End-to-end QoS Management in Self-Configuring TSN Networks

End-to-end QoS Management in Self-Configuring TSN Networks

Andrea Garbugli
University of Bologna
Bologna, Italy
andrea.garbugli@unibo.it

Armir Bujari
University of Bologna
Bologna, Italy
armir.bujari@unibo.it

Paolo Bellavista
University of Bologna
Bologna, Italy
paolo.bellavista@unibo.it

***Abstract*—Industrial networked computing environments are expected to serve a wide range of applications with heterogeneous Quality-of-Service (QoS) requirements. This capability demands for novel, QoS-aware network management and configuration techniques resilient in the face of network changes. State-of-the-art approaches only focus on aspects related to the management of network devices. In this work, we move a step further, proposing an end-to-end QoS management approach in Time-Sensitive Networking (TSN) compliant networks, capable of handling reconfiguration events e.g., link-drop. Shedding some light on our proposal, we first discuss its functional building blocks, successively validating the approach on a real TSN testbed.**

***Index Terms*—time-sensitive networking, management, reconfiguration**

I. Introduction

The Internet of Things (IoT), and in particular its industrial application referred to as the Industrial Internet of Things (IIoT), is at the heart of the Industry 4.0 evolution, seen as an enabler for intelligent and cooperative Cyber-Physical Systems. IIoT represents the convergence of flexible communication technologies, new computing techniques such as cloud/edge computing and the application of the IoT vision to industrial production systems. As a result, industrial devices and machines will use heterogeneous wired/wireless technologies to communicate with applications running on local/global cloud/edge platforms, in order to take advantage of new efficient solutions, such as process scheduling and optimization [1].

Indeed, cloud/edge computing is starting to be seen as a relevant opportunity that can contribute to the convergence of Operation and Information Technology (OT/IT) domains, opening the road for end-to-end optimizations of processes. This objective can be achieved through the on-premise deployment of layer(s) of edge/fog nodes with different capabilities, seamlessly integrating compute, storage and networking functionalities in support of industrial applications. This networked computing environment gives rise to a more fluid model identified as the Cloud-to-Thing Continuum (C2TC) [2].

The C2TC model provides several benefits compared to a pure datacenter, cloud-based approach achieved in terms of application bandwidth/latency and data security/privacy. However, a seamless integration of all the levels of the infrastructure (both cloud and edge) demands for novel management approaches capable of joint orchestration of (virtualized) compute, storage and networking resources aimed at ensuring industrial applications QoS guarantees.

In this depicted scenario, QoS-aware network management and configuration is a necessary building block, providing the basis for higher level, service/application-aware functionalities. In this work in progress study, we present an end-to-end self-(re)configuring solution for use in Time Sensitive Networking (TSN) compliant networks. To this aim, we first discuss some control and data plane functional components along with the envisaged management and configuration interfaces. Next, we assess the proposal on a real TSN testbed, comprised of a multi-hop network topology, servicing industrial control applications. Contributing to the state-of-the-art, our solution extends the configuration and management functionality of TSN to all elements of the network, including end-devices, and is capable of (transparent) handling of reconfiguration events.

II. Background

In this section, we provide a concise survey onto a subset of TSN standards, focusing on QoS and network configuration related issues need to better comprehend this work.

A. Time Synchronization

An essential network functionality required to support low-latency industrial traffic is accurate time synchronization, aimed at defining a shared time reference among all network entities. In the context of TSN, related mechanisms are specified under the IEEE 802.1AS standalone standard. The IEEE 802.1AS is based on a specialized profile of the IEEE 1588 Precision Time Protocol (PTP) standard, called the generic Precision Time Protocol (gPTP). gPTP envisages two main entities: a (i) Clock Master (CM) and a (ii) Clock Slave (CS), both deployed and provisioned on the networked devices, exchanging messages embodying synchronization events. One CM also called the PTP grandmaster, sends the information to each of the Clock Slaves connected to it, for example, using multicast communication. Each of the CSs, also called gPTP instances, must correct the synchronized time received. To this aim, they must add the time delay due to the propagation of the message along the gPTP communication path, from the grandmaster to the PTP instance. Once all devices are

synchronized, we have what is in effect a time-aware network, also referred to as a gPTP domain.

B. Time-Trigger Traffic

The IEEE 802.1Qbv is a standard aimed at supporting real-time traffic in TSN networks. This standard specifies the techniques and mechanisms supporting different types of time-critical flows, introducing the notion of time-triggered communication windows often referred to as a secure traffic windows or time-aware traffic windows. In this context, a window is divided into multiple time slots that repeat cyclically, and it possible to associate traffic flows to selected traffic classes. This helps prevent lower priority traffic, such as best-effort traffic, from interfering with real-time or scheduled traffic transmissions. A so-called guard band precedes the scheduled traffic windows, and packets belonging to other traffic classes are buffered until their traffic class can be transmitted. A time-aware schedule is expressed in terms of slotted communications, implemented by means of a Gate Control List (GCL), identifying time-instants where packets can be transmitted on the medium.

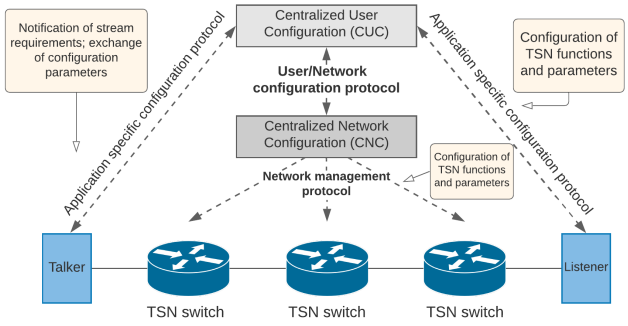


Fig. 1. Centralized Configuration and Management Model.

C. Network Configuration

An important feature available in IEEE 802.1 networks is that using plug-and-play mechanisms to e.g., seamlessly add and configure new devices to the network. A standard for providing similar functionalities on TSN-compliant networks is the IEEE 802.1Qcc, which defines several configuration models ranging from centralized to distributed ones. The standard introduces several entities and among those is the: (i) User-Network Interface (UNI) and the (ii) Centralized Network Configuration (CNC) node. The UNI provides a standard method of requesting L2 services while the CNC interacts with the UNI for performing resource reservation, scheduling, and other types of configuration via a management protocol.

An optional functional entity called Centralized User Configuration (CUC) communicates with the CNC via a standard interface, implementing the so-called fully centralized network architecture. The CUC can be used to discover end stations, talkers and listeners in TSN terminology, retrieve their capabilities, and to configure time-scheduled features in end stations (mainly for industrial control applications).

In this context, the YANG Data Model provides a framework for periodic status reporting and configuration of 802.1 bridges and components. In specific, YANG is a information model used to express configuration and state data for network management protocols. The latter, called the Network Configuration Protocol (NETCONF), provides mechanisms for installing, managing and deleting the configurations of network devices [3].

III. Related work

Herein, we provide a concise survey of state-of-the-art work related to the end-to-end management and control topic in TSN.

In [4], an architecture for real-time systems called Time-sensitive Software-defined Network (TSSDN) is presented. The objective is to schedule and route time-triggered traffic using commodity hardware via a software-defined networking approach. The authors present several scheduling algorithms, assigning time-slots to time-triggered flows, minimizing in-network queuing while maximizing the number of co-existing flows.

Gutiérrez *et al.* propose a heuristic capable of run-time configuration of fog-enabled TSN [5], [6]. The authors in their works take into account time-critical flows that can appear and disappear over time. To this end, they adopt a configuration agent architecture able to reacts to network (traffic) changes.

Gerhard *et al* in [7] present an approach that combines SDN and TSN with emphasis on network management and configuration. Their architecture, called Software-defined Flow Reservation (SDFR), implements the IEEE 802.1Qcc standard and can be integrated with existing SDN solutions. The proposal allows for the configuration of time-triggered traffic flows exploiting a southbound interface protocol.

In this work, we propose a centralized architectural solution and functional building blocks, extending the concept of self-(re)configuration and monitoring to all elements of the network, including end-devices. The proposed solution is validated on a real TSN testbed, showcasing its capability to dynamically adapt to network changes.

IV. System Architecture

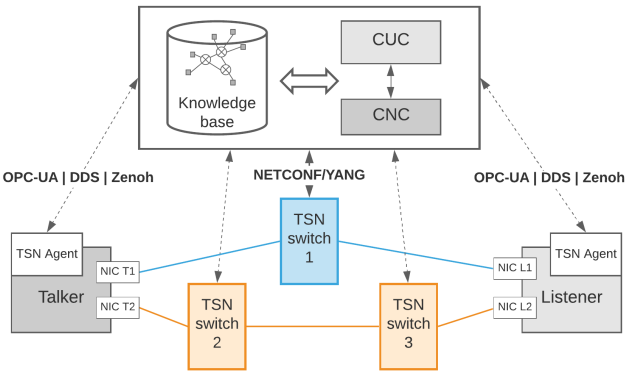


Fig. 2. Proposed QoS Management Architecture.

Fig. 1 provides a high-level overview on our proposal. Architecturally, we follow the centralized management and configuration model comprising the CUC, CNC and UNI entities. The solution is made up of several components, each of which plays a particular role in managing and servicing the different actors of a TSN network.

A. Centralized Network Configuration

The CNC communicates directly with the network devices employing the NETCONF protocol. Once attached to the network, the CNC obtains the capabilities and configurations of the devices. This information is then saved in the knowledge base present in our architecture. Operationally, the CNC is tasked with the management of the networked devices enforcing QoS as requested by TSN communication endpoints. To this aim, the CNC enacts the following mechanisms:

- PTP-based synchronization service management.
- VLAN setup and configuration.
- Network Schedule distribution.

From a monitoring viewpoint, the module takes advantage of the NETCONF Event Notifications protocol, which allows us to express topic interests, subscribing to specific notification events. In particular, we are interested in receiving events related to a configuration changes, errors and possibly other metrics quantifying the data plane performance.

B. Centralized User Configuration

The CUC cooperates with the TSN Agent (later on) in order to manage the configuration of the TSN stream(s) servicing the QoS specifications as mandated by end-devices. The management control flow is designed to work in both directions, that is, an end-device can send a configuration request for a TSN stream to the CUC via the TSN Agent, but also the CUC can push a new configuration towards the TSN Agent. The CUC also takes care of updating the logically centralized knowledge base.

For the CUC-TSN Agent communication interface, one can rely on different options e.g., OPC-UA, DDS, and Zenoh [8]. Our current implementation relies on Zenoh, a data-centric middleware solution, allowing for future extensions and integration of the approach in a C2TC, heterogeneous networking environment.

C. Knowledge Base

The knowledge base represents the module in which information regarding managed elements, including end-devices, is stored. The information stored herein ranges from device configuration, physical and overlay network topology to active communication flows.

To build the knowledge base, all participants in the TSN network send periodical updates using the CUC-CNC interface. The knowledge base can be queried e.g., in case of reconfiguration events.

D. TSN Agent

The TSN agent represents the module deployed on the end-device tasked with communication with the CUC, requesting the setup and configuration of a viable QoS-aware TSN streams. To act upon the request, the CUC queries the knowledge base, computes a viable schedule and sends the configuration parameters to the TSN Agent. Moreover, the CUC also takes care of tasking the CNC to manage the configuration of the network path connecting the devices, setting up the end-to-end TSN stream.

The TSN Agent exploits Netlink, a standard socket-based interface that allows user space applications to communicate and modify the settings of some kernel modules. In specific, the TSN Agent uses the `NETLINK_ROUTE` subsystem, often referred to as $rtnetlink$, enabling the reception of routing and link updates. This also gives us the capability to e.g., modify routing tables (both IPv4 and IPv6), link parameters, neighbors settings, queuing disciplines etc. All of these features are essential to support and enact the concept of a TSN stream. Currently, we use two new queuing disciplines ($qdiscs$) built into the Linux kernel: (i) `taprio` (Time Aware Priority Shaper) implementing a simplified version of the scheduling defined by IEEE 802.1Qbv and (ii) `etf` (Earliest TxTime First) qdisc. This latter discipline allows applications to set a transmission time for each packet, information used by the scheduler to dequeue the packet and forward it over the TSN. The synchronization feature is implemented via the `linuxptp` package.

As far as monitoring is concerned, the TSN Agent exploits the capability of Netlink to subscribe to one or more multicast groups in order to receive networking events. In particular, the TSN Agent subscribes to the group on `RTMGRP_LINK` of the `NETLINK_ROUTE` family. This group allows the TSN Agent to receive events related to the configuration and status updates of network interfaces.

V. In-the-field Experimental Validation

A. Experimental Settings

The goal of the experiment is to validate the approach, showcasing its capability to handle reconfiguration events at run-time. To this end, we have developed a real testbed based on the architecture shown in Fig 2. The testbed comprises a T (talker) and an L (listener), each represented by a UP Xtreme board equipped with 4 TSN NICs (Intel I210), Intel (R) Core (TM) i3-8145UE 2/4 CPU, and 8GB RAM.

With reference to Fig 2, the boards (T, L) are connected to each other via two distinct physical paths. The first one (blue color) connects T's NIC (T1) with L's NIC (L1) through Switch 1 (SW1). We refer to this connection Stream 1 (S1). The second path (orange color) connects T's NIC (T2) with L's NIC (L2) through Stream 2 (S2), which crosses Switch 2 (SW2) and Switch 3 (SW3). The two streams are configured with two different VLANs.

The traffic is UDP-based with a payload varying from 32, 64, 128 and 256 bytes. For each configuration, we send

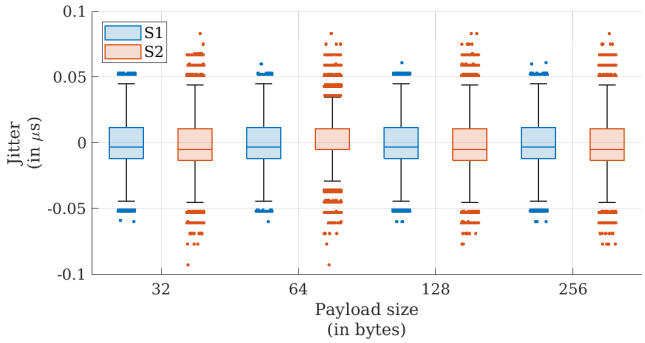
10^5 packets with a 1 ms regular interval. In the first set of experiments, we measure both latency and jitter of the time critical flows.

In the second experiment, we showcase the TSN Agents' capability of identifying a link-drop event. In this scenario, the talker communicates with the listener using the S1 stream, after which a drop occurs between T and SW1. Once the event is noticed by the agent, it stops the transmission of the talker through NIC (T1), requests a new configuration from the CUC, rescheduling it via the viable alternative S2. Once the configuration parameters have been received, the TSN Agent starts the reconfiguration process via NIC (T2), updating the network scheduling parameters, starting the PTP functionality as to as re-synchronize the clocks. Upon termination, packets can flow via the viable alternative S2.

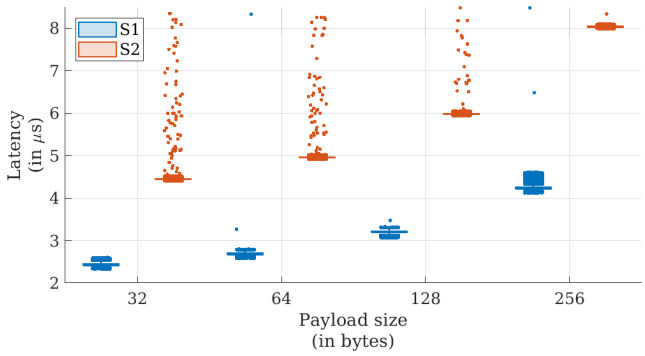
B. Results

In Fig. 3 are reported the jitter and latency values for both the S1 and S2 streams. One can observe that the flows QoS are in line with the 1 ms specification. Although the jitter remains more or less constant both with the increase of the payload and the communication stream used, the latency, on the other hand, is always greater during communication via S2 compared to S1. This is essentially due to the fact that S2 needs to go through two hops (SW2 and SW3), introducing an increase in latency in the order of microseconds. In all configurations, the proposed approach is able to fully satisfy QoS specifications.

Finally, in Fig. 4 is shown the result of the reconfiguration scenario. In this setting, the communication is initially serviced via the S1 and, following a link-drop event, goes through S2 with a downtime period due to the reconfiguration of about 150 ms.



(a) Observed jitter for streams S1 and S2.



(b) Observed latency for streams S1 and S2.

Fig. 3. Jitter and latency of the received UDP packets; metrics are expressed for each packet size and stream.

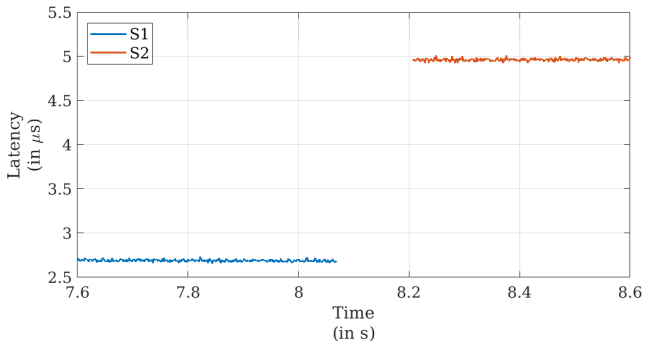


Fig. 4. Observed latencies before and after the reconfiguration phase.

VI. Conclusions

We discussed a practical, end-to-end TSN compliant QoS management approach, validating it in a real testbed. This work is part of an on-going project which has the ambitious object of devising mechanisms and techniques for the end-to-end QoS management of industrial C2TC computing environments comprising heterogeneous communication technologies.

References

[1] 5G-AICA, "Integration of 5G with Time-Sensitive Networking for Industrial Communications," Feb. 2021. [Online]. Available: https://www.5g-acia.org/whitepapers/integration-of-5g-with-time-sensitive-networking-for-industrial-communications/

[2] P. Pop, M. L. Raagaard, M. Gutierrez, and W. Steiner, "Enabling Fog Computing for Industrial Automation Through Time-Sensitive Networking (TSN)," *IEEE Communications Standards Magazine*, vol. 2, no. 2, pp. 55–61, 2018.

[3] L. Lo Bello and W. Steiner, "A Perspective on IEEE Time-Sensitive Networking for Industrial Communication and Automation Systems," *Proceedings of the IEEE*, vol. 107, no. 6, pp. 1094–1120, 2019.

[4] Nayak, Naresh Ganesh and Dürr, Frank and Rothermel, Kurt, "Time-sensitive software-defined network (TSSDN) for real-time applications," *ACN International Conference on Real-Time Networks and Systems*, pp. 193–202, 2016.

[5] M. Gutiérrez, A. Ademaj, W. Steiner, R. Dobrin, and S. Punnekkat, "Self-configuration of IEEE 802.1 TSN Networks," in *IEEE International Conference on Emerging Technologies and Factory Automation, ETFA*, 2017.

[6] M. L. Raagaard, P. Pop, M. Gutierrez, and W. Steiner, "Runtime Reconfiguration of Time-sensitive Networking (TSN) Schedules for Fog Computing," in *IEEE Fog World Congress*, 2018, pp. 1–6.

[7] T. Gerhard, T. Kobzan, I. Blöcher, and M. Hendel, "Software-defined Flow Reservation: Configuring IEEE 802.1Q Time-Sensitive Networks by the Use of Software-Defined Networking," in *IEEE International Conference on Emerging Technologies and Factory Automation, ETFA*, vol. 2019, 2019, pp. 216–223.

[8] A. Corsaro and G. Baldoni, "fogØ5: Unifying the Computing, Networking and Storage Fabrics End-to-end," in *Cloudification of the Internet of Things (CIoT)*, 2018, pp. 1–8.