TSimNet: An industrial Time Sensitive Networking simulation framework based on OMNeT++

Peter Heise*†, Fabien Geyer* and Roman Obermaisser†
*Airbus Group Innovations, 81663 Munich, Germany
†University of Siegen, 57068 Siegen, Germany
Emails: {peter.heise@airbus.com, roman.obermaisser@uni-siegen.de}

***Abstract*—The upcoming IEEE Time Sensitive Networking standard will offer a set of new functionality for Ethernet like frame preemption and frame replication that need careful evaluation. The IETF DetNet working group also works on deterministic layer 3 segments and leverages therefore works from the IEEE group. We present here a simulation framework based on OMNeT++ that implements all non time-based features present in TSN in a modular and easily extendable way. The framework gives other researches a stable base to start evaluating existing and future scenarios. An evaluation of frame preemption as well as per-stream policing based on the framework is presented. It was shown, that frame preemption does not always bring latency advantages and highly depends on configuration of the system. The paper concludes with an overview of the computational cost for the use of this framework.**

Keywords*—IEEE, Time-Sensitive Networking, TSN, OMNeT++, INET, Simulation, Deterministic, Real-Time, Ethernet, Network

I. Introduction

With increasing digitization more devices are being introduced into complex systems like cars and aircraft. Besides the increase in numbers, also the requirements and bandwidth demands are increasing. A candidate that has been extensively used in avionics that keeps up with these two tends is Ethernet.

Standard Ethernet by itself does not provide any performance bounds and standard extensions such as Avionics Full Duplex Switched Ethernet (AFDX), TTEthernet and IEEE's Audio Video Bridging (AVB). All of these extensions aim to make Ethernet a safety-critical network. A network is considered safety-critical, if the failure of a system could lead to consequences that are determined to be unacceptable [1]. To have one common and open standard for such networks, IEEE itself has started to work on mechanisms and features that achieve deterministic network behavior in the Time-Sensitive Networking (TSN) working group.

While most parts are still in draft status, we present here a network simulation framework for OMNeT++ that implements most stable mechanisms of the drafts and allows to evaluate those new features. We focus here on the industrial use-case which is not based on time-synchronization due to certification reasons explained in section 3.

The remainder of this paper is organized as follows: Section 2 gives background on developments and new features implemented in TSN and summarizes related work. Section 3 motivates why time synchronization is not suitable for avionic networks. Section 4 introduces the concept of our simulation framework and how we extended beyond the standard Ethernet model followed by section 5 that gives details about implementation and short configuration examples to present the capabilities of our framework. Section 6 evaluates the framework on a network level as well as giving an overview of resource and computational needs. Section 7 concludes the paper with a summary.

II. Background

A. Fundamentals

The main task of safety-critical networks is to provide guaranteed delivery of all packets within a certain end-to-end deadline. When speaking about deadline guarantees for packets, often the term deterministic is used. For a network to be called deterministic, it must provide mechanisms to map individual packets into streams and enforce well-defined behavior of such streams even in presence of failure.

First attempts of the IEEE Standards Association to give latency guarantees on Ethernet were done with IEEE Audio Video Bridging (AVB). While it supported some amount individual stream and upper latency bounds it was shown to be not sufficiently gradual for industrial use [2], [3].

Pushed by the automotive industry for a common deterministic network, the TSN working group was founded currently defining features in currently more than 30 separate standard documents[1]. Its main features include more robust time synchronization protocols, time-based scheduling, traffic policing, frame preemption, frame duplication for redundancy reasons and centralized and decentralized configuration protocols. We give here a background on those standards implemented within the framework.

B. Frame Preemption

The finalized document 802.1Qbu called *Frame Preemption* defines means on how to pause the transmission of frames and give way to frames of higher priority. The need for Qbu is mainly tied to time-based scheduling systems, where small high priority control frames are scheduled so frequently that no larger low priority frame would fit anymore therefore offering bandwidth advantages. Another feature of Qbu is the smaller latency and jitter that can be achieved for high priority frames. When a low priority frame gets preempted it is suspended with a 4 byte CRC attachment to the fragment. After the

[1]http://www.ieee802.org/1/files/public/minutes/2016-07-closing-plenary-slides.pdf [Last accessed 2016/09]

high priority frame got transmitted normally, a shortened 6 byte preamble followed by a modified start frame delimiter (SFD) and fragment counter the fragment's data transmission is resumed. No data will be sent twice in this process and it is possible to preempt a frame multiple times as long as the fragments are above 64 bytes in size. Stacked preemption is not possible in Qbu.

C. Frame Replication and Elimination for Reliability (FRER)

Another working draft 802.1CB describes how to duplicate frames for reliability. The transmitting ports insert an EtherTag (see Figure 1) into the frame which is then used at the receiving end for identification and deduplication. A simple deduplication algorithm that remembers the last seen sequence number and admits only higher ones is supported as well as another algorithm that uses a history to also block out of order packets. While this removes purposefully generated duplicates, it also protects from ports that get stuck in repeated transmission mode, also known as *Babbling Idiot*.

Redundancy Tag Format [32 bit]	
EtherType [16 bit]	Sequence Number [16 bit]

Fig. 1. TSN Redundancy CB Tag

D. Per-Stream Filtering and Policing (PSFP)

In another working draft called 802.1Qci per-stream ingress policing is defined. The draft defines a mapping of streams to stream gates, which can be closed or opened according to time-based state machines. If a stream gate is open and allows a stream to pass, it further performs checks on maximum frame size and frame count as well as applying token bucket style traffic policing.

E. Related Work

Several simulation frameworks can be found related to real-time networking. To the best of our knowledge no framework covering TSN has been published yet. Most similar to our framework is the Core4Inet implementation by Steinbach et al. [4] implementing AS6802 (TTEthernet) in OMNeT++. The author used it to evaluate TTEthernet for in-vehicle automotive scenarios.

Further to mention is Kawahara et al. [5], who implemented AVB in an OMNeT++ simulation framework to map CAN messages onto such network and verifies end-to-end latencies of CAN messages over AVB.

Another discrete event-based network simulator is ns-3. Tuohy et al. [6] present a simulation testbed for in-vehicle communication based on AVB. Georg et al. [7] used ns-3 to verify time-critical communication in smart grid networks.

Analytical works compare Ethernet to IEEE AVB and evaluate AVB in industrial environments [2], [8], [9]. It was concluded that AVB does not offer sufficient advantages to replace deterministic bus systems and needs to be reevaluated with the AVB successor TSN.

III. Motivation & Use Case

TSimNet was implemented for industrial use-cases and not anticipated to model time-based algorithms or mechanisms. In a synchronized system, where scheduling decisions are based on a global time, protocols need to ensure continuous synchronization of time. However, regardless of the level of redundancy of the system, clock synchronization inevitably becomes the bottle-neck and will impose system failure when clocks become deskewed. Provably correct fault-tolerant clock synchronization algorithms are presented in [10], which are all based on three or more global clocks. Due to the complexity of these mechanisms, as of now, none of them have been deployed in civil avionics due to certification reasons [11].

While other non time-based mechanisms like AFDX [12] do exist already, we expect a mass market potential of TSN due to the interest of the automotive industry and could therefore be a possible cost saver in future avionic networks while at the same time offer all mechanisms needed for a reliable network.

TSN has a subset of features and mechanisms that can run without time synchronization. Such a subset could be specified in a so-called profile, naming exactly which features need to be implemented to be following that profile. An example of this is AVnu's Ethernet AVB Automotive Certification Profile [13] for IEEE AVB. We envision an industrial TSN profile to include explicit stream forwarding, per-stream filtering, frame replication & recovery and possibly frame preemption.

IV. Concept & Model

TSimNet has been implemented as an extension of the INET framework [14], which itself relies on the OMNeT++ [15] simulation tool. OMNeT++ is a discrete event-based simulator with a focus on communication networks freely available for academic users. The INET framework extends OMNeT++ in terms of protocols and mechanisms used in current Ethernet networks. Our framework TSimNet extends INET and introduces new features present in TSN on top of it.

A. TSimNet Switch Model

The most significant difference between standard Ethernet and TSN is explicit identification of individual streams. Different methods are defined to do this mapping, all based on either MAC address, VLAN or IP address. Based on this stream identifier a TSN enabled switch will run a different set of separate actions like policing operations, recovery based on sequence numbers and explicit forwarding to one or several output ports.

B. TSimNet Host Model

A host does not necessarily need to support any additional features. The stream identifier can be explicitly set, or simply matched at the edge port of the switch. However, in our scenario with an intended ingress policing at every hop, the host needs to shape its outgoing data to follow a predefined traffic characteristic like specified in MEF 10.3 [16].

V. Implementation Details

This section gives implementation details of the main components followed by basic configuration examples. Only some components are described here for space reasons. Please download the software for a full documentation.

A. Stream identification & Encapsulation

A new encapsulation module *TSNEncap* has been introduced that does encapsulation in a demonstrative way, see Figure 2 for the respective modules. First off, each frame gets encapsulated in a VLAN tag, which is then used for proper stream identification. In case FRER is enabled on the outgoing link, the packet is then identified and based on the identification, a redundancy tag (see Figure 1) is added.

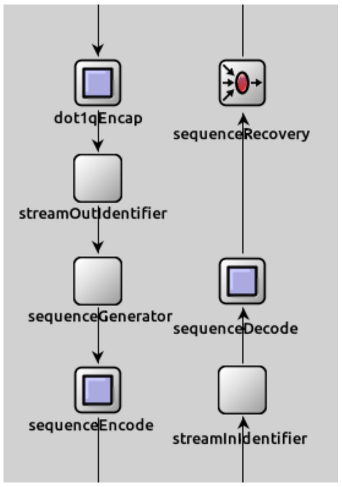

Fig. 2. TSN Stream Identification and Frame Replication Mechanisms

While stream identification is shown here on the outgoing link, the same component is used on the incoming link in a switch. The stream identification module identifies the stream based on selectable algorithms. Implemented currently are mappings from destination MAC and VLAN tag to one stream identifier, source MAC and VLAN tag or a mapping based on the packets source IP address and destination IP address.

B. Per-Stream Filtering and Policing (PSFP)

Inside the MAC the *ingressTC* component that takes care of traffic policing. While the Qci standard has stream gates in place which are controlled by time-based state machines, all filters here are handled by one stream gate that is always set to open at all times to be independent of time synchronization. Such stream gate has a filter table that matches each packet to a stream. Each stream entry has to be specified following the MEF 10.3 [16] definitions with Committed Information Rate (CIR), maximum CIR, Committed Burst Size (CBS), Excess Information Rate (EIR), maximum EIR and a dropOnYellow flag to allow color-aware or color-blind traffic filtering. Further a maximum frame size check is configured. All streams not specified in here get forwarded to the relay unit, where unknown traffic can be dropped or broadcasted.

C. Frame Replication and Elimination for Reliability (FRER)

FRER allows the user to insert an EtherTag into the packet (see Figures 1 and 2) indicating the current sequence number of a stream. Such sequence number is recognized at the next stream recovery element and either checked against a bit-vector (*VectorRecoveryAlgorithm*) or a simple counter if it is a higher sequence number than its predecessor (*MatchRecoveryAlgorithm*). Both mechanisms are implemented in the *TSNRelayUnit* and are configurable according to Listing 1.

```
**.tsnSeqRecHistoryLength      = 4
**.tsnSeqRecoveryAlgorithm     = "VectorRecovery"
```

Listing 1. Frame Replication detailed configuration of *TSNSequenceRecoveryModule*

FRER consists of multiple stages in the relay unit. The first step in FRER is an individual stream recovery with a per-stream filtering at the ingress port before such stream gets admitted into the ingress handler for policing and later forwarding in the relay unit. Once in the relay unit, stream identifiers might get overwritten into a new identifier and by that merged into a single stream. This single merged stream is then subject to cumulative recovery, again based on one of the two previously mentioned recovery algorithms.

D. Non-Standard Additions

The following parts are not part of the TSN standards but help with using our simulation framework.

1) Auto-Configuration Component: The simulation framework requires a lot of parameters to be set which might be error-prone or reduce the usability of the framework. Therefore we extended the configuration component and introduced *TSNConfigurator*. The configurator takes a XML file as input to set the configuration parameters of all nodes, like in *IPv4Configurator* from INET, by introducing a new element: `<tsn-route>`.

For an example see Listing 2 where the tsn-route feature is in use. A path element, not shown, sets the explicit forwarding rules on all switches for this specific stream on its path. Sequence filtering sets individual stream recovery to be applied at each hop of the path while setting traffic parameters for the streaming gates.

This XML file can be used in conjunction with manual configuration in the scenario files, for duplication of traffic across different paths the tsn-route element can be added for the same stream-id.

```
<config>
  <interface hosts='*' address='10.x.x.x'
    netmask='255.x.x.x'/>
  <tsn-route stream-id="1"
    src="host0" dst="host1"
    sequence-filtering="enabled"
    ....
</config>
```

Listing 2. Auto-Configuration config.xml

2) Failure injection: In order to test the new mechanisms, faulty devices and hosts have been added. The first faulty module is the *TSNBabblingIdiotHost*, which keeps on sending an arbitrary amount of data to random destinations besides the traffic it is configured to transmit in normal operation. Using this node we can simulate traffic overload in the network and verify functional traffic policing modules and configuration settings.

The second faulty component is the *TSNRandomSequence-Generator*, which is also included in the aforementioned babbling idiot node. The random sequence generator injects randomly increasing sequence numbers into packets of a particular stream, therefore bypassing the recovery functions of the nodes on the way. In an ill-configured system such duplicates would then be forwarded normally by the relay unit possibly congesting other parts of the network. Such a fault has to be recovered through a traffic policing.

VI. Evaluation & Simulation

In this section we present a first performance evaluation of TSN based on the TSimNet framework and give an overview of the computational cost of our framework.

A. *Simple line congestion*

A first evaluation is looking at the frame preemption capabilities of TSN. We use a simple topology consisting of three nodes like seen in Figure 3, where one node is sending important control traffic to a sink and at the same time a best-effort node is congesting the network with cross-traffic with increasing lineload. Traffic is generated by using random sized packets that transmit at random times, limited by a token bucket shaper that has its token rate set to the indicated lineload.

$$\begin{tikzcd} & \text{Cross-Traffic} \arrow[d] & \\ \text{Flow} \arrow[r] & \text{Switch} \arrow[r] & \text{Sink} \end{tikzcd}$$

Fig. 3. Set-up used for simple line congestion evaluation

Three scenarios are shown in Figure 4, each annotated with its minimum, average and maximum values. The first scenario with no prioritization and no preemption shows typical congestion results with high latencies especially at high load factors. Once prioritization comes into effect, the worst case latency stays lower especially at high load factors. This is due to the fact, that at most one best-effort packet can block the priority packet. In the third scenario we apply different priorities combined with frame preemption. It can be seen, that frame preemption brings down latency for the maximum case and especially on the average case performs well with close to no correlation to the lineload.

We conclude that frame preemption does indeed offer latency advantages over non-preempting scenarios, however, only that of one packet at the cost of adding extra complexity to sender and receiver.

B. *Industrial Line-Topology Preemption*

In a next evaluation we investigate the use of frame preemption in typical industrial line-topology settings. In such a setting, individual devices are usually computational weak and try to minimize extra efforts, therefore backbone traffic is typically scheduled first in order to minimize buffering at the local node. Resembling this, we assume a scenario like seen in Figure 5 for further evaluation, where a flow has to go through three nodes before reaching its destination sink and calculate its worst-case end-to-end latency.

In Figure 6 such latencies for the described path are depicted. The first column represents the latency in an idle store-and-forward scenario, where all queues are free and only transmission delay is taken into account across the switches with a zero processing delay. The second column then represents the worst case option while having no frame preemption enabled, however, prioritization in the outgoing queue, e.g. ring-traffic first. At each hop, a maximum size Ethernet frame might just start transmitting from the local node so that the high priority frame has to queue until the transmission of such is finished, therefore giving a queuing time of one large cross traffic frame at each switch.

Column 3 represents the same scenario as column 2, however, with frame preemption enabled and ring-first scheduling. The value is significantly larger than column 4, where the high priority frame is always prioritized, while in column 3 the high priority frame might be blocked. If in front of a high priority frame is the last part of an already preempted frame, such frame will then start transmitting from the next ring node and cannot be preempted as stacked preemption is not allowed for in the standard. Therefore, if frame preemption is to be used in a backbone-first scenario the latency advantages are not as significant as they could be in a proper configured TSN network.

C. *Computational overhead*

In a last test we checked the overhead of the new functionality in comparison to the INET framework. Table I gives an overview of added events, added CPU time and added memory consumption for the scenario used in Section VI-A with three nodes and one central switch.

Basic TSimNet represents a scenario, where only stream encapsulation, stream identification and explicit forwarding are enabled. Due to the graphical nature of the encapsulation, events are added for each component explaining the increase in events. Similar to this also CPU time increased in this scenario

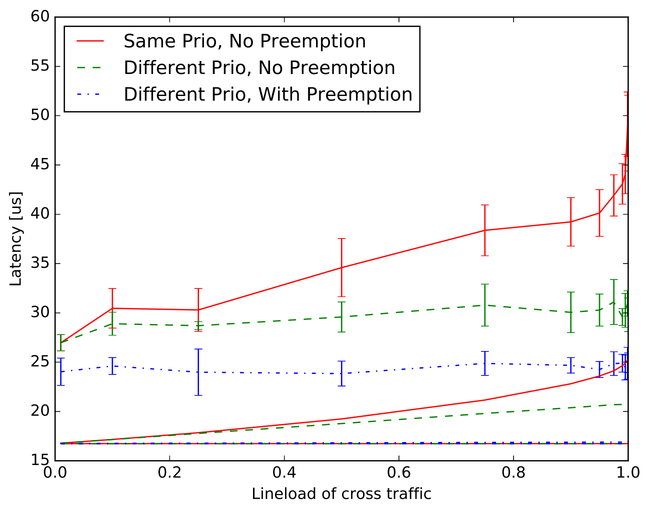

Fig. 4. Comparison of min, max and avg end-to-end latencies with and without Frame Preemption (setup: two nodes; line speed 1Gbps; error bar indicates std deviation)

TABLE I. PERFORMANCE EVALUATION OF TSIMNET SIMULATION FRAMEWORK IN COMPARISON WITH STANDARD INET

	standard INET	**basic TSimNet**	**policing TSimNet**	**recovery TSimNet**	**preemption TSimNet**	**all features TSimNet**
# events [ev$*10^3$]	3799	5401 (**+42%**)	5601 (**+47%**)	5401 (**+42%**)	6268 (**+64%**)	6468 (**+70%**)
CPU time [sec]	20,1	35,54 (**+76%**)	36,55 (**+81%**)	36,04 (**+79%**)	39,28 (**+95%**)	39,72 (**+97%**)
memory [Mbyte]	69	69 (**+0%**)	69 (**+0%**)	69 (**+0%**)	70 (**+1%**)	70 (**+1%**)

$$\begin{tikzcd}
 & \text{Cross-Traffic} \arrow[d] & \text{Cross-Traffic} \arrow[d] & \text{Cross-Traffic} \arrow[d] & \\
\text{Flow} \arrow[r] & \text{Switch 1} \arrow[r] & \text{Switch 2} \arrow[r] & \text{Switch 3} \arrow[r] & \text{Sink}
\end{tikzcd}$$

Fig. 5. Set-up used for industrial line-topology evaluation

to handle all new events. The number of added events will remain a constant factor with increasing numbers of packets.

Traffic policing added a limited number of events, incurred through the extra components in the MAC. Also CPU time increased in order to do the filter look-up and apply policing to the flows. Recovery (FRER) features didn't add any extra events, however, needed extra CPU time to run the recovery algorithms and look-up of sequence history of flows. Frame preemption added extra events, as the fragmented packets get scheduled and retransmitted through the transmission channels. CPU time increased because of that, also memory consumption did go up due to the higher number of packets in the simulation.

With all features enabled, about 70 percent extra events have to be scheduled in for the TSimNet enabled simulation framework, due to the graphical nature some of TSimNetś features were implemented. This value could be reduced at the expense of less readability and no graphical references to the standard documents. Memory consumption stays roughly the same for this simulation as the selected scenario was a small one.

VII. CONCLUSION

In this paper we presented a TSN simulation framework for OMNeT++ that implements all features needed for an industrial profile that is not based on time synchronization. An evaluation of frame preemption as well as per-stream policing based on the framework were presented and it was shown, that frame preemption does not always bring latency advantages and highly depends on configuration of the system. The full source-code as well as documentation is available via http://www.eti.uni-siegen.de. We hope this framework gives means to developers to evaluate new TSN features by open-sourcing this code.

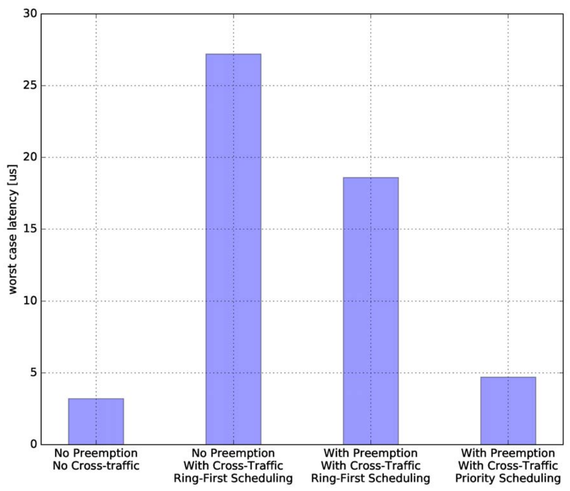

Fig. 6. Comparison of maximum end-to-end latencies in an industrial line scenario with and without frame Preemption (setup: three switches; line speed 1Gbps; control packet size 100B; cross traffic packet size 1000B)

VIII. ACKNOWLEDGMENT

This work was supported in part by the German Federal Ministry of Economics and Technology (BMWi) under the *KASI* project (support codes 20K1306B).

REFERENCES

[1] J. Knight, "Safety Critical Systems: Challenges and Directions," *International Conference on Software Engineering*, 2002.

[2] J. Imtiaz, J. Jasperneite, and L. Han, "A performance study of Ethernet Audio Video Bridging (AVB) for Industrial real-time communication," in *IEEE Conference on Emerging Technologies Factory Automation (ETFA)*, 2009.

[3] E. Heidinger, F. Geyer, S. Schneele, and M. Paulitsch, "A performance study of Audio Video Bridging in aeronautic Ethernet networks," in *7th IEEE International Symposium on Industrial Embedded Systems (SIES)*, 2012.

[4] T. Steinbach, H. D. Kenfack, F. Korf, and T. C. Schmidt, "An Extension of the OMNeT ++ INET Framework for Simulating Real-time Ethernet with High Accuracy," in *4th International ICST Conference on Simulation Tools and Techniques*, 2011.

[5] K. Kawahara, Y. Matsubara, and H. Takada, "A Simulation Environment and preliminary evaluation for Automotive CAN-Ethernet AVB Networks," *arXiv preprint arXiv:1409.0998*, sep 2014.

[6] S. Tuohy, M. Glavin, C. Hughes, E. Jones, and L. Kilmartin, "An ns3 based Simulation Testbed for In-Vehicle Communication Networks," in *27th Annual UK Performance Engineering Workshop*, 2011.

[7] H. Georg, N. Dorsch, M. Putzke, and C. Wietfeld, "Performance evaluation of time-critical communication networks for smart grids based on IEC 61850," in *IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, 2013.

[8] H.-t. Lim, D. Herrscher, and T. Ag, "Performance Comparison of IEEE 802.1Q and IEEE 802.1 AVB in an Ethernet-based In-Vehicle Network," in *8th International Conference on Computing Technology and Information Management (ICCM)*, 2012.

[9] G. Alderisi, G. Patti, and L. L. Bello, "Introducing support for scheduled traffic over IEEE audio video bridging networks," in *18th Conference on Emerging Technologies Factory Automation (ETFA)*, 2013.

[10] R. W. Butler, "Fault-tolerant clock synchronization techniques for avionics systems," in *AIAA/AHS/ASEE Aircraft Design, Systems and Operations Conference*, Atlanta, Georgia, 1988.

[11] I. Moir, A. Seabridge, and M. Jukes, *Civil Avionics Systems*. John Wiley Sons, 2013.

[12] Aeronautical Radio Incorporated, "Aircraft Data Network, Part 7: Avionics Full Duplex Switched Ethernet (AFDX)," 2006.

[13] AVnu, "AVB Automotive Profile."

[14] "INET Framework." [Online]. Available: http://inet.omnetpp.org/

[15] "OMNeT++." [Online]. Available: http://www.omnetpp.org/

[16] MEF Forum, "MEF 10.3 - Ethernet Services Attributes."