaranteed packet delivery and deterministic latency with negligible residual jitter (ns range) [4]. Single packets, cyclic packet flows at any frequency, or even aperiodic flows are carried "as is" – provided they not exceed the reserved capacity. The absence of timing restrictions implies that the timing on parallel isochronous paths is independent of each other. We propose to use IOE's isochronous paths as tunnels for TSN traffic, including management and synchronization messages to form independent TSN domains. End-devices collaborating in the same locations form a so-called TSN *island* which can be connected to other collaborating islands or a controller at different locations by isochronous IOE paths. With the proposed architecture, the network-wide schedule and synchronization is broken down into independent schedules and per domain synchronization, alleviating bottlenecks and enabling the strict time management of more devices.

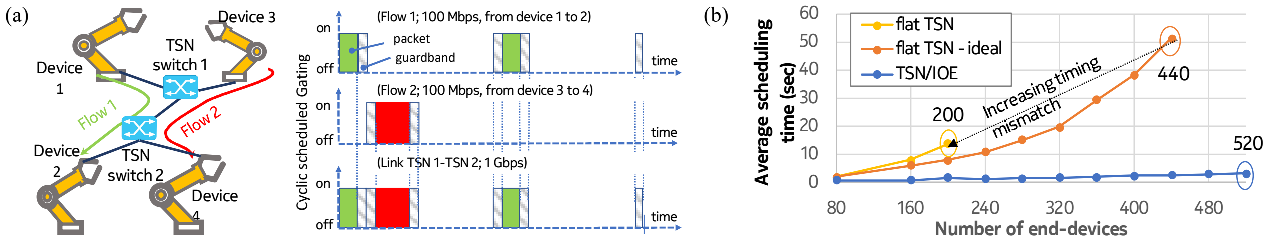

Fig. 2: (a) Cyclic scheduling in flat TSN network; guardbands cover synchronization error and fragmentation arise due to diverging flows' cycles and packet sizes, (b) Average scheduling time and number of end-devices served in flat TSN and TSN/IOE network.

We next provide a quantitative evaluation of the scalability benefits, by benchmarking a (heuristic) scheduler for the flat TSN network and an end-to-end scheduler that we developed for the TSN/IOE network (consisting of an IOE scheduler and independent TSN heuristic schedulers). We consider an industrial network with an increasing number of domains, each with 20 devices. In each domain a TSN switch is connected to an IOE switch in the case of TSN/IOE network, or to a TSN switch in the reference flat TSN setting. Domain controllers are assumed virtualized in a co-located enterprise DC. For each domain we select a cycle and packet size from: {0.1ms/250Bytes, 1ms/1250Bytes, 10ms/5000 Bytes}. We create flows so that devices communicate with their virtual controller and averaged for 50 instances. We assume that the synchronization error is 100 ns (close to lab measurements, Fig. 3b) for a TSN network of 20 devices and hence for all TSN/IOE domains due to their partitioning. This value is, however, "ideal" for a flat TSN with hundreds of devices where scheduling is performed with network-wide synchronization [8]. Achieving such synchronization at scale would require an additional network (white rabbit [6]), would be complicated and/or expensive. For this reason, we also assumed for flat TSN an error of 400 ns for ten times more devices (>200). Fig. 2b reports the average scheduling time as a function of the end-devices. The curves in Fig. 2b stop where the scheduler failed to find free slots for a flow; indicating the maximum number of devices served. From Fig. 2b we see that flat TSN scales drastically worse than TSN/IOE. The effect of fragmentation is visible, at ideal synch error of 100 ns, 440 devices were served, which drops to just 200 at 400 ns error. The heuristic scheduler used may not provide the ultimate fragmentation, but optimal scheduling is intractable [3][7]. The proposed TSN/IOE architecture relaxes these limitations. Its scheduler is still a heuristic but runs per domain where fragmentation is contained, making possible to manage 520 devices. The scheduling time in TSN/IOE grows linearly to the number of devices since it is calculated independently per domain, while the joint scheduling for flat TSN is cumbersome. The only drawback is that IOE backbone introduces latency from its asynchronous operation, which in our study was lower than stringent cycles (100 µs) [9]. Moreover, IOE can tradeoff latency for oversubscription per isochronous reservation [4].

3. TSN over IOE proof of concept

In this section we report on measurements and interoperability experiments that confirm the ability of IOE to support multiple independent TSN domains. The experiments were done with the FPGA-based platform of [4] which performs isochronous path reservations on an optical bus, modified with a smaller container (slot) size of 800 bytes.

In a first series of measurements we proved the strict mutual isolation between the concurrent isochronous path reservations and the best effort (BE) flows. There was no measurable impact on the path service quality, even in case of excessive overload in other tributaries. We do not reproduce the result here since they had minor changes to [4].

Then we investigated the latency and jitter of a reserved isochronous path. Fig. 3a shows the measured latency histogram of a constant bit rate (CBR) flow of 64-byte packets carried over an IOE isochronous path with 2 slots in a cycle of 16 in a 10 Gb/s bus. The mean latency was found to be 8.6µs, with a jitter of 31ns (standard deviation of latency). The primary source of latency is the IOE jitter compensation mechanism, which delays early-arriving packets so that all experience the same latency. This adds latency equal to $T=S/b$, where S is the container size and b the reservation bit rate, i.e. T=5.3µs. Other sources of latency are the transit delay (500ns per node), and the store-and-forward operation of the rate conversion from the reservation rate (1.2Gb/s) to the native port rate of IOE prototype (10GE). This could be canceled by forcing the reservation rate to be equal to native client port rate, i.e. both 1Gb/s.

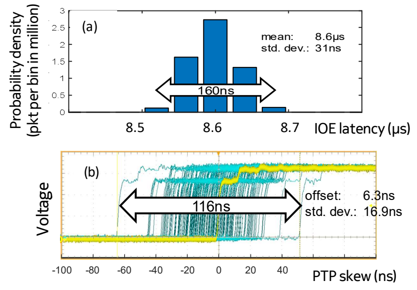

Fig. 3: (a) IOE path latency/jitter measuments, (b) clock alignment of two PTP synchronized TSN devices, master (yellow) and slave (green).

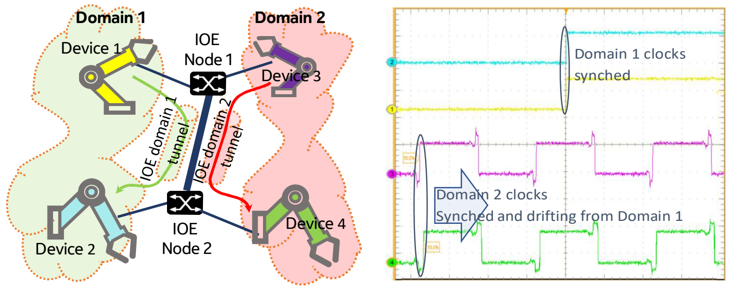

Fig. 4: Testbed setup with four TSN enddevices with independent clock oscillators forming interconnected over IOE to form two domains, and oscillogram of the four clocks.

We then performed interoperability tests with TSN equipment. Our architecture claims to synchronize and preserve timing properties of TSN-Qbv flows at both ends of a path/tunnel, hence assessing the compatibility and accuracy of PTP protocol over an IOE path is important. We used two experimental TSN end-devices and connected them over an IOE path reservation or, alternatively, through a cable. Both devices are equipped with free running clock sources at 50ppm precision. Once connected, the PTP protocol elects one of them as master and synchronizes the other, the slave, to the master. Fig. 3b shows the oscillogram of both hardware clocks. The slave is well aligned around the master clock pulse at mean offset of 6.8ns and a jitter of 16.9ns. For reference, if both TSN devices were connected by a cable, the mean offset was 1.3ns and jitter was 7.0ns. Overall, Fig. 3b shows that PTP effectively reduces the already small residual jitter of IOE (Fig. 3a). Note that the IOE nodes were not synchronized with the devices.

Next, we proved that we can support multiple independent TSN domains over a shared IOE infrastructure (Fig. 1b). We connect two pairs of TSN end-devices over two IOE reserved paths (Fig. 4). Each pair had its own time domain, i.e. own PTP instance and own clock master. Our oscilloscope recorded simultaneously all four clocks and displayed two well-aligned clock pairs and one pair randomly drifting against the other, proving the independence of the two domains (Fig. 4). As before, IOE was not synchronized to any of the two clock masters. Finally, we scheduled two CBR flows with cycles of 8 µs and 16 µs, respectively, over two TSN switches in a 16 µs cycle. We measure their latency across the TSN network (2 directly connected switches) by varying the schedule offset at the second switch. We repeated the experiment after connecting the TSN switches over an IOE tunnel. The latency patterns of the flows were found unchanged, just shifted by the additional latency of IOE. Overall, IOE tunnel performs like a cable with constant delay which can be precisely accounted for by the scheduler to achieve strict deterministic flow performance.

4. Conclusions

To extend the scalability of TSN industrial networks we partition them into clock/scheduling domains and tunnel their traffic over an optical backbone. Leveraging the scheduler that we developed, we predicted that this approach allows for a typical increase of the number of time-critical managed devices by +160% and we proved experimentally the feasibility of our multi domain approach.

5. References

[1] M. Laeger, et al, "The Future of Industrial Communication: Automation Networks in the Era of the Internet of Things and Industry 4.0", El. Mag. 2017.
[2] IEEE Time Sensitive Networking Task Group http://www.ieee802.org/1/pages/tsn.html
[3] S. S. Craciunas, R. S. Oliver, M. Chmelík, W. Steiner, "Scheduling real-time communication in IEEE 802.1 Qbv time sensitive networks", RTNS, 2016.
[4] W. Lautenschlaeger, L. Dembeck, U. Gebhard, "Prototyping Optical Ethernet—A Network for Distributed Data Centers in the Edge Cloud", JOCN 2018.
[5] D. Fontanelli, D. Macii, "Accurate time synchronization in PTP-based industrial networks with long linear paths", ISPCS, 2010.
[6] F. T.Gonzalez, J. Dıaz, E. M.López, R. R. Gómez, "Scalability Analysis of the White-Rabbit Technology for Cascade-Chain Networks", ISPCS, 2016.
[7] K. Jeffay, D. F. Stanat, C. U. Martel, "On non-preemptive scheduling of periodic and sporadic tasks", IEEE Real-Time Systems Symposium, 1991.
[8] D. Mohl, M. Renz, "Improved synchronization behavior in highly cascaded networks", ISPCS 2007
[9] www.electronicdesign.com/electromechanical/virtualization-helps-cnc-machines-consolidate-real-time-processing