TSN Simulation: Time-Aware Shaper implemented in ns-3

Dennis Krummacker
German Research Center for
Artificial Intelligence (DFKI GmbH)
D-67663 Kaiserslautern
dennis.krummacker@dfki.de

Luca Wendling
University of Kaiserslautern
D-67663 Kaiserslautern
lwendlin@rhrk.uni-kl.de

***Abstract*—Recreating real-world circumstances with virtual means is capable of providing high yields in several areas. Research, development or assisting active operation with introducing new functionalities are very different in the way, how they utilize a simulation but can unfold its benefits in several, if not all, business sectors. The more technical a branch is, the greater can be the value of simulation assistance. Specifically in industrial production environments, a mimicking of the operative machine networks is highly valuable. On one hand since access to such working environments for things like trials or risky development activities is not advisable. On the other hand, a digital twin or digital entity of any appearance can be used for operative functionalities such as security measures. Hence, simulations are promising. Targeting our focus towards real-time communication applications, such as industrial production plants, Time-Sensitive Networking (TSN) is a communication technology of high interest. TSN is relatively new, which is why executing hardware or simulation environments are not broadly available yet. For this reason, we decided to start an own implementation of a TSN traffic forwarding mechanism, based on a network simulation framework, which is most flexible and can be adapted to a diverse variety of use-cases. As a result, we designed and implemented a queue discipline fitting into the abstraction model of ns-3.**

***Index Terms*—Industry 4.0, TSN, Time-Aware Shaper real-time communication, deterministic traffic, Industrial Network Communication, Simulation, ns-3**

I. Introduction

The long term goal of industry 4.0 is a modularized, flexible, highly dynamic and fully automated production plant. Achieved by a complex interconnection of sensors and actuators as well as communication technologies serving demands such as ultra reliability and real-time communication. Nowadays, there is still a long way to go to attain this goal. Production lines are mostly very unflexible, in terms of the produced goods as well as for the way of how they are operated. A step by step integration of new technologies paves the way towards industry 4.0 one by one. One of these technologies is TSN, delivering a very valuable feature-set since it is a communication technology granting reliable real-time communication and raising the versatility of network communication in an industrial context. TSN is however currently still actively standardized and not finished. Compliant hardware is currently in the process of progressive market introduction, gradually growing more feature- and performance-rich. A significant expansion within the industrial landscape is still a long way off. Not limited to but especially in such a phase, new developments, trial runs and use-case analysis are highly valuable and required. But industrial communication networks are very specific and often times tightly customized. A real-world replication is not viable and an intrusion for testing of new features is not reconcilable with a running operation in a working environment.

This is where a simulation becomes beneficial. Mimicking the behaviour of a network communication system in a virtual environment can exhibit its merits in multiple facets. For one, it allows to carry out experiments under controlled circumstances without danger for interference with any working operation, which is of great use for reasearch and development, for production life-cycle maintenance, error-analysis and bug-fixing.

But also for assisting a production grade environment, concepts like a digital twin, intrusion detection systems or Honeynets are popular tools which are realized via or are able to benefit from a good simulation framework. The utility from such production grade uses of a simulation range from security over predictive maintenance, monitoring, self-optimization to evaluation and resilience assurance or even enhancing the user-experience or establishment of new functionalities. The applications for a simulation are various.

In this paper, we present a scheduled traffic forwarding mechanism, an implementation of the Time-Aware Shaper (TAS) (IEEE 802.1Qbv) as a queue discipline extension to the network simulator ns-3. As implemented according to ns-3's modules conventions and fitting perfectly into its abstraction model, our implementation maintains all of ns-3's strengths, such as powerful physical layer simulations.

II. Related Work

[1] analyzed different simulation frameworks with regard to an industrial application. Besides application specific, proprietary implementations, there exist two well known simulation frameworks: OMNeT++ & ns-3. We chose ns-3 as our preferred platform for network simulation for certain reasons. A very strong point was the possibility for a live-operation, which means that the simulation can be attached to a real-world network and works life in conjunction, which allows to mimic the behaviour of existing hardware, resulting in real-world influences. Also in favour of ns-3 spoke its flexibility and abstraction model. These grant the capability to adapt all levels

of a communication chain in detail as well as taking precise measurements. Not to forget the very high achievable timing precision and from all those mentioned facts a generally good degree of representation of a real-world network. In addition is ns-3 very powerful in modeling physical layer influences. Simulating wireless channel models and achieving a good, representative estimation of a path loss is where many other simulation frameworks come short. Since wireless TSN is a very promising topic of future industrial networks, the combination of a TSN simulation with a realistic wireless network model is a crucial point deciding about usefulness in industrial applications.

A. Existing Simulation Solutions

Simulation approaches for TSN where already implemented prior to the work at hand. As found by the authors, these were all made with OMNeT++. Previous to the release of IEEE 802.1Qbv, [2] investigated the concurrent influences of AVB traffic shaping and time-triggered transmission, for which purpose they have built simulation modules. Since their work released preliminary to IEEE 802.1Qbv, they developed an own, unfortunately non-conformant solution for time-triggered transmissions. With [3] TSimNet was proposed, an OMNeT++ simulation model focusing on the non-time-based TSN components and thus covering a different scope. [4] presents a simulation model for IEEE 802.1Qbv (traffic scheduling) and IEEE 802.1AS (time synchronization) in OMNeT++. The authors evaluated an example scenario, which showed scheduled traffic unaffected by best-effort traffic. The code developed and used there was not made public. In [5], the authors present modules for OPNET covering IEEE 802.1Qbv and IEEE 802.1Qci. The implementation was not made publicly available. [6] created NeSTiNg, a network simulator for Time-Sensitive Networking, based on OMNeT++/INET. Their contribution was to extend the existing framework by timed gating mechanism (IEEE 802.1Qbv), frame preemption (IEEE 802.1Qbu and 802.3br) and the Credit-Based Shaper (CBS) (IEEE 802.1Qav).

As far as known by the authors of the present paper, no TSN implementations for ns-3 were available as of writing this. Also investigating the feature-list and code-base of ns-3 asserted this. The only work found in this regard was a "simplest possible CPRI fronthaul proof-of-concept" simulation [7]. This only showed the feasibility of a TSN-alike scheduled traffic simulation in ns-3 with one gate and very inflexible. Essentially, the result was just the insight that scheduled forwarding is possible in principle to be simulated in ns-3.

The main challenge for TSN as well as the sole reason for its invention is to support the application domain, that is, to fulfill the communication demands of applications that require a certain QoS for proper functioning. Various works were done to analyse the TSN procedure itself, either for being able to serve very demanding applications [8], to introduce new functionality or to converge various physical layer technologies [9]. These were either relying on simulations or could have profited from the existence of valuable simulation environments. With existing solutions, only wired environments could be investigated, which do not pose the high challenges in reliability like a wireless physical layer does.

The many found simulation approaches regarding the field of TSN as well as their application demonstrate the topicality of this area, whereas the absence of modules for ns-3 was the motive power for us to realize it. Because for an application of such high demands in reliability like the industry, precise and dependable results including realistic wireless influences are indispensable.

III. Background

First, we like to give an introduction to selected topics of TSN as well as ns-3 before we combine them. The section about TSN shall give a basic understanding of the techniques behind our simulator implementation. Along explaining the technical background, we would like to accentuate the complexity of TSN networks and the thus resulting non-triviality in predicting, analysing or mimicking them.

A. TSN & Scheduling

Under the term TSN are assembling a variety of different IEEE standards as amendments to the IEEE standard 802.1Q. Some of them work in conjunction, others formulate alternative approaches for individual sub-functionalities. Heart are the standards addressing the actual execution of a traffic shaping, that is, the scheduling. Particularly, IEEE 802.1Qbv describes a *gate control mechanism*, which is essentially the basis for all QoS assuring techniques. The gate control is a time-triggered mechanism working according to a predefined schedule, which prescribes when traffic of which queue is forwarded, where queues can be associated with traffic classes. With this, a forwarding device (network switch) has access to multiple traffic queues per egress port. Packets to be forwarded through this egress port are sorted into one of these queues according to defined rules, like mapping of specific queues to dedicated traffic classes. Defined by a gate opening schedule, at any point in time certain gates may be defined as either closed or open. Only from a queue with a gate set as open, packets can be forwarded. The

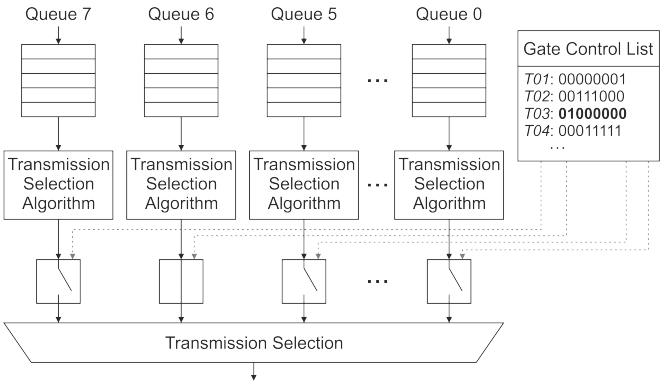

Figure 1. Gate Control and Transmission Selection architecture of a TSN switch, compliant to IEEE 802.1Qbv. The transmission selection algorithms are responsible for shaping the traffic of the individual queues. As a transmission selection algorithm, for instance CBS can be set in place. The figure shows a certain point in time, where only the gate of queue 6 is open.

gate control mechanism is only a first stage in an ensemble of traffic shaping mechanisms (see Figure 1), where optionally further scheduling mechanisms can be combined or attached to individual queues. With priority based scheduling for example, a decision is in place for points in time, when multiple queues are simultaneously open: A packet is chosen from the one open queue with the highest priority, containing a pending packet.

The most important scheduler to ensure deterministic communication within low delay bounds and jitter is the TAS as specified in IEEE 802.1Qbv "Enhancements for Scheduled Traffic". This scheduler defines the basic procedure, introduced earlier and depicted in Figure 1. Per egress port, up to 8 queues may be implemented on a switch. Between each queue and the Transmission Selection, a so-called gate is placed. A packet from a queue can only be transmitted during time-slots when the associated gate is open, which is controlled time-triggered according to a predefined, cyclically repeating Gate Control List (GCL). For transmitting onto the physical channel, the Transmission Selection chooses a packet from one out of all queues with a currently open gate. Yet another scheduling algorithm can be used to decide about this. In the most simple approach, among all open queues with a pending packet, the one with the highest priority is released. For the sorting of incoming packets, it is prerequisite that they contain the additional header information from the IEEE 802.1Q standard. Part of the VLAN-tag inside this is a 3-bit Priority Code Point (PCP). Values of this PCP are mapped into corresponding queues. The GCL is calculated beforehand operation. It contains a sequence of gate configurations. Every entry has a state for each gate, either opened or closed and defines a duration. After the duration of an entry is passed, the current gate configuration transitions to the direct successor (e.g. in Figure 1, when going over to T02 from T01, gate 0 is closed and the gates 3, 4 & 5 are opened). After the whole list was processed, it restarts with the first entry. It becomes evident that precise timing and more still a synchronization of local clocks between distributed devices is an inevitable requirement for a proper operation. This was also shown in [10]. Usually taken for solving this is the standard IEEE 1588 or IEEE 802.1AS / AS-Rev, which is a profile building upon 1588. Clock-synchronization is already no trivial task in itself. Even more, depending on the physical transmission technology. Using a wireless interconnection, clock-synchronization can profit from additional attention beyond standard IEEE-1588 based solutions as approached by [11]. However, currently clock-synchronization is not in the scope of this paper's simulation implementation but yet another piece turning TSN networks very complex in operation and difficult to predict.

To show how various schedulers might be used in conjunction, we'll briefly introduce another one, the CBS (defined in IEEE 802.1Qav "Forwarding and Queuing Enhancements for Time-Sensitive Streams". A CBS can be attached to an individual queue, where it would be placed after the queue, before the gate. The intention behind CBS is to create fairness among services, for which the CBS maintains a *credit* value. A packet is only eligible for transmission if the credit of the queue is non-negative. As long as packets are transmitted from the queue, the credit decreases. The amount by which credit is decreased per time is defined by the *send slope*. While packets are waiting inside the queue, credit is built up by a certain rate called *idle slope*. In case no packet is pending for transmission while the queues would be allowed to, positive credit is reset to zero. This procedure smoothens bursts and – if applied to more than one queue at a time – can prevent lower priorities from being overruled over a long period of time. For deterministic, real-time traffic, TAS is better suited since literature [12]–[14] showed that delay bounds for CBS are often higher.

Other standardized schedulers are the Asynchronous Traffic Shaping (ATS) (IEEE 802.1Qcr) and Cyclic Queuing and Forwarding (CQF) (IEEE 802.1Qch). The major advantage of ATS is that it does not require synchronized clocks still nevertheless providing deterministic delay bounds. ATS is a per-hop/per-flow traffic shaping method. It requires a fixed amount of queues independent of the number of involved flows, which drastically increases it's scalability with respect to number of flows. However, the standardization of ATS is not fully finished yet and thus not in the center of our interest.

An important circumstance to consider is, what happens when a gate closes while a packet is currently being transmitted. With no consideration in place, a frame-sending can start any time during an open gate. In case a frame is too long for a time slot or its transmission is started so late in its slot that the transmission takes longer than the remaining time, the bounds of the time slot are violated. This causes packets of the succeeding time slot to start transmission delayed. This, of course, subverts the whole idea of deterministic traffic and undermines the proper functionality of a scheduling mechanism. Different approaches exist to deal with this obstacle. One is to define a guard band. The guard band is defined as a time slice before the end of a time slot during which no transmission of a frame is allowed to start. If this guard band is set to the time the longest expected frame during this time slot would require to be transmitted, no violation of boundaries can occur. A strict definition for this value can cause a waste of transmission capacity. In a use case with packets of mixed size (which is very likely in most cases) packets shorter than the maximum expected size would be withheld from transmission during the guard band, even when the remaining time is enough for this small frame. While the guard band theoretically is adjustable and could be set to actually expected frame sizes, most of the time the expected frame sizes cannot be accurately predetermined or are just fitting the PDU (protocol data size) resulting in the guard band for TSN to be set equal to the PDU. And while the PDU also theoretically can be adjusted for many transmission technologies, for an Ethernet frame a maximum length of 1534 bytes (1518 bytes (frame) + 4 bytes (VLAN-tag) + 12 bytes (frame spacing)) is usually in place – which corresponds directly to the guard band. A measure to reduce the guard band significantly, is the more sophisticated functionality *Frame preemption*, defined in the IEEE standard 802.1Qbu. With that, the MAC is enabled to interrupt the transmission of a frame, when the guard band is reached. In case of a packet too long, this results in the

transmission of such a frame to be suspended just before the end of its time slot and subsequently to be resumed at the beginning of the next time slot with an open gate for it. Since the minimum size for a still valid Ethernet frame is 64 bytes, using frame preemption allows to reduce the guard band to a total of 127 bytes (A minimum frame of 64 bytes + the maximum size that cannot be preempted of 63 b).

For implementation, we have decided to give preference to the TAS for the reasons of its compatibility, performance and being most mature, while also the concept for this scheduler and its GCLs appears to be the one towards which industrial network switch manufacturers prioritized to orient their product developments.

As can be seen, an operation with conjunction of multiple scheduling mechanisms can make it hard to predict how a network will perform exactly, especially if the TSN-level measurements operate coexistently with higher layer measures. Other hard predictions result not only from the network setup itself, but the whole system setup. Questions arise, such as how best-effort traffics perform alongside a certain amount of scheduled traffic, while working under a certain forwarding mechanic configuration. Without going into detail, it is worth mentioning that a magnitude of further standards is involved in a proper TSN operation. These regard for instance path calculation, redundancy (multi-path, switch failure), handling non-conforming senders, etc.. Summarized is TSN an aggregate of multiple non-trivial mechanisms, which leads to a complex network behaviour.

As we already labeled hardware-based trial runs as not viable due to cost, attempts left for ascertaining an estimation of the systems performance are a formal network analysis or a simulation. A formal analysis is very hard, requires accurate and complex models and can be hardly reused for other setups. Therefore, once again, a simulation approach can offer advantages.

B. ns-3

Ns-3 is a modular open source simulation framework, which is used for scientific investigation of networks and creating networks itself. The modular structure allows a fast development of new network protocols on any imaginable transmission medium. ns-3 handles network traffic by means of a layered structure. Similar to the well known OSI model, for sending, packets are handed downwards starting from an application and for receiving, are handed upwards towards an application. The most important components of ns-3 are the traffic control layer, net device and the Channel. Compared to the OSI model, while not exactly matching, would the *traffic Control layer* be located at the end of layer 3; layer 2 is implemented by the ns-3's *net device* and the *channel* represents the physical transmission medium in OSI layer 1.

IV. TAS IN NS-3

For a resulting solution, it was of special importance, to be a clean and nicely integrable low layer solution. That means no application-layer or prototypical work-arounds, acting as an additional front-end on top of default network handling mechanics. ns-3 granted all capabilities to achieve an implementation well fitting into its modular framework. Accordingly is our final TSN implementation based on the traffic control layer. The traffic control layer is deeply embedded in the architecture of ns-3. Without the user having to configure anything, this layer is active all the time. Every sent data packet must pass through this layer before it is sent to the network controller. Sitting there, our implementation is able to control for every network packet how it is passed onto the transmission medium. Precisely put, is a packet first handed to the network driver, which is wrapping it into the frame for physical transmission and actually transmits it to the medium. Since ns-3 is a strict single-threaded simulation, this does not cause any impact. We implemented a channel access scheme working according to a TAS. This comprises of 8 queues, a procedure to sort packages to send into them, a gate control mechanic to control the gate openings as well as a handler for queue selection algorithms for the event of simultaneously open gates. Our solution supports a configuration for the following parameters:

Packet Filter: The packet filter classifies the priority to which a packet is assigned based on any parameter. To configure the packet filter, a function must be written. This function receives a `queuedisc` item as input. From this item, any desired information of the packet can be analysed. From the configured information, subsequently a decision can be made to add a VLAN-tag. This VLAN-tag is later used to sort the packet into one of the 8 queues. In case the packet filter is not able to classify a packet, it will be treated as best-effort. This mechanism can be dynamically configured during runtime but a dynamic approach consumes a considerably increased computation time, which is why we support a static solution, where the packet filter is set one time at the beginning of a simulation.

GCL: The GCL indicates at which points in time which gates are open and thus which priority classes are allowed to send. Our gate control mechanism adaptively reads a configuration (GCL) during runtime. During permanent operation, its configuration is performed in an endless cycle, as prescribed by TSN. In addition, the currently used configuration can be changed any time during a simulation to set a new GCL. For a transition, all queues are closed and transmission is suspended for the transmission time of a maximum large data packet. Packets that are attempted to be sent during this period remain in the queues until the transition is finished. Then operation continues following the new GCL.

Data packets traversing our TSN implementation are processed in the following order. At the beginning, the data packet is handed over to the traffic control layer, where our TSN handling code is attached “below” as a *root queue*, which is why every data packet is automatically going through the TSN implementation, immediately after the traffic control layer. Then, the packet is classified by the packet filter and inserted to the appropriate TSN queue depending on the result. In the case a packet cannot be properly classified, due to it not carrying the required QoS information, the packet is assigned the priority best-effort (queue 0). As soon as the packet arrives at the first position inside the queue and the corresponding gate is open, the next step is a

decision by a queue selection algorithm, in order to dissolve potential collisions of packets from different queues, in case multiple gates are simultaneously open. Currently implemented by us is the Strict Priority Selection algorithm. Due to the high level of modularity our TSN solution shares, it is possible to implement any other queue selection algorithm and use it in exchange, like e.g. the idle slope selection algorithm. When trying to remove a data packet, the `ListExecute` algorithm is automatically executed. This ensures that the correct entry of the GCL is always active at the time of transmission. The reason this needs to be checked by the removal of a packet is that ns-3 is a time discrete and event-based simulator. Normally, the GCL would be handled by an external clock, which triggers the transaction to the next entry in the GCL. This approach would mean that the simulation cannot end because there would always be an event in the simulators event list. First after ListExecute is finished, the queue selection algorithm tries to pick a packet from an open queue. After this was successful, i.e. at least one pending packet was available, this packet is passed on to the net device. The TSN module then prevents the removal of further data packets for the duration of the transmission. After a transmission, either the next packet is processed or the TSN module waits for a new packet to arrive. By utilizing the traffic control layer, it is possible to use our TSN implementation on every net device and every transmission medium since it is independent from those.

To assure the observance of time slot boundaries, our implementation checks the size of a packet before transmission and compares it to the remaining time. A frame is only sent if the remaining time suffices for the frame. This behaviour is similar to the guard band mechanic but much more precise since it automatically adapts itself to every packet & remaining time occurrence. Such a precise guard band handling is in principle restricted to a simulative solution and not affordable for a real world switch because it requires detailed knowledge about the packet to sent and is thus among other things bound to store-and-forward transmission. To implement *frame Preemption* as defined by IEEE 802.1Qbu, first an implementation of Link Layer Discovery Protocol (LLDP) would be required. Continuing, frame preemption would be implemented on on the *net device*.

V. Evaluation

A Python script is used for validation. This script creates a random configuration which is applied to a test scenario. The test scenario consists of 2 devices, where one is configured as a client and one as a server. The client tries to send UDP messages to the server which will immediately answer with a copy of that message. The client will always send 4 UDP data packets per time slot of the GCL. The packets are evenly distributed over the timeslot. The GCL for those devices is so configured that always only one device can communicate. With this setup, it is easy to automatically check if the GCL has been complied. For analysis, the simulation creates pcap files for every device. Those are then evaluated by the Python script. In 100.000 simulation the implementation has worked as expected.

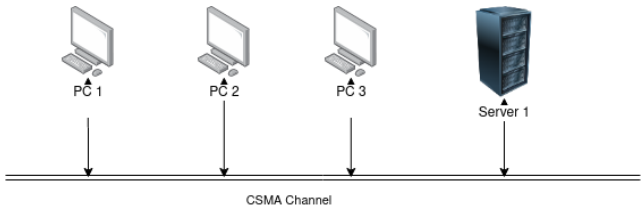

Figure 2. Test setup for Evaluation. PC 2 communicates with the server. PC 1 and PC 3 send with maximal bandwidth to overload the CSMA Channel.

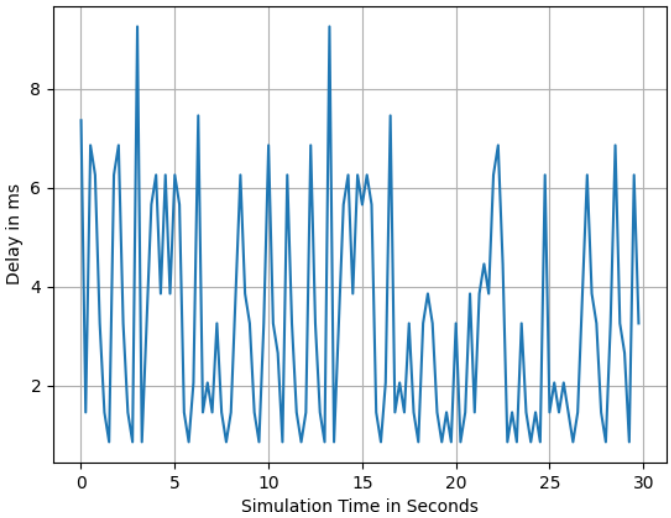

Figure 3. Delay of PC 2 without an active schedule.

Figure 2 shows another test setup. While PC 1 and PC 3 overload the CSMA channel, PC 2 tries every 250 ms to communicate with the server. In the first scenario, no measures were taken, the medium access happened competitive. The resulting delay is plotted in Figure 3 and can be seen to be heavily fluctuating. For the second scenario, applying a schedule results in Figure 4. This schedule granted PC 2 a VLAN-tag with high priority and a reserved time slot for each re-occurring transmission. By that, the fluctuation in transmission time is gone and remaining is the total path propagation delay. Similar measurements were taken for the jitter. Without our TSN implementation, also the jitter was heavily fluctuating, up to ~7.5 ms. With the schedule in place, the jitter basically completely went to 0. For the other two PCs (1 & 3), the transmission delay was increased at instances where a channel access competition with the scheduled traffic would have occurred.

Regarding simulation performance, a numerous series of simulations with an increasing of amount of participating devices

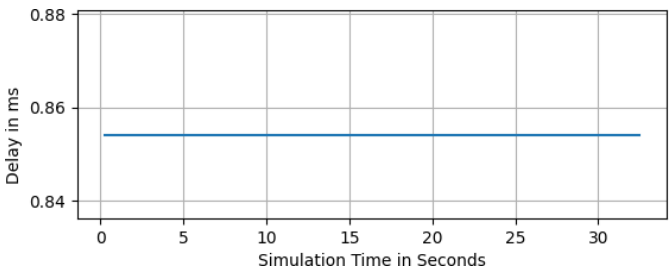

Figure 4. Delay of PC 2 with an active schedule and packet filter.

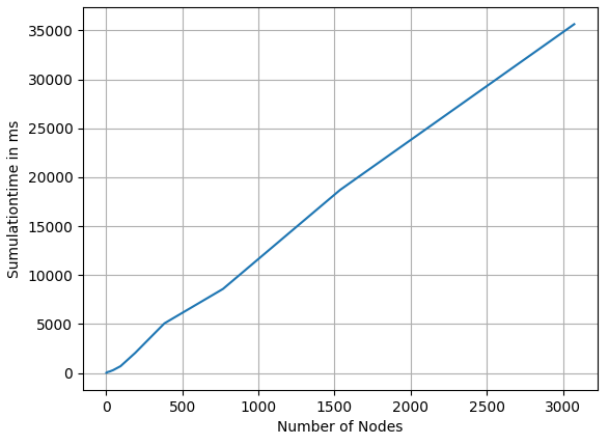

Figure 5. Scalability of TAS simulation.

was carried out. The simulation computed a duration of 30 sec. In each single run ⅓ of all devices was communicating via real-time traffic with a TSN schedule and the remaining ⅔ were spamming the channel with best-effort traffic. Figure 5 illustrates that the required computation time scales linear with the amount of simulated devices.

VI. Conclusion & Future Work

We implemented the 802.1Qbv gate control mechanism in ns-3. That is, a fully functional TAS in form of a queue discipline, a packet filter for sorting into queues and a queue selection algorithm to dissolve gate opening conflicts. In its current state, it supports to simulate the most essential schedule execution of TSN in an ns-3 based simulation. With receiving a GCL, a device is subsequently able to perform deterministic, reliable and QoS-assuring traffic forwarding. While doing so, the transmission of scheduled traffic is guaranteed, predictively received and not disturbed by any best-effort traffic.

A proof-of-concept implementation has verified our implementation to work as intended. It was designed to be in accordance with ns-3's abstraction model and thus versatilely applicable, compatible with other communication standard modules as well as flexible and modular for simulation scenarios using it.

It is planned for future work to utilize the TSN module to create an industrial application analysis environment, where several industrial use cases are simulated. Since reliable and deterministic communication is a key-requirement in such environments, a TSN module was an indispensable prerequisite. Novel ideas to shape the future landscape of industrial production plant communication can finally be developed and evaluated.

ns-3 has the valuable capability to operate not only enclosed in its event base but to also simulate live, connected to and interacting with the real world. For Industry 4.0 (I4.0), wireless communication is a promising topic, not yet reached the market. Especially of high interest is 5G mobile communications. A compatibility of 5G and TSN is envisaged. While 5G is still ongoing standardization and a full market introduction is pending, the OpenAirInterface (OAI) provides faster access to the technology for research purposes. OAI is an open source ecosystem for designing next generation wireless systems and solve next generation wireless challenges, utilizing pure software solutions for e.g. the EPC core network and FPGA code for Software Define Radio (SDR). ns-3's versatility and our implementation allow a simulative TSN approach to be combined with a software controlled 5G system (OAI) to create a flexible and yet realistic research environment.

Acknowledgment

This work is supported by the German *Federal Ministry for Education and Research (BMBF)* within the project »Artificial Intelligence for dynamic Optimization of Network Management (KITOS)« {16KIS1158K}.

References

[1] S. Duque Antón, D. Fraunholz, D. Krummacker, *et al.*, "The Dos and Don'ts of Industrial Network Simulation: A Field Report", International Academy of Science and Engineering for Development (IASED), IASED, 2018.

[2] P. Meyer, T. Steinbach, *et al.*, "Extending IEEE 802.1 AVB with time-triggered scheduling: A simulation study of the coexistence of synchronous and asynchronous traffic", in *2013 IEEE Vehicular Networking Conference*, IEEE, 2013, pp. 47–54.

[3] P. Heise, F. Geyer, and R. Obermaisser, "TSimNet: An industrial time sensitive networking simulation framework based on OMNeT++", in *2016 8th IFIP International Conference on New Technologies, Mobility and Security (NTMS)*, IEEE, 2016, pp. 1–5.

[4] J. Jiang, Y. Li, *et al.*, "A Time-sensitive Networking (TSN) Simulation Model Based on OMNET++", in *2018 IEEE International Conference on Mechatronics and Automation (ICMA)*, 2018, pp. 643–648.

[5] M. Pahlevan and R. Obermaisser, "Evaluation of time-triggered traffic in time-sensitive networks using the opnet simulation framework", in *2018 26th Euromicro International Conference on Parallel, Distributed and Network-based Processing (PDP)*, IEEE, 2018, pp. 283–287.

[6] J. Falk, D. Hellmanns, *et al.*, "NeSTiNg: Simulating IEEE Time-sensitive Networking (TSN) in OMNeT++", in *2019 International Conference on Networked Systems (NetSys)*, 2019, pp. 1–8.

[7] *CPRI "FrontHaul" requirements continuing discussion with TSN*, 2014. [Online]. Available: https://www.ieee802.org/1/files/public/docs2014/new-ashwood-tsn-cpri-fronthaul-1101-v01.pdf.

[8] S. Nsaibi, "Timing Performance Analysis of the Deterministic Ethernet Enhancements Time-Sensitive Networking (TSN) for Use in the Industrial Communication", 2020.

[9] D. Ginthör, R. Guillaume, J. von Hoyningen-Huene, *et al.*, "End-to-end Optimized Joint Scheduling of Converged Wireless and Wired Time-Sensitive Networks", in *2020 25th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, IEEE, vol. 1, 2020, pp. 222–229.

[10] M. Gundall, C. Huber, *et al.*, "Integration of 5G with TSN as Prerequisite for a Highly Flexible Future Industrial Automation: Time Synchronization based on IEEE 802.1 AS", in *IECON 2020 The 46th Annual Conference of the IEEE Industrial Electronics Society*, IEEE, 2020, pp. 3823–3830.

[11] D. Krummacker, C. Fischer, *et al.*, "Intra-Network Clock Synchronization for Wireless Networks: From State of the Art Systems to an Improved Solution", in *2020 2nd International Conference on Computer Communication and the Internet (ICCCI)*, Jun. 2020, pp. 36–44. DOI: 10.1109/ICCCI49374.2020.9145977.

[12] L. Zhao, P. Pop, *et al.*, "Timing analysis of AVB traffic in TSN networks using network calculus", in *2018 IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)*, IEEE, 2018, pp. 25–36.

[13] F. Reimann, S. Graf, *et al.*, "Timing analysis of Ethernet AVB-based automotive E/E architectures", in *2013 IEEE 18th Conference on Emerging Technologies & Factory Automation (ETFA)*, IEEE, 2013, pp. 1–8.

[14] J. Diemer, J. Rox, and R. Ernst, "Modeling of ethernet avb networks for worst-case timing analysis", *IFAC Proceedings Volumes*, vol. 45, no. 2, pp. 848–853, 2012.