2021 22nd IEEE International Conference on Industrial Technology (ICIT) | 978-1-7281-5730-6/20/$31.00 ©2021 IEEE | DOI: 10.1109/ICIT46573.2021.9453628

An Automated Configuration Framework for TSN Networks

Bahar Houtan, Albert Bergström, Mohammad Ashjaei, Masoud Daneshtalab, Mikael Sjödin, Saad Mubeen
Mälardalen University, Sweden
{bahar.houtan, mohammad.ashjaei, masoud.daneshtalab, mikael.sjodin, saad.mubeen}@mdh.se

***Abstract*—Designing and simulating large networks, based on the Time Sensitive Networking (TSN) standards, require complex and demanding configuration at the design and pre-simulation phases. Existing configuration and simulation frameworks support only manual configuration of TSN networks. This hampers the applicability of these frameworks to large-sized TSN networks, especially in complex industrial embedded system applications. This paper proposes a modular framework to automate offline scheduling in TSN networks to facilitate the design-time and pre-simulation automated network configurations as well as interpretation of the simulations. To demonstrate and evaluate the applicability of the proposed framework, a large TSN network is automatically configured and its performance is evaluated by measuring end-to-end delays of time-critical flows in a state-of-the-art simulation framework, namely NeSTiNg.**

I. Introduction

Time-Sensitive Networking (TSN) is a set of standards, developed by the IEEE 802.1 TSN task group [1], to support high-bandwidth, time-critical, and low-latency communication over the switched Ethernet. These standards offer several novel features, e.g., a common notion of time through clock synchronization, resource reservation for various types of traffic, traffic shaping, scheduled traffic support, frame preemption, and much more. There is a huge interest in utilizing TSN in time-critical applications, in particular in the automotive [2] and industrial automation domains [3]. However, there are several challenges that are encountered when utilizing TSN in industrial applications. One core challenge is to perform the TSN network configuration during the design, analysis and simulation phases, while taking the applications' timing requirements into account. These applications often require complex configuration measures at the design and pre-simulation phases, such as creating offline schedules for the scheduled traffic. The existing network design and simulation frameworks for TSN lack automation in the network configuration. Manually configuring a large number of parameters associated to various types of traffic in TSN can be time consuming, error prone and cumbersome for the network designers.

To address this challenge, this paper proposes an automated framework that facilitates the design-time and pre-simulation network configuration of TSN networks. The proposed framework uses one of the state-of-the-art open-source TSN simulation platforms, namely NeSTiNg [4], [5]. The source code for the proposed framework is openly provided in gitlab[1]. The proposed framework is desinged in a modular way and it uses the Extensible Markup Language (XML) format for the information exchange, which allow the framework to be easily adapted to other simulation platforms. The main contributions in the proposed framework are as follows:

- integrating the traffic generation and optimized network schedule synthesis to the NeSTiNg open-source simulation framework for TSN;
- addressing the flow configuration complexity by automatically translating configuration for the scheduled flows' into the NeSTiNg syntax compliant configuration;
- addressing the gate states configuration complexity by automatically translating the synthesized schedules into the NeSTiNg syntax compliant configuration; and
- finally, allowing to automatically configure the configuration files by a Graphical User Interface (GUI) that is integrated into the simulation framework.

II. Background and Related Work

The Time-Aware Shaper (TAS) mechanism in the IEEE 802.1Qbv standard allows arbitration of various types of traffic based on priorities at a TSN switch's egress port. TSN composes eight traffic priorities that are encoded by a 3-bit priority code point (PCP), which is associated with the index of 8 priority queues at TSN switch's egress port. For example, to reserve the switch bandwidth for the highest priority real-time class, the Scheduled Traffic (ST) with $PCP = 7$, a time-division multiple access transmission selection algorithm is used. Egress data flow from each priority queue is controlled by a gate. Based on this algorithm, the offline schedules must be time-stamped in a Gate Control List (GCL), which contains the starting and finishing times of allowed time slots for each queue and associated configuration of the gate states (GS).

There are several works that discuss automatic synthesis of optimized gate schedules in TSN by using solvers like the Satisfiability/Optimized Modulo Theorem (S/OMT), e.g., the works by Craciunas et al. [6], Hashemi et al. [7], Schneider and Santos et al. [8], and Gavrilut et al. [9]. The work by Pahlevan et al [10] uses the Genetic Algorithm to automatically synthesize the gate schedules. There are several simulation frameworks for pre-implementation evaluation of TSN networks. OMNeT++ is an open-source discrete event simulation platform, primarily used for constructing simulations of networks[2]. INET[3] is another widely used open-source simulation

[1]https://gitlab.com/Scipsybee/automated-tsn-configuration-plugin

[2]https://omnetpp.org/

[3]https://inet.omnetpp.org/

framework that models wired, wireless and mobile networks. Core4INET [11] is an open-source TSN simulation framework that is built on OMNeT++. Core4INET allows simulation of TSN networks based on various standards, e.g., IEEE 802.1Q, IEEE P802.1p VLANs and Priorities, IEEE 802.1 AVB and TTEthernet (AS6802). Another notable example of the TSN simulation frameworks is NeSTiNg [4]. NeSTiNg supports several TSN features, inlcuding the scheduled traffic (IEEE 802.1Qbv), frame preemption (IEEE Std 802.1Qbu and IEEE Std 802.3br), credit-based shaper (IEEE Std 802.1Qav), time synchronization (IEEE Std 802.1AS). Schedule and routing in NesTiNg can be done through insertion of XML files, with a strict syntax. This feature allows NesTiNg to be seamlessly integrated to the proposed framework. Hence, the network simulations with NesTiNg can be facilitated by automatic generation of XML configuration files for flow schedules, flow attributes and gate states. Both Core4INET and NeSTiNg require significant amount of manual configurations at the pre-simulation phase as well as for interpretation of the simulation results. The framework proposed in this paper augments automation in these frameworks by means of an automated configuration framework for TSN.

III. Automated Configuration Framework

A. Configuration Complexity in TSN

This subsection demonstrates the level of complexity in the configuration of TSN networks, in particular, the configuration that is required to simulate TSN networks using the NeSTiNg framework. We demonstrate various configuration parameters by an example shown in Fig. 1. The example illustrates a TSN network consisting of two transmitter end stations, **ST1** and **ST2**, and two receiver end stations, **ST3** and **ST4**. The end stations are connected via two TSN switches, **SW1** and **SW2**. **ST1** transmits a periodic flow to **ST3**. Whereas, **ST2** sends a periodic flow to **ST4**. The period of **ST2**'s flow is double the period of **ST1**'s flow.

The network configuration is performed in three steps: (i) configuration of offline schedules for the scheduled traffic on each link, (ii) configuration of gate states in egress queues of each TSN switch port, and (iii) configuration of switch routing tables.

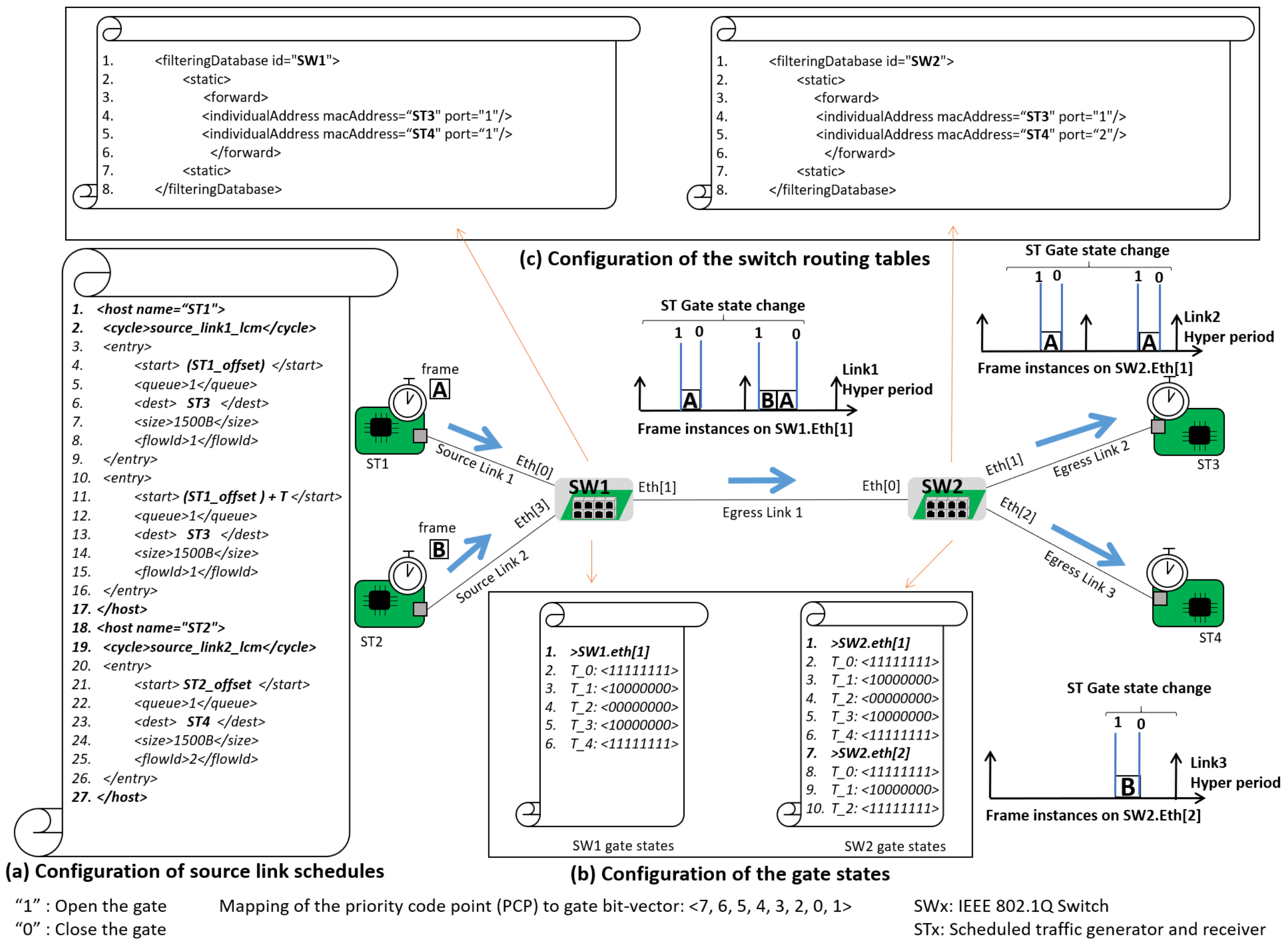
1) Configuration of the Source Link Schedules: In the example shown in Fig. 1, source link schedules are specified by the text in the box (a). In general, there are 27 lines in the flow configuration file. Subsequently, 14 and 7 lines of code are needed to indicate the traffic characteristics of the data flows transmitted from **ST1** and **ST2**. One drawback of the current configuration method is that the traffic settings should be assigned based on the flow generating source end stations. This method becomes challenging when a source end station needs to transmit scheduled traffic with different periods, different data size of flows and via different queues to different destination end stations.

The NeSTiNg configuration approach does not support convenient modelling for TSN flows. In the current modelling perspective, traffic flows must be specified based on the flow generating source end station. Hence, the flow schedules from each source end station should be specified based on the flow offsets in chronological order in an XML file. The bottleneck in this method is the extra effort for the network designers at the traffic configuration stage. Because in case of having different flow profiles from a source end station, it is necessary to add each periodic instance of every flow[4] until *hyper period* of the flows as a new schedule entry to the XML configuration file. The hyper period of the flows is the least common multiple of periods of all flows. It is extremely laborious, time-consuming and error-prone to perform manual configuration of a large number of activation times of the flow instances, even in the case of small networks.

2) Network Gate State Configuration: In order to guarantee timely transmission of the ST flows in the network, the gate opening and closing times in the egress ports of each TSN switch must be assigned, while taking the timing requirements on the flows into account. The TAS provides a mechanism of opening and closing gates of the egress queues, which creates reserved time slots on the egress link. These time slots assign the link capacity to the flows, which are scheduled to pass through the egress link. There are 8 traffic classes that correspond to a gate in the TAS. In order to calculate the reserved time slots on the link, the trace of all the frames passing through the link must be arranged and defined as gate states at the egress port of the TSN switch connected to that link. To specify the ST frame transmission slots on the link, it is necessary to calculate the offset of the frames until the hyper period of the crossing flows from the link. Assigning the gate states can become very complex due to increase in the number of slots that need to be reserved on the link. Conventionally, the gate states can be assigned by manually drawing the network trace and inspecting the arrival times of the ST flows on each link. The box (b) in Fig. 1 shows a sample trace of the two flows on egress link 1 from **SW1** and egress links 2 and 3 from **SW2**. The gate states in the example shown in Fig. 1 can be manually configured by drawing the traces of the network. However, manually performing the gate-states configurations for large-sized networks that include a large number of flows of different traffic classes is impractical, time-consuming and error prone. Even if the schedules for the flows are calculated using optimized schedule generation engines, manual translation of the generated schedules to comply with the input configuration syntax for the simulation frameworks (like NeSTiNg) requires a massive pre-simulation effort.

3) Switch Routing Table Configuration: A routing table statically defines the paths to forward traffic from the source end station to the destination end station's address. The box (c) in Fig. 1 shows the routing table configurations. In case of larger networks, there might be multiple paths to a destination end station, which calls for configuring the routing table of each switch. This requires the designer to acquire knowledge of all the connections between switches and switch port numbers connected to each end station.

[4]A flow instance is represented by a frame



(c) Configuration of the switch routing tables
(a) Configuration of source link schedules
(b) Configuration of the gate states
SW1 gate states
SW2 gate states
ST Gate state change
1 0 1 0
Link1
Hyper period
Frame instances on SW1.Eth[1]
Link2
Hyper period
Frame instances on SW2.Eth[1]
Link3
Hyper period
Frame instances on SW2.Eth[2]
frame
A
B
ST1
ST2
ST3
ST4
SW1
SW2
Source Link 1
Source Link 2
Egress Link 1
Egress Link 2
Egress Link 3
Eth[0]
Eth[1]
Eth[2]
Eth[3]
"1" : Open the gate
"0" : Close the gate
Mapping of the priority code point (PCP) to gate bit-vector: <7, 6, 5, 4, 3, 2, 0, 1>
SWx: IEEE 802.1Q Switch
STx: Scheduled traffic generator and receiver

```
<filteringDatabase id="SW1">
    <static>
        <forward>
            <individualAddress macAddress="ST3" port="1"/>
            <individualAddress macAddress="ST4" port="1"/>
        </forward>
    <static>
</filteringDatabase>
```

```
<filteringDatabase id="SW2">
    <static>
        <forward>
            <individualAddress macAddress="ST3" port="1"/>
            <individualAddress macAddress="ST4" port="2"/>
        </forward>
    <static>
</filteringDatabase>
```

```
<host name="ST1">
 <cycle>source_link1_lcm</cycle>
 <entry>
        <start> (ST1_offset) </start>
        <queue>1</queue>
        <dest>  ST3  </dest>
        <size>1500B</size>
        <flowId>1</flowId>
 </entry>
 <entry>
        <start> (ST1_offset ) + T </start>
        <queue>1</queue>
        <dest>  ST3  </dest>
        <size>1500B</size>
        <flowId>1</flowId>
 </entry>
</host>
<host name="ST2">
 <cycle>source_link2_lcm</cycle>
 <entry>
        <start> ST2_offset </start>
        <queue>1</queue>
        <dest>  ST4 </dest>
        <size>1500B</size>
        <flowId>2</flowId>
 </entry>
</host>
```

```
>SW1.eth[1]
T_0: <11111111>
T_1: <10000000>
T_2: <00000000>
T_3: <10000000>
T_4: <11111111>
```

```
>SW2.eth[1]
T_0: <11111111>
T_1: <10000000>
T_2: <00000000>
T_3: <10000000>
T_4: <11111111>
>SW2.eth[2]
T_0: <11111111>
T_1: <10000000>
T_2: <11111111>
```

Fig. 1: Various sources of the configuration complexity.

It can be seen from the above discussion that simulation of TSN networks requires complex pre-simulation configurations of various parameters. In the case of large TSN networks, manually setting up these configurations becomes time consuming, error prone and impractical. Therefore, an automatic configuration framework is essential, in particular for designing, simulating and evaluating large industrial network.

B. *Modular Architecture of the Proposed Framework*

This subsection presents the modular architecture of the proposed automatic configuration framework as depicted in Fig. 2. The proposed framework consists of four modules: (i) traffic generator, (ii) schedule synthesizer, (iii) automated configurator, and (iv) output and results interpreter.

1) Traffic Generator: The traffic generator module generates attributes of the TSN flows that are required for traffic configuration in the simulator. This module provides flow-based traffic configuration input to the simulator. In this module, periodic flows (a.k.a. streams) are represented by a tuple, S. Each flow s_i, in the set of flows, is represented by Eq. (1):

$$S := \langle \{s_1, \ldots, s_{|S|}\}, \forall [V_a, V_b] \in L : hp_{[V_a,V_b]} \rangle \qquad (1)$$

where, V_a and V_b specify a pair of end stations/switches connected via a link denoted by $[V_a, V_b]$, which is a member of the network link set, denoted by L. The parameter, $hp_{[V_a,V_b]}$, represents the hyper period of the set of flows, which is the least common multiple of periods of flows crossing the link $[V_a, V_b]$. Furthermore, the flow profiles are characterized by Eq. (2):

$$s_i := \langle src_i, t_i, d_i, l_i, p_i, r_i, dest_i \rangle \qquad (2)$$

where, i is the unique ID of each flow. The transmitter end station's ID is represented by src_i. The source end station transmits flows of data to the destination end station with the ID $dest_i$. The period and deadline of the flow are represented by t_i and d_i, respectively. The parameter l_i stores the transmission duration of the flow. The flow priority is denoted by p_i. Finally, r_i holds the list of switches in the path from the source end station to the destination end station.

In addition to setting the flow attributes, this module enables

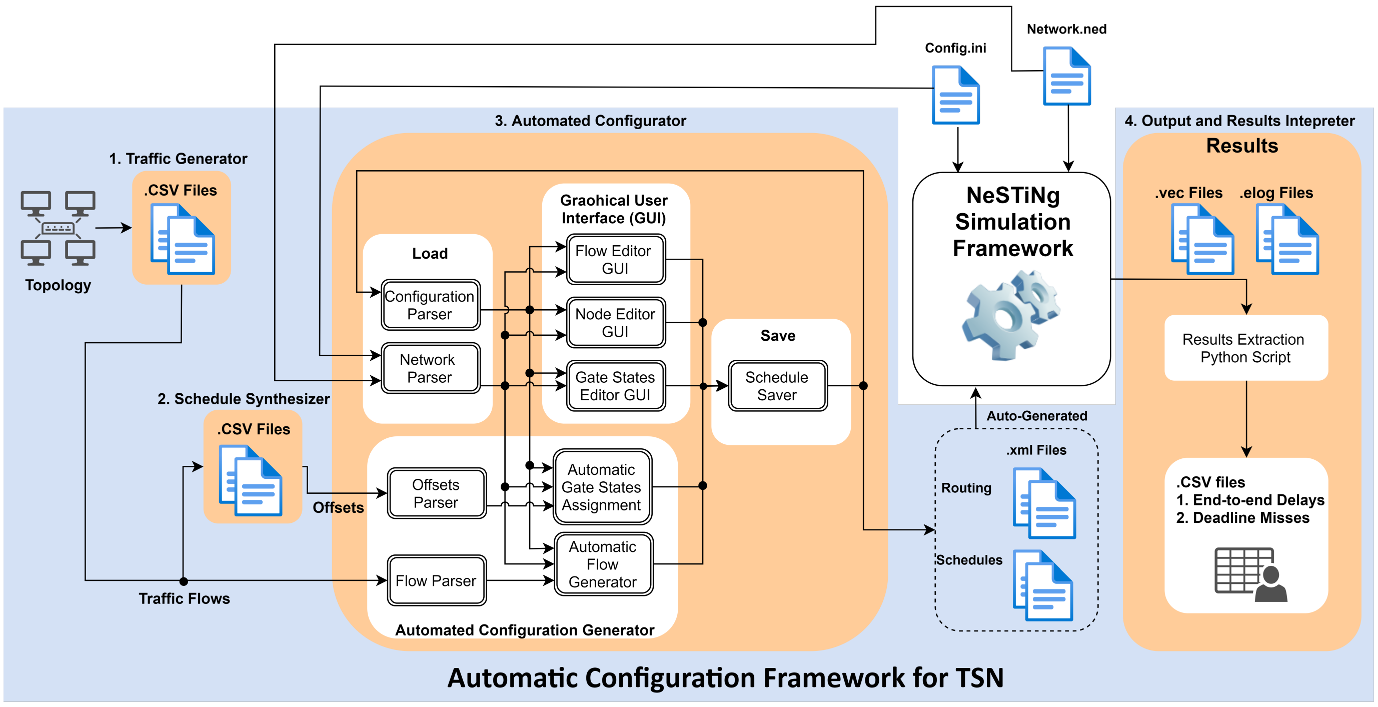


Fig. 2: Modular Design of the proposed automated configuration framework.

editing of the link attributes, which affect the transmission duration of frames and synchronization of the transmitter and receiver end stations. Eq. (3) specifies the link attributes.

$$\forall [V_a, V_b] \in L := \langle speed, delay, tick, queue \rangle \tag{3}$$

The parameter, *speed*, indicates the maximum allowed bandwidth on the link in bit/s. The propagation delay on the link is denoted by *delay*. *tick* is the synchronization factor of the physical layer, according to IEEE 802.1AS. Finally, the maximum number of queues is represented by *queue*. The traffic generator module receives a graph topology of the network. The outputs of this module are flow profiles that are stored in a Comma Separated Vector (CSV) file to be used by the other framework modules.

2) Schedule Synthesizer: The schedule synthesizer is a constraint-based solver that is intended to find solution of a scheduling optimization objective function and a set of real-time constraints on the ST flows. The schedule synthesis module produces offsets that ensure the timing requirements of the time-triggered ST traffic at each hop in the flow's path are satisfied. We specifically employ an open-source S/OMT solver, namely Z3[5] that features both satisfiability and optimization solvers. These solvers are automated theorem provers that search for solutions by examining the satisfiability of possible combination of solutions in a search tree. This module receives a CSV file containing the standard flow set and link set specifications presented by Eq. (1), Eq. (2) and Eq. (3). The schedule synthesis module specifies transmission time of each ST flow on each link. In this work, we apply the network modelling approach presented in [6]. Where, frame i in the k^{th} flow is indicated by $f_{i,k}$. The path from a transmitter end station, V_a, to a receiver end station, V_b, is shown as the set, $[[V_a, V_1], \ [V_1, V_2], ..., [V_i, V_j], ..., [V_{n-2}, V_{n-1}], [V_n, V_b]]$. The transmission gate schedules on each link must be defined by a set of well-defined tuples to be readable by the automatic configuration generation module. Therefore, we consider the following attributes for each frame $f_{i,k}^{[V_i,V_j]}$, $< \phi, \ Len >$. Where, ϕ represents the transmission time, and Len indicates the allowed margin on the link for complete transmissions of the frame. As a results, the list of gate transaction time stamps of each TSN switch is shown in Eq. (4):

$$\begin{aligned} \forall [V_a, V_b] \in L, \forall s_{\{1,...,|S|\}} \in S, \forall f_{i,k}^{[V_a,V_b]} \in s_i^{[V_a,V_b]} : \\ [f_{i,k}^{[V_a,V_b]}.\phi, f_{i,k}^{[V_a,V_b]}.\phi + f_{i,k}^{[V_a,V_b]}.Len] \end{aligned} \tag{4}$$

where, the list of gate states at a switch's egress port is defined by the set of allowed time margins to transmit frames on the associated egress link. The start and length of the time margins are denoted by the flow's frame offset and a predefined window size respectively.

3) Automated Configurator: The automated configuration module is realized as a plug-in for the OMNET++ TSN simulation framework. Hence, it is easliy integrable with the NeSTiNg simulation framework. The plug-in consists of two main sub-modules.

- Graphical User Interface (GUI): Through the GUI sub-module, the flows, schedules, routing settings and gate states

[5]https://pypi.org/project/z3-solver/

can be created and modified manually.

- Automated Configuration Generator: This sub-module receives the simulation setting files from the traffic generation and schedule synthesis modules. The settings are then translated into the NeSTiNg configuration files.

The NeSTiNg XML schedule and routing configuration files can be edited both by the GUI and Automated Configuration Generator. If the XML files contain flow entries, they can be parsed by the load sub-module, which can be further edited and visualized in the GUI. Also, new entries can be written on the files through the GUI or the Automated Configuration Generator. The save sub-module translates the received configurations into the NeSTiNg compliant configurations.

4) Output and Results Interpreter: While it is possible to collect certain useful statistics (e.g., *end-to-end delays*) in the NeSTiNg framework, the manual inspection of numerous recorded parameters for each component in large networks is impractical. The end-to-end delay is a time interval from the transmission of a frame from its source end station until it has arrived at the destination end station. Besides, OMNeT++ and NeSTiNg simulation records are frame-based as they use source-based traffic configuration. Therefore, these simulation frameworks lack a built-in functionality for extraction of the TSN flow metrics. Python scripts are suggested by OMNeT++ official technical guide in [12] to retrieve insights from the simulator's vector (VEC) files. In order to acquire end-to-end delays of the frames, a python script is devised, which constitutes the final sub-module of the framework as shown in Fig. 2. The script traces the vector files, records the transmission and reception times of each flow, and calculates the flow's end-to-end delay. In addition to the aforementioned python script, the final module in the proposed framework also features an OMNeT++ integrated sub-module to extract the maximum end-to-end delays of the flows. The flow extraction component of the plug-in also takes the VEC files containing the transmission and reception times of the flow as input. Thereafter, a CSV file is created with the maximum end-to-end delays corresponding to each TSN flow.

IV. Implementation

This section presents a proof-of-concept implementation of the proposed framework. As shown in Fig. 3, the automated configuration module performs configurations both in manual and automatic mode. There are several configuration algorithms that can be utilized to obtain an optimum network timing behaviour. The proposed framework is implemented as a TSN plugin, which is integrated to the NeSTiNg simulator using the OMNeT++ Java application programming interface and the plug-in development environment in OMNeT++.

A. Framework Configuration Files

1) Pre-simulation Configuration: There is a set of files to be configured before starting the simulation process. The automatic configuration of the pre-simulation process is performed by configuring the following model files.

- Simulation initialization (INI) file: this file contains simulator specific configuration options. This file can be either edited directly or modified in a wizard that is embedded in the OMNeT++ IDE [13]. Our framework uses this file to obtain the Media Access Control (MAC) addresses of the end stations and processing delays in TSN switches.
- Network Topology Description (NED) file: the component modules, sub-modules, channels and overloaded component types can be defined in this file. The OMNeT++ IDE allows setting this file in visual or non-visual modes [13]. Our framework reads the NED files and obtains information such as network topology and available paths from the source end station to the destination end station.
- XML schedule file: transmission schedules for ST flows must be defined and inserted in the NeSTiNg simulator using XML files. Our proposed framework enables automated schedule generation, editing of schedules by a GUI and visualization of each frame in a flow by accessing this file.
- XML routing file: this file statically defines the paths to forward traffic according to the destination end station's address. The XML routing file defines the forwarding database for each switch. By using information from the NED file, such as topology and available paths from the source end station to the destination end station, the framework specifies fixed routes, thereby generating the XML routing file.
- CSV flow file: the set of data received from the traffic generator module is saved in this file. The file contains flows' profiles such as queue index, frame size and destination end station, which are used in the automated mode to further configure the ST flows in the XML schedule file.
- CSV gate synthesis file: this file contains the set of data received from the schedule synthesizer, including offset of frames on each egress link. Accordingly, this file can be used in the automated mode to set the switch gate states in the XML schedule file.

2) Post-simulation Files: The NeSTiNg simulator generates vector outputs of the network event parameters and their occurrence times after the simulation. These files include:

- Results' vectors (VEC): after the simulation, the events and time stamps are recorded in the vector files. The proposed framework's post-simulation analysis module uses these files to obtain end-to-end delay and deadline miss metrics.
- Event logs file (elog): This file stores the frame transmission traces that can be visualized in OMNeT++ by the GUI. We refer the reader to [5] for further details about the elog files.
- End-to-end delay and deadline miss log (CSV): this file contains interpreted results using the simulator's VEC files. This file includes the end-to-end delays and the number of deadline misses experienced by the flows.

B. Configuration of Inputs and GUI Layout

The automatic configuration process supports two modes: (i) manual; and (ii) automatic. Fig. 3 demonstrates the flowchart of the manual and automated configuration input methods. In the manual mode, the GUI allows the user to select input files from the work space. The manual mode requires the

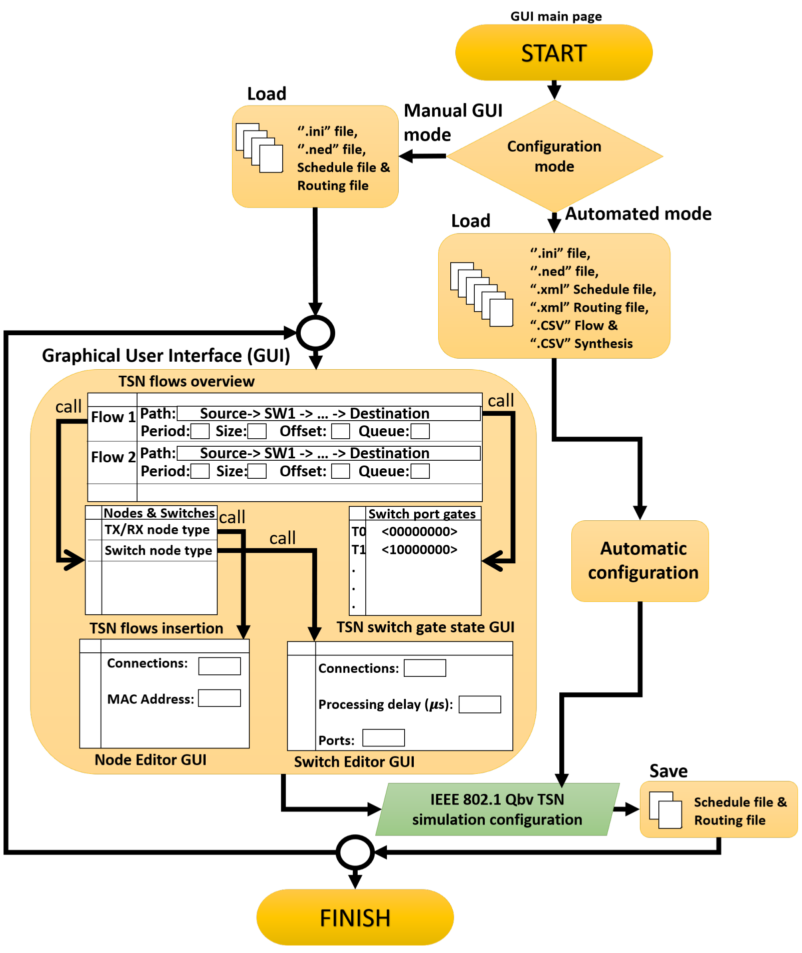

GUI main page
START
Configuration mode
Manual GUI mode
Load
".ini" file, ".ned" file, Schedule file & Routing file
Automated mode
Load
".ini" file, ".ned" file, ".xml" Schedule file, ".xml" Routing file, ".CSV" Flow & ".CSV" Synthesis
Graphical User Interface (GUI)
TSN flows overview
call
Flow 1 Path: Source-> SW1 -> ... -> Destination
Period: Size: Offset: Queue:
Flow 2 Path: Source-> SW1 -> ... -> Destination
Period: Size: Offset: Queue:
call
Nodes & Switches
TX/RX node type
Switch node type
call
call
Switch port gates
T0 <00000000>
T1 <10000000>
TSN flows insertion
TSN switch gate state GUI
Connections:
MAC Address:
Node Editor GUI
Connections:
Processing delay (μs):
Ports:
Switch Editor GUI
Automatic configuration
IEEE 802.1 Qbv TSN simulation configuration
Save
Schedule file & Routing file
FINISH

Fig. 3: Automated flow and gate configurator flowchart.

NeSTiNg's INI and NED files, as well as two empty XML files to be loaded in the fields; "INI file", "NED file", "XML schedule file" and "XML routing file". Manual entries for schedule configurations are saved in the associated XML file. The routing database is saved in the XML routing file based on existing physical channels, as described in the NED file. The GUI allows to append the flow path entries through the "TSN flows insertion" function, which enables to select the source and destination end stations, as well as the specification of static route among the switches within the path from the source end station to the destination end station. After specifying the flow path, other flow attributes can be viewed and adjusted in the "TSN flows overview" function. These attributes include offset, flow size and a dedicated switch queue for the flow. Furthermore, physical characteristics of end stations and switches can be adjusted by calling the "Node Editor GUI" and "Switch Editor GUI" functions. The "TSN switch gate state GUI" function enables inserting the gate states for each port in a TSN switch. Besides, it assists to adjust and modify the gate configurations by visualizing the gate states specified for a frame scheduled on each link. The results of the manual GUI are finally saved in the XML files that are used by the simulation framework.

In the automated configuration mode, insertion of the flow and gate-state configurations can be performed automatically by loading well-defined flow entry and gate-state data files, which must be provided by ".CSV flows" and ".CSV Synthesis" fields in the main GUI along with the rest of the configuration files. We note here that the automated configuration data are prepared by preceding automated sub-modules: traffic generator and schedule synthesizer. Therefore, the automated mode removes the risk of human errors that can be introduced with manual configurations. After the automatic generation of the schedules and routing configurations, the output can be visualized, inspected and adjusted by the manual GUI. The save sub-module translates configurations to the NeSTiNg compatible syntax. We refer the reader to [14] for further details about the implementation and user manual of the GUI. If the proposed framework is to be integrated to a simulation platform other than NeSTiNg, the save sub-module can be adapted to comply with the syntax of the simulation platform.

V. Evaluation

To evaluate the proposed framework we designed and configured a TSN network as illustrated in Fig. 4, which is a hybrid ring-mesh topology connecting 12 end stations through 5 TSN switches. The network speed on each link is set to 1 Gbit/s. We assume that there are no processing delays in the TSN switches. Furthermore, the delays on the links themselves are considered negligible. We generated 40 flows of the scheduled traffic class in TSN. The periods of the flows are chosen from the set [1000 μs, 2000 μs, 5000 μs], which complies with the set of recommended periods in industrial automotive systems, as presented in the real-world automotive benchmarks [15]. We assume implicit deadlines for the flows, i.e., the deadline of each flow is equal to its period. The length of each frame in all the flows is considered to be equal to the maximum size of the Ethernet frame, i.e., 1542 Bytes. The use case topology is input to the schedule synthesis module that implements a schedule synthesis algorithm presented in [6]. This module is executed for approximately 20 hours on an HP Elite Book 820 running Ubuntu OS 18.04.4 LTS with CPU Core i5, 4 * 2.20 GHz Cores and 16 GB RAM.

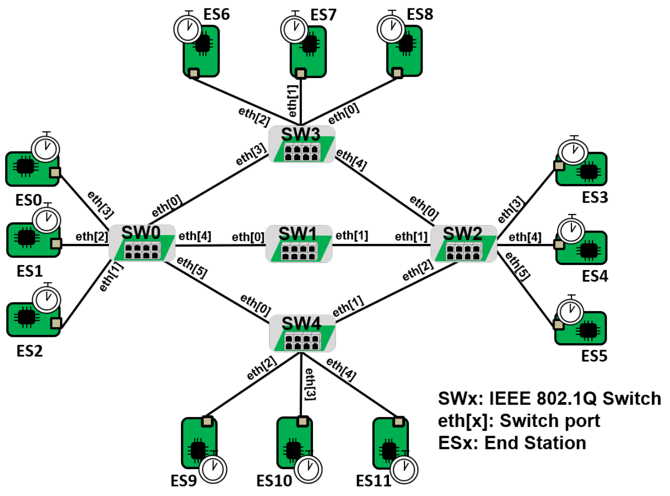
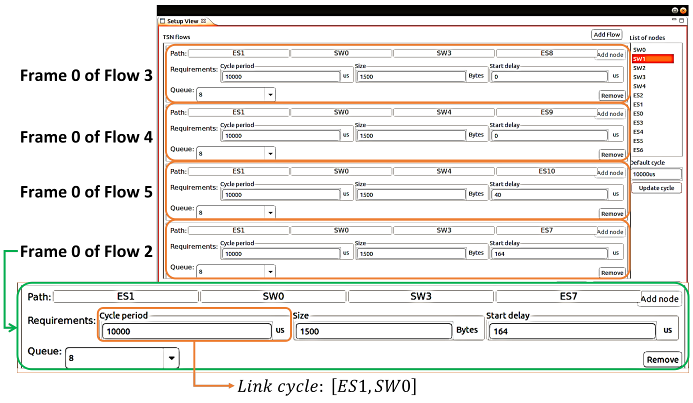


Fig. 4: The use-case topology.

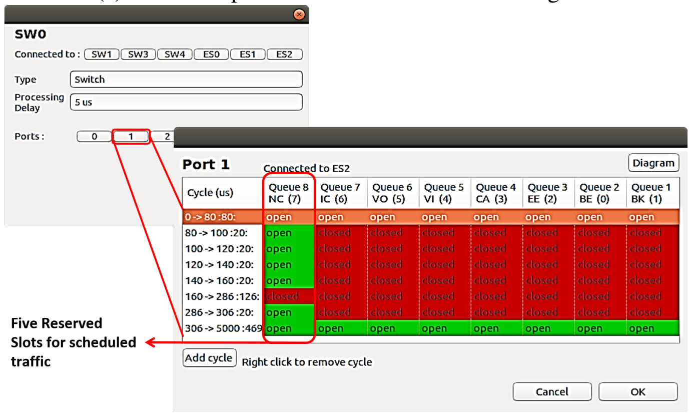
We can take advantage of the GUI's visualization feature to view these large configuration files, as demonstrated in Fig. 5. The results of the schedule synthesis module provide offsets to start the transmission of the flows. Hence, there are 40 offset values to set for each ST flow. In case there are multiple flows from a source end station to multiple destination end stations, we need to calculate the transmission offsets of frames transmitted from the same source end station up to the hyper period of the flows. Therefore, there are going to be 118 frame entries to configure all ST flow instances. Fig. 5(a) shows the first instance of each flow from ES1. There are four flows from ES1 to the destination end stations ES7, ES8, ES9 and ES10 with periods 2000 μs, 1000 μs, 2000 μs and 5000 μs, respectively. On the link from ES1 to SW0, $[ES1, SW0]$, the offsets of frames are calculated for the duration of the hyper period, i.e., 10000 μs. In other words, it is required to calculate 10 offsets for a flow that has a period of 1000 μs within a hyper period of 10000 μs. Moreover, the switch gate states are specified by the scheduled time stamps to



Default cycle: 10000us

Frame	Path	Cycle period (us)	Size (Bytes)	Start delay (us)	Queue
Frame 0 of Flow 3	ES1 SW0 SW3 ES8	10000	1500	0	8
Frame 0 of Flow 4	ES1 SW0 SW4 ES9	10000	1500	0	8
Frame 0 of Flow 5	ES1 SW0 SW4 ES10	10000	1500	40	8
Frame 0 of Flow 2	ES1 SW0 SW3 ES7	10000	1500	164	8

(a) Flow and path visualization with TSN Plug-in.


Add cycle
Right click to remove cycle
Cancel
OK
Diagram
Five Reserved Slots for scheduled traffic

SW0

Connected to: SW1 SW3 SW4 ES0 ES1 ES2

Type: Switch

Processing Delay: 5 us

Ports: 0 1 2

Port 1 Connected to ES2

Cycle (us)	Queue 8 NC (7)	Queue 7 IC (6)	Queue 6 VO (5)	Queue 5 VI (4)	Queue 4 CA (3)	Queue 3 EE (2)	Queue 2 BE (0)	Queue 1 BK (1)
0 -> 80 :80:	open	open	open	open	open	open	open	open
80 -> 100 :20:	open	closed	closed	closed	closed	closed	closed	closed
100 -> 120 :20:	open	closed	closed	closed	closed	closed	closed	closed
120 -> 140 :20:	open	closed	closed	closed	closed	closed	closed	closed
140 -> 160 :20:	open	closed	closed	closed	closed	closed	closed	closed
160 -> 286 :126:	closed	closed	closed	closed	closed	closed	closed	closed
286 -> 306 :20:	open	closed	closed	closed	closed	closed	closed	closed
306 -> 5000 :469	open	open	open	open	open	open	open	open

(b) Switch gate visualization by the GUI.

Fig. 5: Configuration visualizations with TSN Plug-in.

enable deterministic transmission of the ST flows from egress ports of the network switches. Table I shows the total number of gate entries to be configured in the switches for duration of the hyper period of each link. Since different ST flows with different profiles might be scheduled on the same link, the lengths, periods, and egress gate synthesis cycle should take hyper period of all the periodic ST flows on the link into account.

TABLE I: Configuration complexity.

Switches	Total gate state changes
Switch 0	135
Switch 1	21
Switch 2	80
Switch 3	123
Switch 4	74

The next step, after obtaining the configuration parameters, is to insert them to the topology created in the NeSTiNg simulator using the plug-in module. Accordingly, setting up the frame's offsets and gate states by the plug-in GUI is quite a tedious task, hence we use the plug-in's automated mode sub-module and apply the configuration data files from the traffic generator and schedule synthesis modules. Consequently, 867 lines for configuration of scheduled frames, 1964 lines of gate states configurations, and 77 lines for the switches' routing tables were automatically generated to setup the NeSTiNg simulator. Fig. 5(b) demonstrates the gate states of port "1" at the switch "SW0" connected to ES2 up to the hyper period (5000 μs) of the link from SW0 to ES2. The Queue8 column in Fig. 5(b) shows the reserved ST transmission slots on corresponding link. The size of the frame transmission slot is set to 20 μs because the complete transmission of an Ethernet frame on a link with 1 Gbit/s speed is 13 μs. Hence, we assume a 7 μs safety margin for the ST frame to pass through the link and the switch. We assume that open state is the default value of the switch gates. Furthermore, there are five 20 μs slots on the link, $[SW0, ES2]$, which are reserved for transmission of the five ST frames. This is the link where the traffic from the source end stations ES3, ES5, ES7, ES10, ES11 are transmitted, each with a period of 50000 μs.

The framework's final module, enables extraction of end-to-end delays of all ST flows after running the simulation for a desired amount of time. The feasibility of the schedules is inspected by checking if the maximum end-to-end delay of each flow is less than or equal to the flow period. Table II presents the maximum end-to-end delay for each ST flow retrieved by the results extraction sub-module, where TX and RX indicate the source end station and the destination end stations. The results confirm the feasibility and determinism of the offline schedule.

TABLE II: End-to-end delay per transmitted flow.

ID	TX	RX	Period(μs)	Delay(μs)
1	ES0	ES6	5000	52
2	ES1	ES7	2000	52
3	ES1	ES8	1000	52
4	ES1	ES9	2000	92
5	ES1	ES10	5000	37
6	ES2	ES3	1000	93
7	ES2	ES7	1000	82
8	ES3	ES1	5000	81
9	ES3	ES2	5000	64
10	ES3	ES4	2000	24
11	ES3	ES5	2000	37
12	ES4	ES3	5000	53
13	ES4	ES8	1000	65
14	ES4	ES9	1000	92
15	ES4	ES11	5000	53
16	ES5	ES0	2000	1888
17	ES5	ES2	5000	72
18	ES5	ES4	2000	37
19	ES5	ES6	5000	54
20	ES6	ES5	5000	64

ID	TX	RX	Period(μs)	Delay(μs)
21	ES6	ES7	5000	34
22	ES6	ES8	2000	24
23	ES7	ES1	5000	57
24	ES7	ES2	5000	77
25	ES7	ES4	1000	72
26	ES8	ES0	2000	158
27	ES8	ES10	1000	72
28	ES8	ES11	5000	75
29	ES9	ES0	5000	970
30	ES9	ES1	5000	53
31	ES9	ES3	5000	37
32	ES10	ES2	5000	61
33	ES10	ES5	5000	73
34	ES10	ES6	1000	112
35	ES10	ES7	5000	136
36	ES10	ES11	5000	37
37	ES11	ES2	5000	61
38	ES11	ES7	1000	97
39	ES11	ES9	5000	45
40	ES11	ES10	5000	49

VI. Conclusions

This paper proposed an automated modular framework to facilitate automated offline scheduling, pre-simulation configurations and interpretation of simulation results in TSN networks. The proposed framework addresses the challenge of configuring synthesized schedules to the state-of-the-art simulation framework, namely NeSTiNg. The proposed framework automatically translates the TSN flow schedules into the simulation platform's compatible syntax without any manual intervention. The proposed framework is implemented as an open-source TSN plugin, which is integrated to the NeSTiNg simulation framework. The applicability of the framework is demonstrated by automatically configuring a large TSN network and performing the simulation-based evaluation of the network. The evaluation results show the feasibility of the proposed framework for TSN networks. Furthermore, the results demonstrate the interoperability of the proposed framework with the state-of-the art simulation frameworks for TSN. The benefits of the proposed automated configuration framework are magnified in the case of large TSN networks, where manual configurations can be error prone, time-consuming and very challenging to perform, thus rendering the existing methods impractical for industrial applications. As the future work, we aim at integrating the proposed framework with existing industrial tools for modeling of automotive embedded systems that use TSN for on-board network communication.

Acknowledgement

The work in this paper is supported by the Swedish Governmental Agency for Innovation Systems (VINNOVA) through the projects DESTINE and PROVIDENT, and by the Swedish Knowledge Foundation (KKS) through the projects DPAC, HERO & FIESTA. The authors thank all industrial partners, especially Arcticus Systems, Volvo CE, and HIAB Sweden.

References

[1] "IEEE Time-Sensitive Networking (TSN) Task Group." [Online]. Available: https://1.ieee802.org/tsn

[2] L. Lo Bello, R. Mariani, S. Mubeen, and S. Saponara, "Recent advances and trends in on-board embedded and networked automotive systems," *IEEE Transactions on Industrial Informatics*, 2019.

[3] L. Lo Bello and W. Steiner, "A Perspective on IEEE Time-Sensitive Networking for Industrial Communication and Automation Systems," *Proceedings of the IEEE*, June 2019.

[4] J. Falk, D. Hellmanns, B. Carabelli, N. Nayak, F. Dürr, S. Kehrer, and K. Rothermel, "NeSTiNg: Simulating IEEE Time-sensitive Networking (TSN) in OMNeT++," in *Proceedings of the 2019 International Conference on Networked Systems*. IEEE, March 2019.

[5] D. Hellmanns and J. Falk. (2020) Nesting - network simulator for time-sensitive networking. [Online]. Available: https://gitlab.com/ipvs/nesting

[6] S. S. Craciunas, R. S. Oliver, and T. AG, "An overview of scheduling mechanisms for time-sensitive networks," *Proceedings of the Real-time summer school (ETR)*, 2017.

[7] H. Farzaneh *et al.*, "A modeling framework to facilitate schedule synthesis of time-sensitive networking," Ph.D. dissertation, Technische Universität München, 2019.

[8] A. C. T. dos Santos, B. Schneider, and V. Nigam, "TSNSCHED: Automated schedule generation for time sensitive networking," in *2019 Formal Methods in Computer Aided Design*. IEEE, 2019.

[9] V. Gavriluţ, L. Zhao, M. L. Raagaard, and P. Pop, "AVB-aware routing and scheduling of time-triggered traffic for TSN," *IEEE Access*, 2018.

[10] M. Pahlevan, R. Obermaisser, "Genetic algorithm for scheduling time-triggered traffic in time-sensitive networks," in *23rd International Conference on Emerging Technologies and Factory Automation*, 2018.

[11] J. Jiang, Y. Li, S. H. Hong, A. Xu, and K. Wang, "A time-sensitive networking (TSN) simulation model based on OMNET++," in *2018 IEEE International Conference on Mechatronics and Automation*, 2018.

[12] (2020) OMNeT++ technical articles - result analysis with python. [Online]. Available: https://docs.omnetpp.org/tutorials/pandas/

[13] A Quick Overview of the OMNeT++ Intgrated Development Environment, 2020, https://doc.omnetpp.org/omnetpp/IDE-Overview.pdf .

[14] A. Bergström, Automatic Generation of Network Configuration in Simulated Time Sensitive Networking (TSN) Applications, Master Thesis, School of Innovation, Design and Engineering, Mälardalen University, Sweden, 2020.

[15] S. Kramer, D. Ziegenbein, and A. Hamann, "Real World Automotive Benchmarks for Free," in *6th Int. Workshop on Analysis Tools and Methodologies for Embedded and Real-Time Systems*, 2015.