CISTER
Research Centre in
Real-Time & Embedded
Computing Systems

Conference Paper

On the adequacy of SDN and TSN for Industry 4.0

Luis Silva
Paulo Pedreiras
Pedro Fonseca
Luis Almeida

CISTER-TR-190402

On the adequacy of SDN and TSN for Industry 4.0

Luis Silva, Paulo Pedreiras, Pedro Fonseca, Luis Almeida

CISTER Research Centre

Rua Dr. António Bernardino de Almeida, 431

4200-072 Porto

Portugal

Tel.: +351.22.8340509, Fax: +351.22.8321159

E-mail:

https://www.cister-labs.pt

Abstract

Industry 4.0, Industrial Internet of Things, Cyber-Physical Production Systems and Smart Factories are closelyrelated emerging concepts expected to drive significant improvementsin industrial production systems, with gains in efficiency,cost and customer satisfaction. These concepts are intimatelyassociated with highly distributed and cooperative architecturesthat rely, naturally, on the network infrastructure. However,traditional industrial communication technologies hardly providethe required level of integration, flexibility and performance.Seeking a solution to this mismatch, we assess two technologiesthat appeared recently in the industrial realm, namely IEEE802.1 Time-Sensitive Networking (TSN) and Software-DefinedNetworking (SDN). TSN and SDN are fundamentally different,thus having distinct strengths and weaknesses. This paper reviewstheir fundamental operation principles, evaluating themqualitatively against the specific requirements posed by Industry4.0.

On the adequacy of SDN and TSN for Industry 4.0

Luis Silva, Paulo Pedreiras, Pedro Fonseca
IT - Instituto de Telecomunicações
DETI - Universidade de Aveiro
Aveiro, Portugal
{lems,pbrp,pf}@ua.pt

Luis Almeida
CISTER - Research Center in Real-Time Systems
DEEC / FEUP - Universidade do Porto
Porto, Portugal
lda@fe.up.pt

***Abstract*—Industry 4.0, Industrial Internet of Things, Cyber-Physical Production Systems and Smart Factories are closely related emerging concepts expected to drive significant improvements in industrial production systems, with gains in efficiency, cost and customer satisfaction. These concepts are intimately associated with highly distributed and cooperative architectures that rely, naturally, on the network infrastructure. However, traditional industrial communication technologies hardly provide the required level of integration, flexibility and performance. Seeking a solution to this mismatch, we assess two technologies that appeared recently in the industrial realm, namely IEEE 802.1 Time-Sensitive Networking (TSN) and Software-Defined Networking (SDN). TSN and SDN are fundamentally different, thus having distinct strengths and weaknesses. This paper reviews their fundamental operation principles, evaluating them qualitatively against the specific requirements posed by Industry 4.0.**

***Index Terms*—Industry 4.0, Time-Sensitive Networking, Software-Defined Networking, Real-Time Communications**

I. Introduction

INDUSTRY 4.0 aims at the digitization of production systems, allowing a seamless integration of physical objects, humans and logical entities. This new paradigm comes with the promise of significant improvements in industrial production systems, such as fewer equipment failures and downtime, reduced maintenance, higher throughput and higher flexibility, to name just a few, with gains in efficiency, cost, and customer satisfaction [1].

This paradigm, inherently distributed and cooperative, relies on industrial networks to enable data exchanges and, ultimately, the cooperation among assets, from elemental sensors and actuators, to complex production machinery, maintenance systems, management and logistics, *etc*. Data streams reflect this diversity, varying in bitrate, criticality and timeliness. These communication requirements are beyond what conventional fieldbus protocols, *e.g.* CAN and PROFIBUS, can provide due to limitations in bandwidth, physical range, addressing space, *etc*. Real-Time Ethernet protocols are increasingly used in new machinery and connection systems, exploiting Ethernet high bandwidth, low cost and standardized hardware. However, these protocols still fail in providing the integration needed in Industry 4.0 due to incompatible options in their stacks. As consequence, integration requires deploying gateways that increase cost, latency, and management complexity. Moreover, current Real-Time Ethernet protocols were mostly designed to favor real-time performance over run-time flexibility, and those that support this flexibility offer limited real-time performance [2].

Attempting to cater for Industry 4.0 requirements, two technologies appeared recently in the industrial realm, arriving from different domains [3] [4]. One is Ethernet IEEE 802.1 Time Sensitive Networking (TSN), which aims at very low latency and high availability, with origin in audio/video transmission but later enhanced for the automotive and industrial control domains. The other is Software-Defined Networking (SDN) that introduced the concept of network programmability and with origin in data networks management. SDN offers an unprecedented level of run-time flexibility, with benefits in management, configuration efficiency, and performance. While TSN takes an evolutionary approach, extending the Ethernet base to fulfill specific requirements, *e.g.*, time-triggered traffic, reservations and configuration, SDN takes a disruptive approach, introducing a framework that allows implementing, programmatically, the communication services.

This paper reviews the concepts of Industry 4.0 and Smart Factories (Section II), focusing on the communication requirements, as well as the operation principles of SDN (Section III) and TSN (Section IV). Our contribution is a qualitative comparison of how efficiently TSN and SDN address the requirements of Industry 4.0 (Section V).

II. Industry 4.0 communication requirements

Industry 4.0 means the large scale application of Information and Communication Technologies (ICT) to industrial production systems. The expression originated from a strategic initiative of the German government to exploit Germany's leadership in machinery industry, ICT competences and embedded systems [5] but it soon became a global buzz-word for the digitization of production processes, associated to the concept of Digital or Smart Factory.

A Smart Factory can reconfigure itself based on instantaneous information from the production system through a myriad of sensors. This requires full horizontal and vertical integration. The former means interoperability between all stages of the value chain, from the moment raw materials enter the premises, through the different steps of the production

This work has been supported by the European Regional Development Fund (FEDER) through a grant of the Operational Programme for Competitivity and Internationalization of Portugal 2020 Partnership Agreement (PRODUTECH-SIF, POCI-01-0247-FEDER-024541).

process, to the delivery to the customer. The latter means interoperability through all levels of the production system, from Enterprise Resource Planning (ERP) systems to the control of every machine in the plant. The digital horizontal and vertical coupling allows an unprecedented level of flexibility and autonomy, such as production batches of size one, where every product is customized.

Supporting this level of digitization is a combination of an ever increasing computing power at lowering costs, ubiquitous communication networks, increasing data storage capability and the emergence of machine learning algorithms. This is revolutionizing how production systems operate. With information on every detail of the production process a virtual factory replica is created, its digital twin, which, in turn, allows simulating the production system, testing solutions before they are implemented in the plant. This implies a complex exchange of multiple and heterogeneous flows of data visible, for instance, in ISA-95 [6], the ANSI standard for Entreprise-Control Systems Integration. ISA-95 defines a function hierarchy that organizes factory control and supervision systems in 4 levels, starting in Level 1 (sensing and manipulating the production process), going through levels 2 (monitoring, supervisory control and automated control of the production process) and 3 (work flow / recipe control to produce the desired end products, maintaining records and optimizing the production process) up to level 4 (establishing the basic plant schedule: production, material use, delivery, and shipping and determining inventory levels). These levels differ significantly in their time frames and size of transactions, from few bits in milliseconds scale in level 1 to large files in daily scale in level 4. And all exchanges must co-exist over the network, i.e., the download of an 800 MB CNC program by the Manufacturing Execution System to a machine for the next production item cannot interfere with the sensor and actuator flows involved in producing the current item.

Currently, a common solution to support such disparate exchanges is using separated networks. Field-buses handle the frequent and small transactions of ISA-95 level 1 while large and less frequent transactions use general purpose IP networks. However, this becomes an obstacle to integration and runtime flexibility, requiring gateways to interconnect the multiple networks, adding complexity, latency and potential faults.

Conversely, virtualizing large bandwidth networks, *e.g.*, Ethernet-based, can also support those heterogeneous transactions, providing virtual channels to distinct flows, each with guaranteed Quality of Service. It simplifies the global communications architecture, facilitating vertical and horizontal integration and enhancing operational flexibility.

III. Background on SDN

Software Defined Networking (SDN) is a network management paradigm that emerged to tackle the complexity in managing large networks, *e.g.* campus or data center networks, providing the flexibility to support server virtualization and cloud computing. SDN can be briefly characterized by the separation of traffic control (control plane) and traffic forwarding

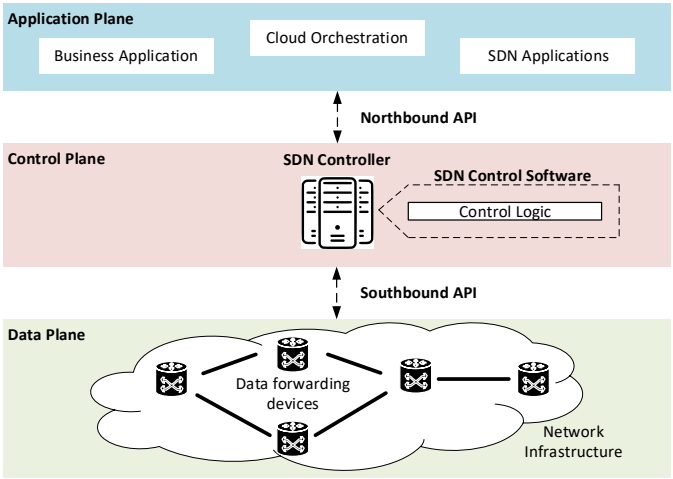

Fig. 1. Architecture of an SDN-enabled network

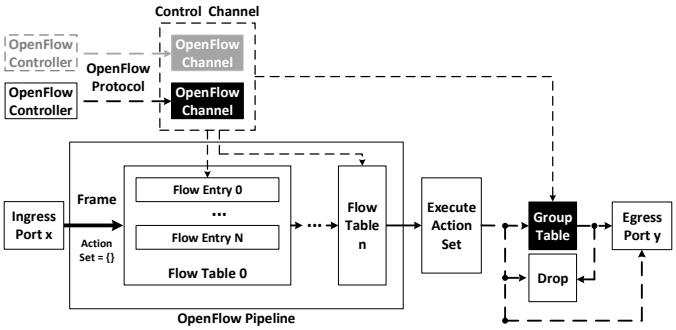

Fig. 2. Overview of OpenFlow objects

(data plane) (Fig. 1). In SDN, traffic control rules are defined by an SDN controller that performs admission control of new traffic flows and instructs network devices, *e.g.*, switches, on how to forward traffic. Controller and devices interact via the control *southbound interface* that abstracts away differences among data plane technologies. The current *de facto* standard southbound interface is the OpenFlow protocol. User applications run in the application plane and request network services through the *northbound interface* [7] [8].

A. OpenFlow Protocol

OpenFlow, maintained by the Open Network Foundation (ONF), defines a set of objects, procedures and signalling messages that OpenFlow-enabled forwarding devices must adhere to [9]. An Openflow switch (Fig.2 [9]) consists of a forwarding pipeline with one or more *flow tables* (the *OpenFlow Pipeline*), a *group table*, a *control channel* comprising one or more logical *OpenFlow channels* to external controllers, and a set of *ingress/egress OpenFlow ports*. An OpenFlow device must support three types of OpenFlow ports: (i) *physical*, *i.e.*, switch hardware ports, (ii) *reserved*, *i.e.*, logical ports for specific predefined processing, such as forwarding packets to a set of physical ports; (iii) *logical*, for processing methods defined outside the protocol. Packet, time and other counters, used for statistics and bandwidth control, are also mandatory.

OpenFlow switches receive frames at ingress ports and send them to the first flow table of the OpenFlow pipeline for processing [9]. Each flow table contains a set of *Flow Entries* that comprise: (i) a priority level; (ii) match fields; (iii) associated instructions; (iv) fields for liveliness and statistics. Filters can address several frame fields such as Virtual Local Area Network (VLAN) ID, UDP/TCP ports, and Ethernet and IPV4 addresses. In a flow table, frames are matched against flow entries following priority order. When a frame matches the filters of a flow entry the associated instructions are performed, possibly changing the list of actions associated to the frame (*Action Set*), modifying certain fields of the frame, or explicitly directing it to a subsequent flow table for further processing. Once the frame reaches the last table, or is not directed to a subsequent one by the matched entry, its current action set is executed. Thus, a frame can be sent to a group table, forwarded to a given OpenFlow egress port, or dropped. Frames that do not match a flow entry are either dropped or, if configured, processed according to a special entry and, for example, sent to the controller (*packet-in* transaction). Group tables contain a subset of instructions similar to those of flow tables, with similar outcomes, and a set of special actions (*buckets*) to facilitate complex operations such as fast fail-over and link aggregation.

Optionally, devices can provide a *Meter Table* with traffic shapers (*Meters*) that can be configured and targeted by actions to constrain the traffic rate at egress ports. Each meter measures rate of traffic flows assigned to it and can be configured to drop frames or increase the drop precedence of DiffServ flows upon reaching a programmable rate threshold.

Finally, there are OpenFlow messages for SDN controllers to configure device capabilities (*e.g.*, number of flow tables), filters and instructions of entries belonging to flow and group tables, meters, and to retrieve state information, etc. Communication between controllers and switches uses OpenFlow channels that provide secure and reliable message delivery.

B. SDN in industry

The unprecedented level of flexibility provided by SDN soon raised interest on using it in industry as reported in the scientific literature, including qualitative and quantitative evaluations, as well as methods to extend its functionality.

1) Qualitative and Performance Evaluations: Both Henneke *et al* [10] and Ehrlich *et al.* [11] identify industrial communication requirements that are not adequately supported by SDN, yet, *e.g.*, expressing timing and Quality-of-Service (QoS) guarantees. Thiele *et al.* [12] performed a formal analysis of OpenFlow deployed on Time-Sensitive Networking (TSN) Ethernet considering the communication with the controller, the network topology and the scalability limits for real-time usage, showing that latency values below 50ms are possible. Herlich *et al.* [13] highlight the possible gains in supporting arbitrary network topologies, dynamic (re)configurations, and fast fail-over by using SDN on real-time Ethernet networks.

2) SDN Extension for Industrial Scenarios: In [14], Ternon *et al.* investigate how the Flexible Time-Triggered (FTT) paradigm can be instantiated on standard OpenFlow hardware making it suitable for real-time scenarios. The paper presents a new protocol, FTT-OpenFlow, that enhances the response time of sporadic real-time traffic, shown in an avionics scenario, and of non-real-time traffic, too, in single switch scenarios. In [15], Nayak *et al.* exploit the logical centralization of SDN to build a global view of the network and compute routes and transmission schedules that reduce in-network queuing of time-triggered traffic. Despite obtaining low latency and jitter, the proposal does not address coexistence with sporadic real-time traffic. Ahmed *et al.* [16] propose SDPROFINET, an SDN deployment over PROFINET networks. Similarly to [15], the central controller acquires network information and configures the data plane PROFINET data channels according to the desired routes. Despite the gains in network management, operational flexibility is constrained by PROFINET. In [17], Ishimori *et al.* propose a hierarchical scheduling approach, similar to that of Linux Traffic Control (TC), to overcome the limitations of the First-In First-Out (FIFO) queues in OpenFlow devices. It supports HTB (Hierarchical Token Bucket), RED (Randomly Early Detection), and SFQ (Stochastic Fair Queuing) that, however, provide bandwidth-based traffic shaping, only, with limited latency control. Finally, Silva *et al.* [18] propose extending OpenFlow with the capacity to manage real-time reservations provided by certain switches in the data-plane , *e.g.*, HaRTES. They also propose a new controller extended to handle real-time attributes and provide schedulability analysis for time-triggered traffic.

IV. Background on TSN

TSN is a set of technical standards developed by the IEEE 802.1 time-sensitive networking task group [19], the successor of the audio/video bridging (AVB) task group, to improve the real-time behaviour of IEEE 802 network technologies. TSN focuses on four main aspects: temporal synchronization among devices, end-to-end bounded latency and high reliability for real-time traffic streams, as well as management of network resources. An overview of the current set of standards and amendments is depicted in Fig.3. Greyed out standards/amendments are currently being under development. As not all enhancements may be supported by VLAN bridges, TSN defines a set of profiles (see Fig.3), specifying the necessary features and performance requirements for a given application area, that can be used as reference for vendors.

In TSN networks, IEEE 802.1Q VLAN bridges [21], enhanced with TSN features, interconnect *end stations* (application nodes) following topologies identical to those found in standard switched Ethernet networks. End stations and bridges can be synchronized in time (IEEE 802.1AS and IEEE P802.1AS-Rev [22]) to ensure jitter and synchronization requirements of time-sensitive applications such as audio and video streaming. To provide seamless redundancy and increase the reliability of real-time communications, source end stations may transmit duplicated frames (pertaining to a given stream)

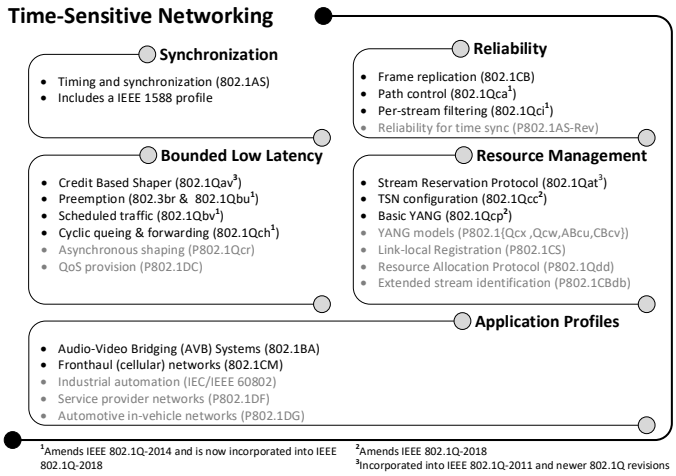

Fig. 3. Time-Sensitive Networking (TSN) set of standards/amendments [20]

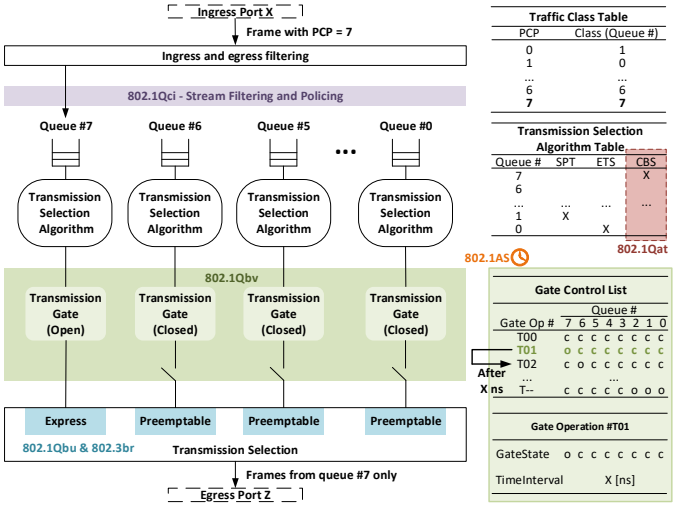

Fig. 4. TSN enhancements to the forwarding process of IEEE 802.1Q VLAN bridges

across different paths in the network (IEEE 802.1CB [23]). Frame replicas are automatically discarded at convergence points in the network and/or at sink end stations. Network engineers may use TSN's path control services (802.1Qca [21]) to configure the necessary disjoint network paths.

The key TSN features with focus on QoS provisioning, in particular on the latency and determinism aspects of traffic with timeliness requirements, augment the *forwarding process* of standard VLAN bridges as shown in Fig.4 (TSN-related enhancements are colored). Next, we present an overview of the forwarding process and how these enhancements are framed in it. In Section IV-B we summarize the operation of the most relevant QoS services, while in Section IV-C we introduce the models for the configuration of real-time reservations in TSN.

A. Forwarding process

In enhanced VLAN bridges, frames received at ingress ports are first subjected to ingress and egress filtering which considers distinct types of information, *e.g.* active topology and frames' MAC addresses, to set the potential egress ports for each frame. Then, the bridge consults a *Traffic Class Table* to translate the priority conveyed by the Priority Code Point (PCP) field of frames' 802.1Q tag into a certain *traffic class*. Frames of a certain traffic class are stored into the respective FIFO-like queue, *e.g.* class 7 on queue #7 and class 0 on queue #0. By default, untagged frames convey the priority level 0. With IEEE 802.1Qci, enabled bridges can use a set of meters and time-aware filters to police traffic to, for example, direct traffic from misbehaving classes to low priority queues and prevent queue flooding [21]. As the PCP field comprises 3 bits, up to 8 traffic classes with separate queues may be supported. Moreover, all traffic class tables may be configured by management or TSN's reservation services.

Finally, each queue is associated with a *Transmission Selection Algorithm* that selects frames for transmission if: (i) its operation determines that there is a frame available for transmission, and (ii), the algorithm of numerically higher traffic classes, *i.e.* with higher priority, determines that there are no frames available. On each port, a configurable *Transmission Selection Algorithm Table* assigns to each traffic class one of the following algorithms [21]:

- ***Strict priority***: a frame is available if the queue contains one or more frames;
- ***Credit-Based Shaper (CBS)***: a frame is available for transmission if the queue is not empty and the shaper credit is zero or positive;
- ***Enhanced Transmission Selection (ETS)***: a frame is available if: (i) the queue is not empty, (ii) the class has not surpassed the allocated bandwidth, (iii) there are no frames available in queues running strict priority or CBS algorithms.

The *Transmission Selection* component verifies the state of each selection algorithm in descending order of priority, *i.e.* from queue #7 down to queue #0, and sends selected frames through the egress port. If the bridge supports frame preemption (802.1Qbu and 802.3br [21], [24]), ongoing transmissions of traffic classes configured as *preemptable* are deferred in favor of classes deemed *express*. Additionally, available frames from queues whose *Transmission Gate* (explained bellow) is on *close* state are not serviced for transmission.

B. Forwarding QoS services

1) Frame preemption (IEEE 802.1Qbu and IEEE 802.3br): Frame preemption allows frames from express traffic classes to suspend the transmission of classes deemed preemptable. Preempted frames are divided into fragments carrying a minimum of 60 bytes from the original frame, 8 bytes as preamble and control, and 4 bytes for error detection (Frame Check Sequence (FCS) from the original frame in the last fragment) [24]. Therefore, frames sized less than 123 bytes or with less than 64 bytes left to transmit can't be preempted. Moreover, fragmentation is confined to a single link between two hops, *i.e.* a frame is reassembled before being sent to the following link.

2) Credit Based Shaper (IEEE 802.1Qav): The CBS algorithm shapes the transmission of traffic from egress queues according to the bandwidth negotiated for a stream reservation while also controlling burstiness. Following the operation of CBS, a queued frame is selected for transmission if the medium is free, no higher priority traffic awaits transmission, and the credit value is zero or positive. Credit decreases or increases at distinct rates when a frame is being transmitted or waiting for transmission, respectively. The rate slopes are proportional to the amount of reserved bandwidth for the queue. Bounds for the maximum accumulated credit limit the maximum burst size a class may perform after awaiting transmission due to interference [21].

3) Traffic Scheduling (IEEE 802.1Qbv): IEEE 802.1Qbv allows TSN devices to transmit frames according to a time schedule [21]. All queues are associated with a *Transmission Gate* which is able to enable (gate *Open*) or disable (gate *Closed*) the associated transmission selection algorithm, allowing or preventing it from selecting frames from the concerned queue. Transmission schedules can be built by configuring, on each port, the respective *Gate Control List*. This list contains an ordered set of *Gate Operations* that dictate, for a given time interval, the state of each transmission gate. Each control list can be (re)configured, set to execute at given time instant, and repeated periodically following a configurable schedule cycle time. An example of a configuration using transmission gates is presented in Figure 4.

The aforementioned services can exploit the time synchronization provided by 802.1AS to implement distinct scheduling models, *e.g.* Time-Division Multiple Access (TDMA). Cyclic Queuing and Forwarding (CQF) (IEEE 802.1Qch), colloquially known as Peristaltic Shaping, combines the special time-aware stream filters of IEEE 802.1Qci and the scheduling facilities of IEEE 802.1Qbv to devise a model where frames progress through the network cyclically while stopping at each hop for exactly one schedule cycle time [21]. The *Paternoster* algorithm [25] and Asynchronous Traffic Shaping (ATS) (P802.1Qcr [26]) are alternative models to CQF that do not require global time synchronization and reduce forwarding latency at the cost of higher jitter and additional resources.

As traffic target by TSN scheduling services is typically sensitive to jitter, a guard window must be enforced, *e.g.* using gates to block non-scheduled queues, at the start of every schedule cycle to prevent interference from non-scheduled traffic. If the bridge supports frame preemption, special gate operations can suspend on-going transmissions before the start of a new cycle and minimize the length of the guard window.

C. Configuration Models (IEEE 802.1Qcc)

The TSN traffic model uses the concept of *Stream* as a unidirectional flow of time-sensitive data sent by a single *Talker* to one or more *Listener* end stations. To configure network bridges with the necessary QoS services, users (Talkers and Listeners) must first provide a set of information regarding their capabilities, resources, and characteristics of associated streams. In summary, frames are associated to streams via fields such as VLAN ID and MAC/IP addresses, streams are classified into traffic classes using the PCP code and their properties expressed via maximum frame size, transmission period (interval), and the number of frames per interval [27].

To exchange the aforementioned information and enact configurations, TSN provides the following configuration models [27].

1) Fully distributed model: Following this model, user requirements are propagated along the network, from talkers to listeners, without the use of a centralized configuration entity. As requirements propagate through the network, each bridge locally verifies whether it has enough resources to fulfill the requesting stream's requirements, and configures the necessary QoS services accordingly. TSN relies on the Stream Reservation Protocol (SRP) [21], extended by 802.1Qcc [27], to enable users to setup bandwidth reservations for up to 7 configurable classes (*Stream Reservation* - SR - A up to G). Due to its distributed nature and reliance on SRP, this model only offers the ability to set CBS shapers to queues and configure their parameters [27].

2) Centralized network - distributed user model: Here, akin to the distributed model, user requirements are sent by talkers/listeners to the nearest bridge. However, the information is now relayed to a Centralized Network Configuration (CNC) entity which configures all bridges using a remote network management protocol, *e.g.* NETCONF, and Yet Another Next Generation (YANG) models (IEEE 802.1Qcw [28]). The enhanced SRP can also be used here for the communication between talkers/listeners and the CNC. As the CNC has a complete view of the network topology and capabilities of each bridge, and employs powerful device management protocols, the following TSN services can now be configured [27]: CBS shapers, frame preemption and replication, scheduled traffic, and per-stream filtering and policing.

3) Fully centralized model: Akin to the previous model, a CNC is responsible for the configuration and management of the whole network. However, talker/listener information is now provided directly to the CNC by a Centralized User Configuration (CUC) entity. TSN does not define a protocol for the communication between CUC and CNC, however, it requires that the necessary information is exchanged. YANG models can be used for both the network management and the CUC-CNC interface. All TSN services can be managed using this model.

V. Evaluation of SDN and TSN in the context of Industry 4.0

This section evaluates SDN, SDN extensions and TSN, with respect to the requirements of Industry 4.0. The requirements of Industry 4.0 are very broad, ranging from strict real-time performance to security and dependability, see e.g. [10], [11].

In this paper we focus on a subset of these requirements related to real-time performance, QoS and flexibility. More precisely:

- **Real-time performance:** compliance with latency and jitter figures of real-time traffic;

- **Overhead:** evaluates bandwidth consumed and/or wasted by the protocols;
- **Mutual isolation:** support to heterogeneous traffic types without mutual interference;
- **Granularity of QoS control:** diversity and parametrization of allowed QoS policies;
- **Traffic management architecture:** supported logical management architectures and how it affects resource management efficiency;
- **Flexibility:** applications should be able to create and modify reservations promptly, in order to adapt to environment or production changes;

For sake of conciseness, the SDN/OpenFlow extensions reviewed in Section III-B2 are labeled as: FTT-Openflow [14];TSSDN [15]; SDPROFINET [16]; SDN-HSF [17]; and OpenFlow-RT [18].

A. Real-time performance

TSN provides services specifically tailored for traffic with Real-Time (RT) requirements, namely CBS and Transmission Gates, eventually set to peristaltic mode. The former service provides a shaping service which allows gross bandwidth reservations suitable for real-time Event Triggered (ET) streams. However, the limited number of traffic classes (up to 6, as the maximum number is 8 and two are reserved for background and management), flat server structure and hard to analyze (at least for real-time purposes) server type, constraint the response-time and jitter of this kind of traffic. Transmission Gates are specifically designed for periodic traffic, creating contention-free time-based transmission slots that suit well Time-Triggered (TT) communications, allowing low latency and jitter.

OpenFlow (OF) was not designed for real-time systems, not distinguishing RT traffic from Non Real-Time (NRT) one, and there is no explicit support to real-time activation modes (TT, ET). Time issues are only mentioned to allow the application of synchronized updates on a given set of OF switches. In addition, priorities are supported on output queues, which usually are available in a limited number, thus constraining the support of scheduling policies for realistic cases. As such, the real-time performance of OF is poor.

FTT-OpenFlow is an implementation of FTT-SE [29] on OpenFlow, which preserves the original periodic traffic management of FTT-SE while improving the handling of sporadic real-time traffic by modifying the signaling mechanism associated to these messages. It thus allows an efficient handling of both TT and ET RT traffic.

TSSDN and SDPROFINET bring support to TT traffic on SDN, allowing low figures of jitter and latency. In the former case this is achieved by synchronizing end nodes to avoid contention among TT packets. Interference between TT and other traffic is handled via priorities, eventually combined with frame preemption mechanisms. In the latter case, SDN is used to manage PROFINET switches, thus inheriting the real-time attributes of this protocol. In both cases there is good support to TT traffic, but there is no explicit handling for RT ET traffic. Moreover, TSSDN relies solely on controlling the transmission instants at end nodes, without depending on bridge-level scheduling services. As such, the schedulability level for TT traffic is reduced, as the protocol does not allow any kind of overlapping in flow paths.

In turn, SDN-HSF does not explicitly addresses RT traffic, but the enhancements on queuing disciplines brings significant improvements on traffic isolation and bandwidth control, with a positive impact on ET RT traffic.

Finally, Openflow-RT supports explicitly both ET and TT traffic by means of dynamic explicit scheduling in the former case, and hierarchical servers in the second one. This allows low latency and jitter for both kinds of traffic.

B. Overhead

Attaining low jitter in communication systems supporting event-triggered traffic is complex and, usually, impacts negatively on bandwidth utilization efficiency. This is particularly noticeable when there is joint support for TT traffic, as this kind of traffic is often associated with very strict jitter requirements.

TSN introduces a frame preemption mechanism that allows to interrupt the transmission of less important and/or jitter tolerant messages (classified as preemptable) in favor of other messages more jitter sensitive (classified as express). There is a minimum size before preemption can occur, which implies wasted bandwidth in each transmission slot for the case TT traffic is managed by Transmission Gates. Moreover, preemption also adds overhead, as control bytes must be added to the packet segments. TSSDN behaves similarly to TSN, as the use of frame preemption is allowed and it comprises the notion of slots for TT messages.

On the other hand, SDPROFINET, FTT-OpenFlow and OpenFlow RT use dedicated windows for TT traffic, blocking eventual ET transmissions that could otherwise overrun TT transmission windows. This represents network idle-time that translates to overhead. In general the impact of the inserted idle-time is moderate, as it is inserted once per window, not once per message.

Control messages are another potentially relevant source of overhead. In this regard, FTT-OpenFlow and OpenFlow RT are penalized by the need to disseminate periodically elementary cycle's transmission schedule. The impact is inversely proportional to the elementary cycle duration, starting to be relevant for cycles below $1ms$.

OF and SDN-HSF don't have relevant overheads as they don't implement the functionalities above described.

C. Mutual isolation

TSN provides a set of mechanisms that allow some degree of traffic isolation. There are distinct traffic classes that can be associated with different forwarding mechanisms, which include the selection algorithms (prioritized transmissions, shapers) and the Transmission Gates. Moreover, the availability of filtering and policing also impacts positively on isolation by allowing to bound the interference of misbehaving

streams. Although this set of features is interesting from the traffic isolation point of view, the performance of TSN in this regard is impaired by the limited number of priorities, which are associated with traffic classes. As mentioned above, in practical terms only 6 classes can be used, so per-stream confinement and isolation is far from reach.

SDPROFINET is based on PROFINET and, as such, segregates traffic in NRT, RT and IRT. The IRT traffic comprises exclusive transmission slots for isochronous traffic streams. Therefore, IRT streams are completely isolated form each other and from the other classes. However, for RT traffic, the isolation is not so strong, as the mechanisms are based only on communication stack adaptations (IP partially abandoned and direct use of Layer 2/OSI services). Moreover, RT ET traffic is not explicitly supported.

TSSDN only segregates RT TT traffic from the remaining one. No other mechanisms, except references to the use of traffic prioritization, are provided. So, this protocol is quite limited in this respect.

FTT-OpenFlow and OpenFlow RT are based on the FTT paradigm. As such, they isolate the transmission of TT, ET and NRT traffic, which have exclusive transmission windows. Moreover, in both these protocols RT ET traffic is managed by hierarchical servers, which allows a fine grain stream composition (from individual streams to subsystems and systems) with bounded and predictable interference. In the particular case of OpenFlow RT, the HaRTES-based bridges implement traffic policing and have separated memory areas for the different traffic types, assuring the robustness of the isolation mechanisms.

Support of isolation on OF and SDN-HSF is poor, as there is no notion of traffic types. The simple use of priorities and improved queuing management policies are not enough to attain an acceptable level of performance in this regard.

D. Granularity of QoS control

In what concerns QoS granularity for real-time systems, TSN performance is modest. On the one hand, the TSN set of standards lacks support to some attributes commonly used in real-time systems (*e.g.* activation paradigm, precedence constraints, offsets), and the existing QoS attributes are of limited usefulness in what regards real-time applications. For example, CBS parameters are specified as bandwidth, and reservations are issued based on number of frames per time interval and maximum latency, only. Moreover, QoS is in practice specified per class, not per stream because, as mentioned above, the number of priorities/classes is low (up to 6, in practice).

OF also does not support the set of attributes commonly used in real-time systems. QoS specification is limited to bandwidth limitations and priorities. As such, its performance in this regard is poor.

TSSDN is also rather limited. The only specific reference to QoS parameterization of real-time traffic is the specification of the periodicity for TT streams, which is assumed to be expressed as an integral multiple of a base-period that corresponds to a minimum system-wide transmission period that can be supported. No other attributes or traffic types are explicitly supported. There are references also to the use of priorities to favor the transmission of TT traffic when a time slot cannot be found.

SDPROFINET uses formal specifications based on the behavioral type concept, to identify interdependencies between different devices. The authors propose using regular expression based specification mechanisms to capture a sequence chart of communication messages. Reference [16] is does not go deep enough to allow a well supported evaluation of the protocol in this regard, but the approach should allow capturing the essential attributes.

FTT-OpenFlow and OpenFlow RT are based on the FTT paradigm and support the full set of attributes commonly used in real-time systems. Streams are individually associated with attributes such as period, deadline,offsets and activation mode (TT/ET). These attributes as, in practical terms, unconstrained, *e.g.* there are no limits to the number of priorities. Moreover, for the RT ET it is possible to specify QoS at diverse levels, thanks to the presence of hierarchical servers. As such, these protocols excel in this aspect.

SDN-HSF is quite limited in what concerns QoS granularity. The improvements over SDN are restricted to queue management and the implemented disciplines (HTB, RED and SFQ) only provide bandwidth-based traffic shaping. As such, despite improving performance, explicit support to real-time traffic QoS is still poor.

E. Traffic Management Architecture

TSN inherited AVBs fully distributed model based on the Stream Reservation Protocol. Traffic reservation requests are propagated along the network and each device (end nodes and bridges) decide on the admissibility of each request. It should be noted, however, that SRP only permits to manage stream reservations associated with CBS. Centralized architectures are known to enable more efficient and responsive resource management, supported by a broader knowledge on existing resources and requirements. TSN traffic management was recently augmented with a centralized management option, where reservations are directed to a Centralized Network Configuration entity that decides about the admissibility of reservations and then, when appropriate, uses remote management protocols (*e.g.*, SNMP, NETCONF) to configure bridges in accordance with the requirements. As such, TSN excels in this aspect, by allowing both distributed (with limitations) and centralized management architectures, that have distinct advantages and disadvantages.

SDN prescribes a logically centralized architecture, where the controller is the sole entity responsible for configuring the data-plane switches. As mentioned above, this architecture is arguably more efficient in what concerns the management of network resources, but detractors also point weaknesses, *e.g.* in what concerns scalability. The enhancements brought by SDN-HSF and TSSDN to SDN do not impact significantly on the traffic management architecture of SDN, thus sharing essentially the same properties.

SDPROFINET allows the existence of diverse domain controllers, but prescribes that copies of controllers shall reside on a remote control center to allow topology changes and other modifications to be carried out centrally. The objective is to allow network stabilization and instantiate modifications consistently, eliminating the possibility of instability and transients during updates.

FTT-OpenFlow and OpenFlow RT also have strongly centralized architecture, inherited both from the base FTT architecture and SDN, which are both centralized. In fact, the more elemental functionalities of these protocols depend on the presence of a (logically) centralized entity, which performs activities that go well beyond handling reconfigurations, *e.g.* periodic traffic scheduling. Thus, these protocols depend almost entirely on the permanent availability of the controller.

F. Flexibility

TSN provides mechanisms to configure all the services directly and indirectly related with stream forwarding, *e.g.* set CBS parameters, configure the Transmission Gates schedule, issue or remove a stream reservation, *etc*. Despite that, there are some limitations. For example, modifications to stream attributes are not directly supported, and must be instantiated as a tear down followed by a creation. This procedure requires multiple message exchanges and involves several timeout mechanisms, limiting the responsiveness to system adaptations, particularly for the case of distributed architectures, as messages have to be propagated through the entire path. More importantly, TSN only provides the basic mechanisms to parametrize the configuration of network devices, not assisting the applications in managing the allocated QoS. As such, QoS management is entirely left to the application side, which is a severe limitation in terms of flexibility and adaptability. Finally, TSN does not allow the creation of application-specific protocols. The application can only choose which protocols to use (from a predefined set) and configure them.

SDN is, in some sense, in the opposite side of TSN. Due to its programmatic approach, the set of protocols that can be deployed is virtually unlimited, being possible to design applications taking standard protocols eventually complemented with custom ones, specifically tailored for individual applications. The centralized architecture also potentiates quick modifications to the system configuration, so the manipulation of stream attributes and system configuration changes can be carried with low delay. As TSN, SDN but does not assist applications in QoS management.

TSSDN, SDPROFINET, and SDN-HSF are SDN-based and thus, share the characteristics of OF in what regards flexibility and adaptability. TSSDN and SDPROFINET may impose a penalization in latency upon system changes due to the computational complexity of the methdos used to derive the schedules and routes. As TSN and OF, these protocols do not provide QoS management, which must also be performed by applications.

As for the case of TSN and OF, the centralized architecture of FTT-OpenFlow and OpenFlow RT enables fast reconfigurations. These protocols allow to create, delete, and modify message streams without service disruption. A unique feature of FTT-OpenFlow and OpenFlow RT that stems from its FTT roots is the native QoS management provided by the network. These protocols provide admission control capabilities along with a QoS manager to which applications may send requests specifying acceptable levels of QoS, *e.g.* in the form of acceptable ranges or bounds for periodicity or deadline. When resources are insufficient, the QoS manager interacts with the admission control to try to find feasible configurations. Therefore, these protocols provide a much better support to flexibility and adaptability than the other ones.

G. Overall evaluation

Table I summarizes the results of the above discussion, mapping the performance of the protocols in each criteria in a qualitative scale that ranges from 1 (Worse) to 5 (Better). There are some interesting conclusions that can be withdrawn. TSN performs well or very well in all criteria. The limitations it exhibits result essentially from backward compatibility issues. The limited number of priorities is particularly relevant as it constrains, in a fundamental way, several aspects of the protocol performance (*e.g.* traffic isolation, event-based messages). The overall complexity that results from the combination of an huge number of protocols, several of them not designed for real-time applications, turns the resulting system hard to analyze and to prove correct.

SDN takes a radically different approach, promising an unprecedented degree of flexibility thanks to its programmability. However, it was designed for data centers and lacks expressiveness to handle real-time scenarios, therefore its overall real-time performance is very poor.

The potentialities of SDN have been recognized and thus several contributions eventually appeared, having all in common the objective of enriching SDN with real-time services, while preserving its essential attributes. Some of the approaches are simpler but still very limited in terms of real-time performance (SDN-HSF). Others go one step ahead, using dedicated hardware and/or modifications to the communication stack and global management, improving significantly the real-time performance of SDN (TSSDN and SDPROFINET). Finally, FTT-OpenFlow and OpenFlow RT exploit the QoS management and flexibility, characteristic of the FTT paradigm, to enhance OF with efficient real-time and QoS management services.

Summarizing, the qualitative evaluation herein presented clearly shows that TSN performs well, but it is far from perfect, having inherent performance limitations. On the other hand, it also shows that SDN can be augmented, in different ways, to support effective and efficiently applications with real-time requirements, thus being a promising alternative to TSN.

VI. Conclusions

Industry 4.0 poses new requirements that traditional industrial communication protocols cannot cope with, in particular

TABLE I
EVALUATION OF SDN, SDN EXTENSIONS, AND TSN WITH RESPECT TO PERFORMANCE, QOS, AND FLEXIBILITY

Criteria		TSN	OpenFlow	FTT-OpenFlow	TSSDN	SDPROFINET	SDN-HSF	OpenFlow RT
RT Performance	TT	5	1	5	4	5	1	5
	ET	3	1	5	1	3	3	5
Overhead		4	5	3	4	4	5	3
Mut. Isolation		4	1	5	2	3	1	5
QoS Granularity		3	1	5	2	4	2	5
Management Arch.		5	3	3	3	4	3	3
Flexibility		3	4	5	4	4	4	5

From 1 (Worse) to 5 (Better)

the need for integrating large sets of heterogeneous real-time traffic with strict demands in terms of flexibility and reconfiguration. Two innovative network technologies, TSN and SDN, are emerging as candidate solutions to fill in this gap. In this paper we evaluate qualitatively TSN, OpenFlow and a set of SDN/OF-based protocols. The comparison shows clearly that TSN performs well, but has intrinsic limitations that arise form its evolutionary nature. While SDN, due to its roots, is unsuitable for industrial scenarios, its inherent flexibility allows the development of extensions that bring real-time and predictable services. As such, SDN proves to be an effective base to develop communication frameworks competitive with TSN in Industry 4.0 scenarios.

REFERENCES

[1] V. Koch, S. Kuge, R. Geissbauer, and S. Schrauf, "Industry 4.0: Opportunities and challenges of the industrial internet," *Strategy & PwC*, 2014.

[2] M. Ashjaei et al., "Improved message forwarding for multi-hop hartes real-time ethernet networks," *Journal of Signal Processing Systems*, vol. 84, no. 1, pp. 47–67, Jul 2016.

[3] S. Hoppe, "Opc foundation extends opc ua including tsn down to field level." [Online]. Available: https://opcfoundation.org/news/press-releases/opc-foundation-extends-opc-ua-including-tsn-down-to-field-level/

[4] K. Ahmed, N. S. Nafi, J. O. Blech, M. A. Gregory, and H. Schmidt, "Software defined industry automation networks," in *2017 27th International Telecommunication Networks and Applications Conference (ITNAC)*, Nov 2017, pp. 1–3.

[5] Henning Kagermann, Wolfgang Wahlster, and Johannes Helbig, "Recommendations for implementing the strategic initiative INDUSTRIE 4.0 - Final report of the Industrie 4.0 Working Group," Federal Ministry of Education and Research - Germany, Tech. Rep., Apr. 2013.

[6] H. M. Hashemian, "ANSI/ISA-95.00.01-2010 (IEC 62264-1 Mod) Enterprise-Control System Integration - Part 1: Models and Terminology," 2010.

[7] T. N. D. and K. Gray, *SDN: Software Defined Networks*, 1st ed. O'Reilly Media, Inc., 2013.

[8] D. Kreutz, F. M. V. Ramos, P. E. Verssimo, C. E. Rothenberg, S. Azodolmolky, and S. Uhlig, "Software-defined networking: A comprehensive survey," *Proceedings of the IEEE*, vol. 103, no. 1, pp. 14–76, Jan 2015.

[9] "Openflow switch specification version 1.5.0 (protocol version 0x06)," *Open Networking Foundation*, pp. 1–277, Dec. 2014.

[10] D. Henneke, L. Wisniewski, and J. Jasperneite, "Analysis of realizing a future industrial network by means of Software-Defined Networking (SDN)," in *2016 IEEE World Conference on Factory Communication Systems (WFCS)*, May 2016, pp. 1–4.

[11] M. Ehrlich et al., "Software-Defined Networking as an Enabler for Future Industrial Network Management," in *2018 IEEE 23rd International Conference on Emerging Technologies and Factory Automation (ETFA)*, vol. 1, Sep. 2018, pp. 1109–1112.

[12] D. Thiele and R. Ernst, "Formal analysis based evaluation of software defined networking for time-sensitive Ethernet," in *2016 Design, Automation Test in Europe Conf. (DATE)*, March 2016, pp. 31–36.

[13] M. Herlich, J. L. Du, F. Schrghofer, and P. Dorfinger, "Proof-of-concept for a software-defined real-time Ethernet," in *2016 IEEE 21st International Conference on Emerging Technologies and Factory Automation (ETFA)*, Sep. 2016, pp. 1–4.

[14] C. Ternon, J. Goossens, and J.-M. Dricot, "FTT-OpenFlow, on the Way Towards Real-time SDN," *SIGBED Rev.*, vol. 13, no. 4, pp. 49–54, Nov. 2016.

[15] N. G. Nayak, F. Dürr, and K. Rothermel, "Time-sensitive Software-defined Network (TSSDN) for Real-time Applications," in *Proceedings of the 24th International Conference on Real-Time Networks and Systems*, ser. RTNS '16. New York, NY, USA: ACM, 2016, pp. 193–202.

[16] K. Ahmed, J. O. Blech, M. A. Gregory, and H. Schmidt, "Software defined networking for communication and control of cyber-physical systems," in *2015 IEEE 21st International Conference on Parallel and Distributed Systems (ICPADS)*, Dec 2015, pp. 803–808.

[17] A. Ishimori, F. Farias, E. Cerqueira, and A. Abelém, "Control of Multiple Packet Schedulers for Improving QoS on OpenFlow/SDN Networking," in *2013 Second European Workshop on Software Defined Networks*, Oct 2013, pp. 81–86.

[18] L. Silva, P. Gonçalves, R. Marau, P. Pedreiras, and L. Almeida, "Extending OpenFlow with flexible time-triggered real-time communication services," in *2017 22nd IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, Sep. 2017, pp. 1–8.

[19] IEEE 802.1 Working Group, "Time-sensitive networking task group." [Online]. Available: http://www.ieee802.org/1/pages/tsn.html

[20] J. Farkas, "Overview of ieee 802.1 tsn and ietf detnet." [Online]. Available: https://mentor.ieee.org/802.11/dcn/18/11-18-2027-00-0000-overview-of-ieee-802-1-tsn-and-ietf-detnet.pdf

[21] "IEEE Standard for Local and Metropolitan Area Network–Bridges and Bridged Networks," *IEEE Std 802.1Q-2018 (Revision of IEEE Std 802.1Q-2014)*, pp. 1–1993, July 2018.

[22] "IEEE Draft Standard for Local and Metropolitan Area Networks - Timing and Synchronization for Time-Sensitive Applications," *IEEE P802.1AS-Rev/D8.0 January, 2019*, pp. 1–446, Jan 2019.

[23] "IEEE Standard for Local and metropolitan area networks – Frame Replication and Elimination for Reliability," *IEEE Std 802.1CB-2017*, pp. 1–102, Oct 2017.

[24] "IEEE Standard for Ethernet Amendment 5: Specification and Management Parameters for Interspersing Express Traffic," *IEEE Std 802.3br-2016*, pp. 1–58, Oct 2016.

[25] M. Seaman, "Paternoster policing and scheduling." [Online]. Available: http://www.ieee802.org/1/files/public/docs2017/cr-seaman-paternoster-policing-scheduling-0317-v03.pdf

[26] "IEEE Draft Standard for Local and Metropolitan Area Networks-Bridges and Bridged Networks Asynchronous Traffic Shaping," *IEEE P802.1Qcr/D0.5, June 2018*, pp. 1–112, Jan 2018.

[27] "IEEE Standard for Local and Metropolitan Area Networks – Bridges and Bridged Networks - Amendment 31: Stream Reservation Protocol (SRP) Enhancements and Performance Improvements," *IEEE Std 802.1Qcc-2018*, pp. 1–208, Oct 2018.

[28] "P802.1Qcw YANG Data Models for Scheduled Traffic, Frame Preemption, and Per-Stream Filtering and Policing."

[29] R. Marau, L. Almeida, and P. Pedreiras, "Enhancing real-time communication over cots ethernet switches," in *2006 IEEE International Workshop on Factory Communication Systems*, June 2006, pp. 295–302.