An SMT-based Tool for Automatic Schedule Generation for Time Sensitive Networking

Aellison Cassimiro T. dos Santos
Federal University of Paraíba
Paraíba, Brazil
Email: cassimiroaellison@gmail.com

Ben Schneider
fortiss
Munich, Germany
Email: schneider@fortiss.org

Vivek Nigam
fortiss
Munich, Germany
Email: nigam@fortiss.org

***Abstract*—With the increasing demand for fast, reliable and deterministic communication, a group of standards is currently being developed aiming to deliver these requirements to standard Ethernet networks. This set of standards is titled Time Sensitive Networking (TSN), and uses time scheduling to enable deterministic communication. Due to the complexity of the scheduling problem and size of industrial networks, configuring TSN networks is a challenging task, specially if done manually. Using formal methods to model the problem and SMT solvers to obtain valid answers, we present TSNSCHED, a flexible, modular tool capable of automatic generation of schedules for TSN networks. It takes as input the topology of the network and creates the constraints to the Z3 SMT solver. As shown in previous work, we see that TSNSCHED is capable of generating schedules to a number of realistic-size networks.**

I. Introduction

The composite of standards titled Time-Sensitive Networking (TSN), was created with the purpose of offering reliability and determinism to standard IEEE 802.1 and IEEE 802.3 Ethernet networks [6], allowing its usage in time-critical systems. In-vehicle network applications can be mentioned as examples of systems with such requirements, as it is vital that applications such as driver-assistance have its packets delivered with speed and reliability, having in mind the consequences that can be generated by the failure of this system [7].

From these standards, one of the basis of the time-triggered communication paradigm is the IEEE 802.1Qbv, responsible for the management of queues and traffic in switches present on TSN networks. Its implementation works as the Time-Aware Shaper (TAS) of a switch; a module responsible for handling the opening and closing of gates of egress packet queues of each port of a switch. Simply put, with the TAS the TSN technology can assure determinism by controlling the moment of transmission for certain packets at each hop of the communication path. One of the available approaches to do so is by assigning priorities to streams of packets, and, at each hop of the path, the packets are placed in a queue according to their priority. Packets can only be transmitted when the gate of the queue they belong to are open. The points in time where each of the queues' gates open and close are repeated after a fixed amount of time, creating a cyclic behavior for the transmission of the packets. Discovering the size of these cycles, the moment when each of the queues' gates should be open, how much time each gate should remain open, and assigning the priorities to the streams such that the transmission of every packet complies with the maximum latency and jitter allowed per flow are the core of the scheduling problem.

Nigam and Cassimiro have been partially supported by CNPq projects 425870/2016-2 and 303909/2018-8. This project has received funding from the European Unions Horizon 2020 research and innovation programme under grant agreement No 830892.

To exemplify the purpose of this scheduling mechanism, picture two communication flows that share a link in their path, going through the same port of a switch in a vehicle. Flow A carries data from collision sensors to the airbag system, while flow B carries media to displays of passengers. Due to the critical nature of the data sent in flow A, the transmission of its packets must be prioritized over the transmission of other packets. With the TAS, each flow can be assigned to a different priority, guaranteeing that packets from flow B will not consume the bandwidth necessary for the transmission of packets from flow A.

This scheduling problem can easily be solved manually for small topologies, but the difficulty of the problem escalates exponentially, creating a hindrance for the development of feasible automated solutions, given that the problem is NP-complete [3].

In this paper, we present a highly flexible application for automatically generating schedules for TSN networks, assigning priorities to flows and obtaining the values of the size of cycles, in addition to the opening and closing time of the gates. This application is titled TSNSCHED[1].

II. Schedule Generation Process

The core idea behind TSNSCHED lies in the manipulation of the communication streams of the TSN network, also referred to as flows. In order to enhance the control over the possible constraints which can be applied by the scheduler on different nodes of a convergent network, each flow is broken into *flow fragments*, such as it can be seen in Figure 1. The fragments can also be interpreted as an abstraction of a network link in a temporal perspective.

Each flow fragment contains information about the stream of packets in the port of the switch it belongs to. With it, is

[1]TSNSCHED can be found at https://github.com/ACassimiro/TSNsched.

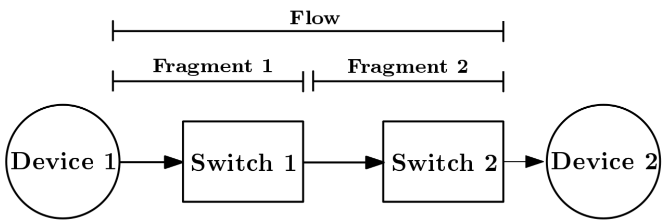

Fig. 1. Illustration of a flow and its fragments.

possible to easily obtain timing values of the packets that go through that port (*i.e.* time of departure of last node, time of arrival in the current node and time of departure from current node) and references to the priority queues to which the packets from this flow were assigned to.

Aside from the usage of flow fragments, TSNSCHED offers a low level of configuration of each node in the network, as ports of a switch can have its parameters configured individually including the constraints used in the scheduling process. This allows for the creation of networks with multiple nodes working under different implemented standards and mechanisms (*i.e.*, a convergent network).

The user builds the topology of the network as input to TSNSCHED by specifying devices, switches and communication flows. Information such as periodicity of packet sending, time to travel between switches, maximum jitter and latency per flow and other properties are specified here. With the topology ready, TSNSCHED proceeds to break the flows into fragments.

Once a flow is broken into fragments and the fragments are assigned to its respective ports, the process of translation of the properties of the network to variables used by the SMT solver begins. With the created variables, each port iterates over its fragments to assert the constraints to be used in the solution of the scheduling process.

Based on the structure and constraints of the specified problem, TSNSCHED queries Z3 for values of the unknown variables. If a solution is found, these values can be used to configure the cycles of each individual port of the switches, finishing the configuration process of the schedule of the network.

III. Z3 Encoding Advantages and Expressiveness

As TSNSCHED was developed in Java using the Z3 Java library, we observe that the union of functionalities offered by an Object Oriented programming language and the problem solution power of an SMT solver creates a flexible, modular and expressive approach to tackle the traffic scheduling problem in TSN networks. Such approach simplifies the usage of formal methods for schedule generation as we eliminate the hindrance of manually formally specifying the whole network.

With the basis offered by TSNSCHED, for example, we were able to handle the remaining non-priority time windows for best effort traffic, avoiding its starvation. This allows for the maintenance of the QoS for non time-critical applications which share ports with priority traffic.

The modular design created by the usage of flow fragments also provides a simple process of implementation and modification of constraints, resulting in a concise translation from constraints of the formal model on a link level. A consequence of this fact is the ease of implementation of different communication structures such as unicast and multicast flows, aside from the capacity of implementation of problem variants, such as the ones explored in [3], [4] and [5].

With TSNSCHED, we also provide the control of the constraints used on a port to the user. This allows the usage of different scheduling. An example of this is the capacity of integration of the Per-Stream Filtering and Policing standard (IEEE 802.1Qci) together with the TAS (IEEE 802.1Qbv), which belongs to another standard. This slightly modifies the scope of the scheduling problem, as packets from a stream can have different priorities in different hops of its path [2].

Lastly, we prove in previous work that TSNSCHED is capable of generating schedules to realistic sized industrial applications in [1], being able to generate schedules with latency and jitter requirements for networks with up to 73 subscribers and 10 multicast flows in the explored experiments. We also analyze the execution time of the tool, where it is possible to see the exponential behavior resulted by scaling an NP-Complete problem. While this might be an issue for the tool, TSNSCHED scales well enough to generate schedules for realitic-sized industrial networks.

IV. Future Work and Conclusion

As future work, we believe that TSNSCHED's flexibility and configuration level gives us the necessary grounds to explore new approaches for the improvement of schedule generation for TSN. We also aim to implement other mechanisms which compose the TSN standards, giving a broader set of possible configurations for convergent networks.

Currently, we are studying the integration of TSNSCHED into the 4diac framework, as we believe that the tool already has the necessary interfaces to enable an automatic deduction of network topology and scheduling requirements, as well the adaptation of TSNSCHED in order generate outputs ready for deployment to TSN switches.

References

[1] A. Cassimiro T. dos Santos, B. Schneider, and V. Nigam. Tsnsched: Automated schedule generation for time sensitive networking. In *Formal Methods in Computer-Aided Design (FMCAD)*, 2019.

[2] S. S. Craciunas, R. S. Oliver, and T. C. AG. An overview of scheduling mechanisms for time-sensitive networks. *Proceedings of the Real-time summer school LÉcole dÉté Temps Réel (ETR)*, 2017.

[3] S. S. Craciunas, R. S. Oliver, M. Chmelík, and W. Steiner. Scheduling real-time communication in ieee 802.1 qbv time sensitive networks. In *Proceedings of the 24th International Conference on Real-Time Networks and Systems*, pages 183–192. ACM, 2016.

[4] S. S. Craciunas, R. S. Oliver, and W. Steiner. Formal scheduling constraints for time-sensitive networks. *CoRR*, abs/1712.02246, 2017.

[5] S. S. Craciunas, R. S. Oliver, and W. Steiner. Demo abstract: Slate xns–an online management tool for deterministic tsn networks. In *2018 IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)*, pages 103–104, April 2018.

[6] R. Hummen, S. Kehrer, and O. Kleineberg. White paper: Tsn-time sensitive networking. *Belden, St. Louis, MI, USA, Tech. Rep*, 2017.

[7] S. Thangamuthu, N. Concer, P. J. Cuijpers, and J. J. Lukkien. Analysis of ethernet-switch traffic shapers for in-vehicle networking applications. In *Proceedings of the 2015 Design, Automation & Test in Europe Conference & Exhibition*, pages 55–60. EDA Consortium, 2015.