Quality Checks After Production: TSN-based Industrial Network Performance Evaluation

Mohamed Seliem
School of Computer Science and IT
University College Cork
Cork, Ireland
m.seliem@cs.ucc.ie

Ahmed Zahran
School of Computer Science and IT
University College Cork
Cork, Ireland
a.zahran@cs.ucc.ie

Dirk Pesch
School of Computer Science and IT
University College Cork
Cork, Ireland
d.pesch@cs.ucc.ie

***Abstract*—The automation of quality checks of a product in a smart manufacturing environment poses several challenges for the underlying industrial network. Such a network involves a variety of connected devices, e.g. sensors, actuators, embedded computers, to perform detection of a newly arriving product, visual inspection for quality checks, and classification to decide on the final quality of the product. These devices generate different types of data flows depending on their function. Here, we model the use case of quality checks using visual inspection after production. We consider the industrial network to be based on Time Sensitive Networking (TSN) standards to meet the different data flow quality of services (QoS) requirements. We use OMNET++ to model our use case. In addition, we extend an existing TSN module implementation to configure the network layer for our application model. We conduct a set of simulations while considering worst case analysis with infinite and finite queue sizes, and realistic data traffic models. Our simulation results show that using a combination of Time Aware (TAS) and Credit-based (CBS) shaping outperforms standard and priority Ethernet queuing strategies and achieves a high delivery ratio of 100% for critical data traffic and 94% for burst traffic, while meeting end-to-end latency requirements.**

***Index Terms*—TSN, Industrial Automation, IIoT, QoS, OMNET++.**

I. Introduction

Industrial Internet of Things (IIoT) applications integrate a large number of devices (e.g., sensors and actuators) that generate data traffic with strict quality of service (QoS) requirements in terms of transmission reliability, bounded delays, and transmission rate. The TSN standards (IEEE 802.1) family [1] provide a network protocol framework to realize such requirements while efficiently utilizing industrial network resources. The TSN design incorporates four core principles: 1) Synchronizing all devices to the same master clock [2]; 2) Traffic isolation with different traffic streams classified in different queues for processing according to their priority, i.e., QoS requirements [3]; 3) Flexible traffic scheduling and shaping mechanisms (e.g., time-based [4] or credit-based [5] shaping) to control the delivery of different flows to meet delay limits; and 4) Proactive fault detection and elimination to ensure transmission reliability [6].

Product quality checking represents a typical IIoT application in smart manufacturing that generates flows, each with distinct characteristics, e.g., frame size and frame arrival time, and requirements, e.g. delay limits and bandwidth. In such a use case (see depiction in Fig. 1, the arrival, on a conveyor belt, of a new product leads an object detection sensor to generate a control message to trigger a quality inspection. As the product moves on the belt, the control message has strict requirements on the delay ($\sim 100\mu$s). The inspection sensor, e.g. HD camera, captures and streams image data to a decision entity responsible for visual quality checking. In addition to those critical flows, the supporting industrial network might carry other sources of traffic, such as other visual monitoring traffic that adds strain on the network bandwidth or web traffic that carries production data. Moreover, other traffic may also traverse the network for data display and system notifications.

Several research studies have investigated real-time data transmissions in TSN based industrial automation networks [7]. Silva et al. [8] evaluated the performance of TSN and Software-Defined Networks (SDN) in meeting the strict demands of Industry 4.0 applications on IIoT networks. The qualitative evaluation of their study concludes that TSN performs well, yet it faces some fundamental scalability challenges. The authors in [9] used Sercos III, a standardized open digital interface over TSN to support scheduled real-time protocols with high data rate and network extensions. Authors in [10], [11] developed TSN simulation modules for OMNET++ [12]. The developed modules can schedule traffic under configurations defined in terms of time aware shaping and frame preemption.

In this paper, we implement the application model of the quality checks after production use case in OMNET ++. In addition, we extend the TSN implementation in the INET 4.0 [13] framework to configure a supporting network layer for the application model. Moreover, we use three different test cases to evaluate our models as follows: a) worst case deterministic traffic with infinite queue size, b) worst case deterministic traffic with finite queue size, c) stochastic traffic models for realistic data flows with finite queue size of 1000 packets. Finally, we compare the performance of different

This publication has emanated from research conducted with the financial support of Science Foundation Ireland under Grant number 16/RC/3918. For the purpose of Open Access, the author has applied a CC BY public copyright licence to any Author Accepted Manuscript version arising from this submission

network configurations in meeting the QoS requirements, e.g. delay deadlines and delivery ratios. The paper is organized as follows: Section II introduces the application model and the characteristics of the generated data flows. Section III covers the network architecture and design to support the modelled application. The performance evaluation of our use case and network configurations is covered in section IV. Section VI provides conclusions and key points for future work.

II. Quality Checks After Production (QCAP)

Quality check is a typical process in any manufacturing system. It guarantees that the final product has passed the inspection tests with no deficiencies. A modern approach for running quality inspections is deploying a high-definition camera that capture visual data of the product under inspection, e.g., media files like images or a short video clip. The automation of quality control may be considered an IIoT use case with a layered architecture as depicted in Fig.1. For simplification, we consider the following three layers: 1) Device layer - a set of devices, sensors and actuators, collect data and perform actions according to the application layer configuration. 2) Network or communication layer - a set of network devices, e.g. switches, routers and network management servers, that carries the data and enables communication between devices of the device layer and the apps of the application layer. 3) Application layer - includes patches of the product under inspection, monitoring application, and system configuration application. In our scenario, this infrastructure runs the following operations for each production line, to execute the quality checks:

Object Detection: each production line is equipped with an object detection unit, e.g. presence sensor, that notifies the application layer of the arrival of a new product on the conveyor belt to trigger the quality checks process. The detection process is critical so as not to miss the passing product and requires strict upper bounded delay on delivering the trigger message. All detection flows are of high priority and classified as control data traffic. Product dimensions and arrival time have a direct influence on the status of a detection unit, either idle or active, which impacts on the characteristic of a detection flow.

Object Inspection: the visual inspection unit may monitor a single or multiple production line(s). The visual unit receives a request to inspect the recently arrived product, upon which it transmits a recording or set of images of the product to the processing unit to run quality checks. Such data flow has a bursty nature. As such, it affects the offered QoS, i.e. available bandwidth, of other data flows. In addition, it requires low jitter for the quality checks to run smoothly.

Object Classification: After processing the received data from the inspection unit, a control signal is sent to the robotic arm to classify the inspected product. A decision is made based on the quality checks, either pass or fail, which may trigger a robotic arm movement to remove the product. This data flow requires determinism as the check result has to be ready

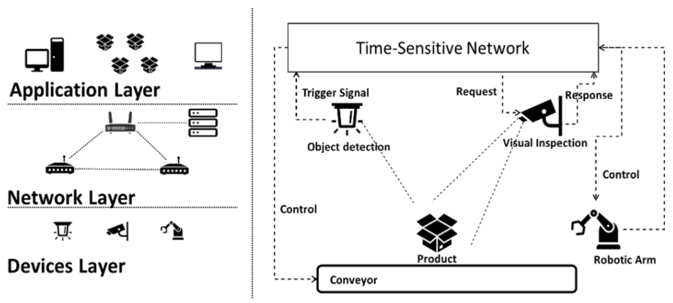


Fig. 1. Quality Check After Production (QCAP) use case.

before the robotic arm can be triggered, and the product has been moved on by the conveyor belt.

For each production line that adheres to the previous operations, a set of critical data flows have to be carried over the supporting industrial network. Additionally, to better utilize the inspection cameras, they may be used to monitor the production floor in between product inspections. This results in another class of data traffic, which targets better network utilization if scheduled efficiently. Moreover, web-based traffic may flow through the network to display for example recent data on production yields. The characteristics of all data flows in our QCAP use case are as follows:

Control Data Traffic (CDT): critical traffic with predefined low end-to-end latency. It is tagged with the highest priority code point (PCP) to be processed. The traffic parameters are:

1) Packet size = 174 bytes (128 payload + 46 headers).
2) Packet interval = 500 μs.
3) Max. data rate = 2.8 Mbps.
4) PCP = 6.
5) Max. allowed delay = 100 μs.
6) Idle time = $(3600 \times s - l \times \lambda)/\lambda \times s$, where $s(m/s)$ is the speed of the conveyor belt, $l(m)$ is the length of the product, and $\lambda(product/hour)$ is the number of product units per hour.

Class A Traffic: Such traffic is generated by the inspection unit once a request is received. It is tagged high priority critical traffic as it represents data of the inspected product. Therefore, it requires high reliability with guaranteed upper bounds on end-to-end delay. In addition, low jitter is essential as quality checks have to provide live results for the product to be classified. The traffic parameters are:

1) Packet size = Truncated Pareto distribution (α = 1.1, L = 180 bytes, H = 558 bytes), where L is the payload length and H the header size.
2) Packet interval = Truncated Pareto distribution (α = 1.1, L = 125μs, H = 500μs.)
3) Max. data rate = 35.7 Mbps.
4) PCP = 5
5) Max. allowed delay = 60 ms.
6) burst duration = 2.0 seconds.

Class B Traffic: to efficiently utilize the camera inspection units, they are also used to monitor the production floor, which results in a streamed video data flow that is directed to the monitoring unit. A single camera produces high resolution 24 frames per second. MJPEG compression is used to reduce video stream size while maintaining good image quality. Such traffic competes for network resources, e.g. bandwidth. This additional application offers a realistic stress test for the industrial network. The following parameters define the video traffic nature:

1) Packet size = Truncated Pareto distribution (α = 1.2, L = 20 bytes, H = 1246 bytes).
2) Packet interval = Truncated Pareto distribution (α = 1.1, L = 500μs, H = 5ms.)
3) Max. data rate = 19.94 Mbps.
4) PCP = 3
5) Max. allowed delay = 70 ms.

Best Effort Traffic: web traffic is pushed across the network layer to efficiently utilize the available resources. To model such data flows, we consider two types of applications, which are HTTP 1.0 based web browsing and file transfer protocol based data transfer. Both applications follow a random distribution for request arrival period and response size. We adopt and implement the models from [14] as detailed in the next section.

III. Network Design and Implementation

In this section, we present the network layer architecture and design. In addition, we present the implementation algorithms for the application models.

A. Network structure and design

We consider a hybrid, tree + star, network topology of five switches and 19 end devices. Every switch is configurable to operate a single, or mix, of the following queuing, scheduling or shaping techniques:

First In First Out (FIFO): An outgoing frame, through an egress port, is stored in a single queue. All queued frames are served in order of their arrival.

Priority Queuing (PRIO): An outgoing frame, through an egress port, is classified to be stored in a designated queue based on its priority. The PCP is 3-bit field in the VLAN (Virtual Local Area Network) tag, which is used to decide on the priority of the frame. The higher the PCP value, the higher the priority of a frame.

Time-Aware Shaping (TAS): Outgoing frames, stored in multiple queues, are controlled by a set of gates. Each gate is either ON, to allow a frame to be transmitted, or OFF to block a transmission from that queue. Gates are controlled by a schedule called the Gate Control List (GCL).

Credit-based Shaping (CBS): Outgoing frames, of a single queue, are controlled by the queue credit. Transmitting frames leads to spending credit with a *transmission slope* rate, while a queue obtains credit, with *idle slope* rate, by being idle and waiting for other queues to finish their transmissions. A queue stops transmitting when it has negative credit and may start transmitting again when it has zero or more credit.

End systems are used to run the implemented application models. Each end system generates different types of traffic

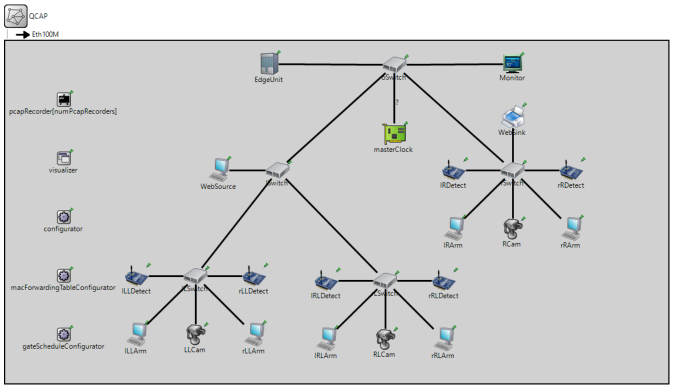


Fig. 2. Network structure for QCAP in OMNET++.

TABLE I
QCAP Data Flow Characterizations.

flow	type	Source	Sink	d_{max}
$f_1 ... f_6$	CDT	Sensors	Edge unit	100 μs
f_7, f_8, f_9	Class A	Cameras	Edge unit	60 ms
f_{10}, f_{11}, f_{12}	Class B	Cameras	Monitoring	70 ms
f_{13}	Class BE.	Web Source	Web Sink	-
$f_{14} ... f_{19}$	CDT	Edge unit	R.Arms	100 μs
f_{20}, f_{21}, f_{22}	Class A	Edge unit	Cameras	60 ms
f_{23}, f_{24}, f_{25}	Class B	Monitoring	Cameras	70 ms
f_{26}	Class BE.	Web Source	Web Sink	-

to perform the quality checks as previously stated in Section II. We classify the end systems as follows:

1) Detection sensors: six units are deployed and used to perform the detection part of QCAP.
2) Cameras: three units are deployed and used to perform the inspection part of QCAP. In addition, they perform live production floor monitoring.
3) Edge unit: single central unit that commands the inspection and classification units to perform their QCAP operations.
4) Monitoring: single unit that display all the streamed data flows from the monitoring cameras.
5) Robotic arms: six units are deployed to execute an action received from an Edge unit.
6) Web source/sink: Client/Server web traffic over the network.

The data flows generated by those end systems are summarized in table I. d_{max} represents the maximum allowed end-to-end delay that the frames of these flows can sustain.

B. Application model implementation

We implement six applications to model the behaviour of QCAP, which are as follows:

UDP burst source: modelled by a two-state machine, with states *sleep* and *burst*. While in *burst* state, the application generates a network packet every transmission interval t_{send} that is of a size equal to pkt_{size}. The application switches to *sleep* state after the burst duration ends, and reverts back to *burst* after time t_{sleep}. Pseudocode 1 depicts the main procedures to implement this application. The generated packets are sent through a UDP socket, with respective IP address and port numbers of a destination.

UDP sink: models a UDP (User Datagram Protocol) server that listens all the time to configured ports. Once a packet is received, it processes its content and initiates the appropriate procedure.

Pseudocode 1 UDP burst source

```
Input: start time = $t_{start}$, stop time = $t_{stop}$, burst duration = $t_{burst}$, sleep duration = $t_{sleep}$, transmission interval = $t_{send}$, destination address = $addr_{dst}$, destination port = $port_{dst}$.
Output: packet send = $pkt_{send}$, Number of sent packets = $nPKT_{sent}$
Require: $t_{start}, t_{stop}, t_{send}, t_{burst}, t_{sleep} \geq 0$
  & $t_{stop} \geq t_{start}$
  procedure START( )
      $nxtSleep = simTime();$
      $nxtBurst = simTime();$
      $nxtPkt = simTime();$
      $actvBurst = True;$
      $timer \leftarrow SEND$
      $GenerateBurst();$
  end procedure
  procedure GENERATE BURST($t_{stop}, t_{send}, t_{burst}, t_{sleep}$ )
      if $simTime() \leq t_{stop}$ then
          $now \leftarrow simTime();$
          if $nxtPkt < now$ then
              $nxtPkt \leftarrow now;$
          end if
          $nxtPkt+ = t_{send}$
          if $actvBurst$ & $nxtBurst \leq now$ then
              if $t_{burst} == 0$ then
                  $actvBurst = false;$
              else
                  $nxtSleep = now + t_{burst};$
                  $nxtBurst = nxtSleep + t_{sleep};$
              end if
          end if
          $pkt_{send}(addr_{dst}, port_{dst});$
          $nPKT_{sent} + +;$
          if $actvBurst$ & $nxtPkt \geq t_{stop}$ then
              $nxtPkt = nxtBurst;$
          end if
          if $t_{stop} \geq 0$ & $nxtPkt \geq t_{stop}$ then
              $timer \leftarrow STOP$
              $nxtPkt = t_{stop};$
          end if
          $ScheduleAt(nxtPkt, timer);$
      end if
  end procedure
```

Video streaming client: models UDP client. It transmits a single request to a unique destination address and port number, and it expects to receive the streamed data from the unique address. Request re-transmission occurs when the server is not receiving data for a period of t_{idle}.

Video streaming server: implements UDP server process. The UDP server receives a request for data streaming and replies with a stream of packets of a configured file size according to the Class B traffic model in Section II.

TCP client: models the client process of a generic TCP request/response service using sessions. It opens a TCP connection to a configured destination address, then transmits several requests. The connection remains open until the complete data arrives before sending a new request.

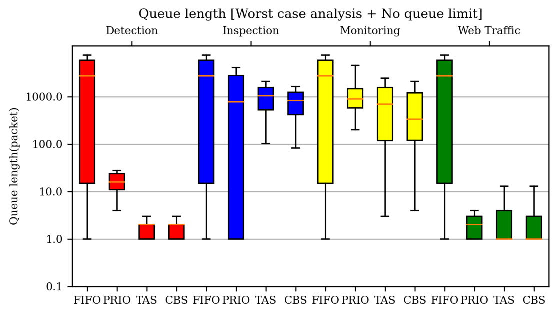
TCP server: models the server process of a generic TCP request/response service using sessions. It listens to a configured port and expects to receive connection requests. After processing a request, it replies with the same message but with a different length according to the request. Replies could be delayed by a configured constant time.

Each end system of the network in Fig. 2, runs single or multiple instances of those applications, which results in the data flows described in Table I. To model the QCAP use case, we configure the end systems as follows:

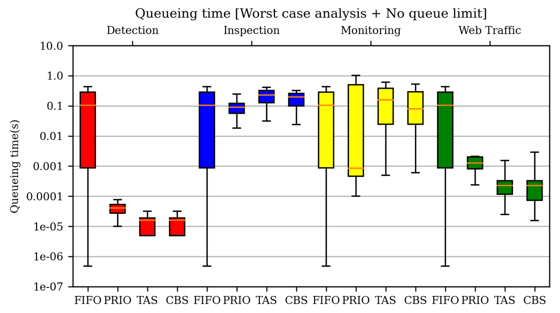
1) Detection sensor: runs a single UDP burst application to generate traffic upon detecting a new product and then proceeds to sleep until the next product arrival.
2) Camera: runs 2 applications: 1) UDP burst: to transmit inspection data upon receiving a request. 2) Video streaming server: to monitor the production floor of the associated production lines.
3) Edge unit: runs 18 instances of two application: 1) UDP sink: to receive data from the detection sensors and inspection cameras. 2) UDP burst source: to send triggers to the inspection units and the classification robotic arms.
4) Monitoring: runs three instances of a single application, which is a video streaming client to send requests to each camera.
5) Robotic arm: runs a single UDP sink application to listen for requests to classify a product.
6) Web source/sink: one runs a TCP server and the other runs a TCP client to exchange web requests and replies.

IV. Performance Evaluation

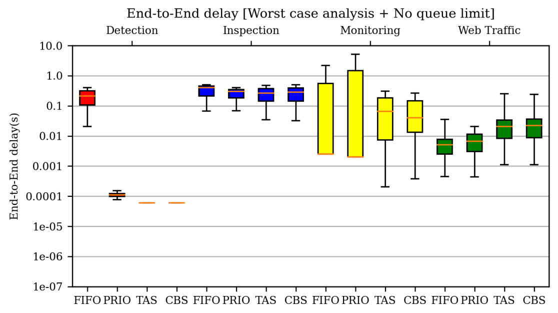
We evaluate our work by running the network and application implementations in Section III. We use the parameters listed in Table II to configure our simulation. To thoroughly evaluate the different network configurations and assess which is the optimal one, we simulate three different test cases. First, we consider the worst case scenario where all flow sources are transmitting with the maximum data rate. Therefore, we can assess the state of an overloaded network. We consider two scenarios here, one with infinite queue sizes, and the other with limited queue size of maximum 1000 packets. Both cases consider deterministic traffic, which ease the computation of transmission schedules. For the third test case, we consider stochastic traffic models. We assess the network performance in terms of end-to-end delay and delivery ratio for different network switch configurations. In addition, we analyse the behaviour of the overloaded network bottleneck link, which is the link between the root switch and the left switch as most



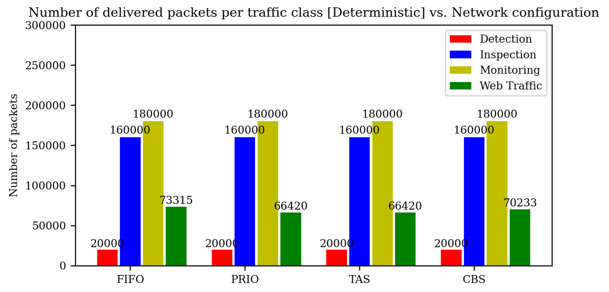
(a) Queue Length (packet)



(b) Queueing time (s)



(c) End-to-End delay (s)



(d) # of Delivered packets

Fig. 3. Worst Case with infinite queue size (Deterministic traffic)

TABLE II
NETWORK AND SIMULATION PARAMETERS

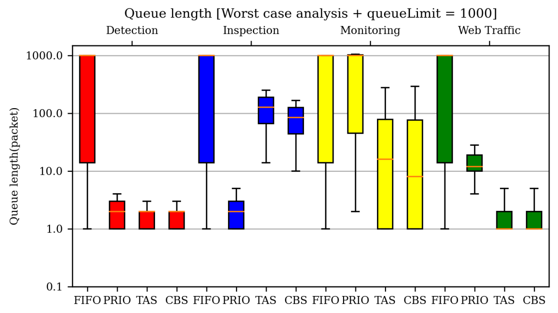
Parameter	value
Simulation time	90 s
Processing delay of a switch	1 μs
Network link speed	100 Mbps
Propagation delay	0.05 μs

of the data flows pass through it. We investigate the lengths of the queues and the queuing times for this link.

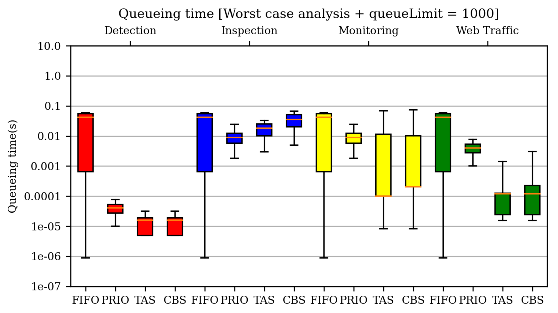
Fig. 3 shows the results from the first test case. First, Figs. 3.a and 3.b provide insights into the behaviour of the congested link. For the FIFO configuration, different types of traffic are stored in the same queue and experience the same amount of queuing delay. In contrast, the PRIO configuration classifies different data flows, which leads to a smaller queue length for high priority traffic and therefore much less queuing time. However, the queue length of lower priority queues, such as for monitoring data flows, is still high with the same for the queuing time. The reason is that Class A flows are flooding the network, which prevents Class B traffic from being delivered. Moreover, it blocks the web replies, leading to fewer requests to be made by the web client.

Next we assess the end-to-end delay, Fig 3.c, for the different types of data flows, against the different network configurations. As expected, all different configurations, except FIFO, provide within range acceptable delay for the detection system. However, the other types of traffic experience higher delays than the maximum allowed. The delay becomes significantly high for monitoring traffic, Class B, when PRIO configurations are used. Finally, we measure the number of delivered packets in Fig. 3.d. The different network configurations manage to achieve 100% delivery ratio for the critical, class A, and class B traffic. However, the number of delivered packets of the web traffic, class BE, is of interest, as it indicates the capability of the network to handle the coexistence of both operational technology and information technology traffics. The FIFO configuration delivers the highest number of best effort packets, but at the expense of end-to-end delay. In contrast, PRIO and TAS configurations deliver 90.6% of packets, which happens due to oppressing or delaying web client requests by higher priority traffic or by controlling gates with a priority transmission schedule. The CBS configuration performs better by successfully delivering 96% packets, compared to FIFO, of the web traffic. This is due to the credit shaping where class A and class B have to gain credit before transmitting, which allows class BE traffic to be successfully transmitted.

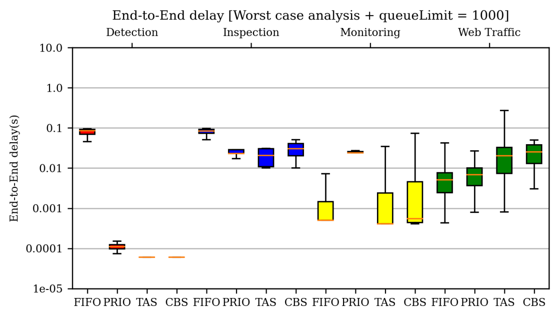
In the second test case, we limit the queue size for the egress ports of all the network switches. Queues are configured with a drop tail mechanism. Fig. 4 shows the results for this case. First, Figs. 4.a and 4.b depict the distribution of the queue length and queuing time over the simulated period. The PRIO configuration significantly decreases the queuing time for the critical traffic, same for TAS and CBS, when compared to FIFO, which is a good indication that they meet the QCAP's deadlines requirements. However, PRIO performance drops compared to TAS and CBS while handling lower priority traffic. We assess the overall performance with end-to-end



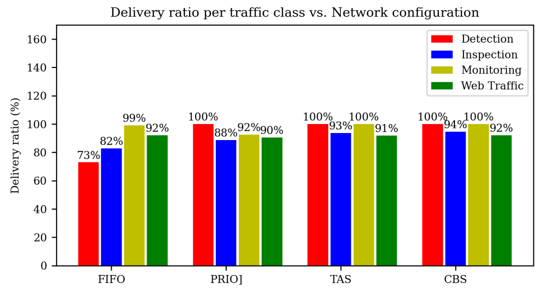
(a) Queue Length (packet)



(b) Queueing time (s)



(c) End-to-End delay (s)



(d) Delivery ratio (%)

Fig. 4. Worst Case with limited queue size (deterministic traffic, queue size = 1000)

	Standard Ethernet [FIFO]		Standard Ethernet [PRIO]		TSN [TAS]		TSN [TAS+CBS]	
	Avg. end-to-end delay (s)	Delivery ratio (%)	Avg. end-to-end delay (s)	Delivery ratio (%)	Avg. end-to-end delay (s)	Delivery ratio (%)	Avg. end-to-end delay (s)	Delivery ratio (%)
CDT	75×10^{-3}	73	111×10^{-6}	100	61×10^{-6}	100	61×10^{-6}	100
Class A	79×10^{-3}	82.8	49×10^{-3}	88.7	35×10^{-3}	93.7	48×10^{-3}	94.6
Class B	13×10^{-3}	99.1	61×10^{-3}	92.6	5.7×10^{-3}	100	6.5×10^{-3}	100
Class BE	78×10^{-3}	92.1	10×10^{-3}	90.6	24.5×10^{-3}	91.9	24×10^{-3}	92.1

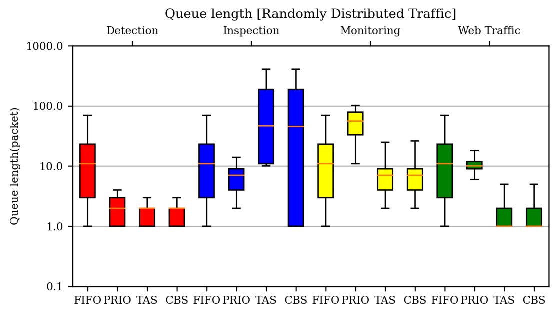
TABLE III
SUMMARY OF THE AVERAGE END-TO-END DELAY AND DELIVERY RATIO OF QCAP SIMULATION RESULTS.

delay and delivery ratio measurements as depicted in Figs. 4.c and 4.d and summarized in Table III. While PRIO prioritizes the critical traffic, it could not meet the deadline requirements. In contrast, both TSN configurations provide deterministic upper bounds that are below the QCAP deadlines. Moreover, using only the PRIO configuration, for the monitoring system, a higher end-to-end delay is shown. This is due to the bursts in class A traffic, yet TAS and CBS configurations provide better control over those and therefore demonstrate overall better end-to-end delay for all the subsystems. In addition, CBS provides the highest delivery ratios for all the subsystems and the web traffic.

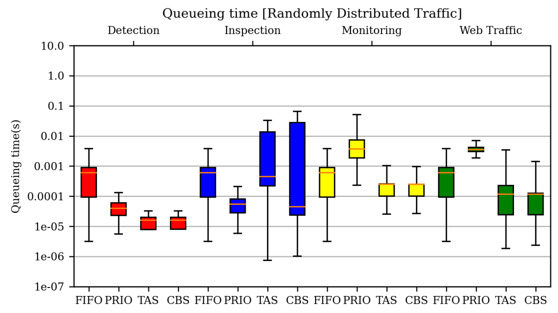
Finally, we assess our system with realistic stochastic traffic models to measure the resilience of the TAS and CBS configurations against the random changes in the data flows behaviours. Fig. IV.a depicts the queue length distribution for the TAS and CBS. We can see that TAS and CBS outperform the other configurations, except for the inspection process. This is due to the bursty nature of class A traffic. Fig. IV.b shows the queuing times at the congested link for the different system processes. For TAS and CBS, detection traffic is below 100μs, inspection, monitoring and web traffics are below 100ms. Finally, Figs IV.c and IV.d present the two main measures, which concludes that using TAS and CBS guarantee reliable delivery for the critical traffic with a deterministic end-to end latency. In addition, they provide better utilization for the network resources by allowing web traffic to coexist without affecting the QoS of the critical traffic.

V. Conclusion and Future Work

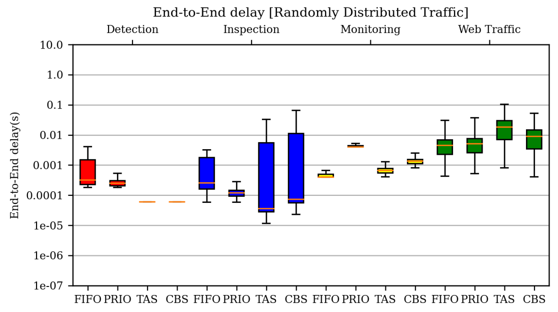
In this study, we model a typical IIoT application, the quality check using visual inspection after production use case. The objective is to measure the performance of different network configurations, e.g., FIFO, PRIO, TAS and CBS, to support the different QoS requirements of the several data flows within this application. A set of application models have been implemented to mimic the data flows behaviour. The conducted experiments are a set of simulations using OMNET+ for three different test cases. The test cases consider the worst case scenarios, queuing size, and the network traffic nature. The simulation results indicate that using both TSN configurations, time aware traffic shaping to schedule the



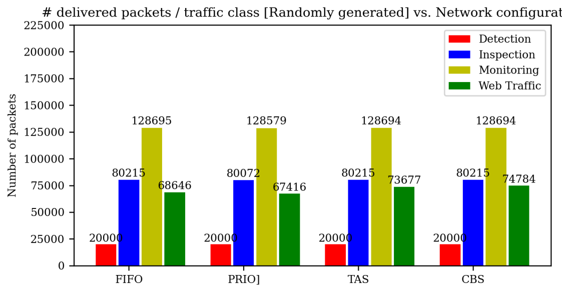
(a) Queue Length (packet)



(b) Queueing time (s)



(c) End-to-End delay (s)



(d) # of Delivered packets

Fig. 5. Randomly generated traffic (queue size = 1000))

traffic and credit based traffic shaping to control the data rates of traffic bursts, can satisfy the delay requirements of the implemented use case. The experimental results illustrate that the average end-to-end delay of the critical traffic has a guaranteed value of 61μs with no losses. In addition, TSN [TAS + CBS] outperforms the other configurations in many aspects, including the coexistence of more web traffic while meeting the QoS requirement of the automation system traffic. The computation of the TSN schedules and shaping techniques are NP-hard [15] problems, and finding scalable solutions for those motivates our future work.

Disclosure and conflicts of interest

The authors declare that there is no conflict of interest.

References

[1] ”Time-Sensitive Networking Task Group” . [Online]. Available: https://www.ieee802.org/1/pages/tsn.html

[2] “P802.1AS-Rev – Timing and Synchronization for Time-Sensitive Application” . [Online]. Available: https://1.ieee802.org/tsn/802-1as-rev/

[3] “802.1Q-2014 - Bridges and Bridged Networks”. [Online]. Available: https://www.ieee802.org/1/pages/802.1Q-2014.html

[4] ”Standard for Local and Metropolitan Area Networks-Media Access Control (MAC) Bridges and Virtual Bridged Local Area Networks Amendment: Enhancements for Scheduled Traffic”. [Online]. Available: https://www.ieee802.org/1/pages/802.1bv.html

[5] ”IEEE Standard for Local and Metropolitan Area Networks—Virtual Bridged Local Area Networks - Amendment: Forwarding and Queuing Enhancements for Time-Sensitive Streams.” [Online]. Available: https://www.ieee802.org/1/pages/802.1av.html.

[6] ”Standard for Local and Metropolitan Area Networks-Media Access Control (MAC) Bridges and Virtual Bridged Local Area Networks - Amendment: Frame Preemption” [Online]. Available: https://www.ieee802.org/1/pages/802.1bu.html

[7] Li, Jian-Qiang, F. Richard Yu, Genqiang Deng, Chengwen Luo, Zhong Ming, and Qiao Yan. ”Industrial internet: A survey on the enabling technologies, applications, and challenges.” IEEE Communications Surveys & Tutorials 19, no. 3 (2017): 1504-1526.

[8] L. Silva, P. Pedreiras, P. Fonseca and L. Almeida, ”On the adequacy of SDN and TSN for Industry 4.0,” 2019 IEEE 22nd International Symposium on Real-Time Distributed Computing (ISORC), 2019, pp. 43-51, doi: 10.1109/ISORC.2019.00017.

[9] S. Nsaibi, L. Leurs and H. D. Schotten, ”Formal and simulation-based timing analysis of Industrial-Ethernet sercos III over TSN,” 2017 IEEE/ACM 21st International Symposium on Distributed Simulation and Real Time Applications (DS-RT), 2017, pp. 1-8, doi: 10.1109/DISTRA.2017.8167670.

[10] J. Jiang, Y. Li, S. H. Hong, M. Yu, A. Xu and M. Wei, ”A Simulation Model for Time-sensitive Networking (TSN) with Experimental Validation,” 2019 24th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA), 2019, pp. 153-160, doi: 10.1109/ETFA.2019.8869206.

[11] J. Falk et al., ”NeSTiNg: Simulating IEEE Time-sensitive Networking (TSN) in OMNeT++,” 2019 International Conference on Networked Systems (NetSys), 2019, pp. 1-8, doi: 10.1109/NetSys.2019.8854500.

[12] OMNeT++ Discrete Event Simulator. [Online]. Available: https://www.omnetpp.org/. Accessed on: Jul. 1, 2022.

[13] M´esz´aros, Levente ”Time Sensitive Networking in INET 4.4”. Available: https://summit.omnetpp.org/2021/assets/pdf/slides-omnetpp6-overview.pdf.

[14] Traffic Model for M2M Services, document TSGRAN WG2 Meeting #68-bis R2-100204, 3GPP, Sophia Antipolis, France, Jan. 2010. [Online]. Available: http://www.3gpp.org/ftp/tsg_ran/WG2 RL2/TSGR2 68bis/docs/R2-100204.zip

[15] Dürr, Frank, and Naresh Ganesh Nayak. ”No-wait packet scheduling for IEEE time-sensitive networks (TSN).” In Proceedings of the 24th International Conference on Real-Time Networks and Systems, pp. 203-212. 2016.