Towards Developing a Supervisory Agent for Adapting the QoS Network Configurations

Sebastian Leclerc[1], Kasra Ekrad[1], Bjarne Johansson[1,2],
Inés Alvarez Vadillo[2], Mohammad Ashjaei[1], Saad Mubeen[1]
[1]Mälardalen University, Västerås, Sweden [2]ABB AB, Västerås, Sweden
{sebastian.leclerc, kasra.ekrad, bjarne.johansson, mohammad.ashjaei, saad.mubeen}@mdu.se
{bjarne.johansson, ines.alvarez-vadillo}@se.abb.com

***Abstract*—As the industry trend is moving toward Ethernet-based Industrial Control Systems (ICSs) in line with the industry 4.0 paradigm, the network traffic must be monitored and managed to keep the systems reliably available. A flat converged network architecture where all the controllers, Input/Output (I/O), and supervisory systems are connected introduces mixed traffic classes competing for network resources and access to the shared medium. These traffic classes are typically managed with different Quality of Service (QoS) techniques. The challenge lies in managing end-to-end QoS in a heterogeneous environment and configuring switches according to the vendor, model, and operating software. Meanwhile, emerging greenfield technology are not deployable with switches used in today's ICS. We propose a distributed QoS supervisory agent, capable of detecting and correcting faulty QoS configurations in a contemporary heterogeneous Layer 2 switch environment. The initial results are achieved through two experimental testbeds in a use case where our agent successfully corrected the QoS configurations.**

***Index Terms*—QoS Supervision, Faulty QoS Detection and Correction, Industrial Control Systems**

I. INTRODUCTION

Industrial Control Systems (ICSs) are responsible for many complex safety-critical processes and require sufficient dependability to operate, remain profitable, and maintain reliability [1]. The industry trend is shifting from the automation pyramid architecture in which different parts of the industrial infrastructure are isolated from each other, toward using Ethernet and a network-centric architecture connecting everything, to be inline with the industry 4.0 advancement [2], [3]. This architecture leads to mixed traffic classes using the shared network resources, e.g., bandwidth resources [4]. Regardless of this paradigm change, the ICSs often interact with the real world and must be carefully designed to ensure that the correct control output is produced at a bounded time, i.e., in real-time manner [1]. Commonly, the traffic transmitted from industrial applications is predetermined, designed, or scheduled to ensure that their deadlines are met. This implies that an expected rate of the traffic can be known beforehand. Ethernet-based networks typically manage these traffic classes by utilizing Quality of Service (QoS), configured from end-to-end according to the traffic requirements.

Even if the industrial application correctly prioritizes different traffic classes during transmission, the receiving switch will not adhere to the priority levels if it is not configured correctly [4]. Fig. 1. illustrates traffic referred to three priorities that reach a switch without QoS configuration and are forwarded with a varying bandwidth utilization over time without consideration of requirements. Misconfigurations of the QoS can be difficult to detect and may lead to costly faults and unwanted downtime [5]. This raises the question of how faults in QoS configuration can be detected and mitigated in a network.

Adding to the situation's complexity, the QoS configuration must be correct from end-to-end perspective, ideally, deployable across diverse network device vendors. Different approaches exist to tackle the QoS problem. A typical method employed in the Information Technology (IT) domain is to overprovision the resources in networks [6]. However, this fails to offer the specific traffic guarantees required in an ICS. Furthermore, this method is not scalable, as when the traffic load increases, it is no longer possible to provide enough bandwidth for overprovisioning without changing the complete infrastructure. Instead, Traffic Engineering (TE) can aid the end-to-end QoS by supervising and optimizing the network according to traffic requirements. The supervision can be centralized [6], [7], [8], which introduces a single point of failure. Distributed supervision has also been suggested by mirroring traffic to external nodes [5], which requires additional hardware. The QoS supervision is often based on greenfield technology which is slowly being adopted and is not employable in *today's* industrial switches, e.g., Software-Defined Networking (SDN) [6], [9], In-band Network Telemetry (INT) [10], [11], or Time-Sensitive Networking (TSN) [7], [8], to name a few.

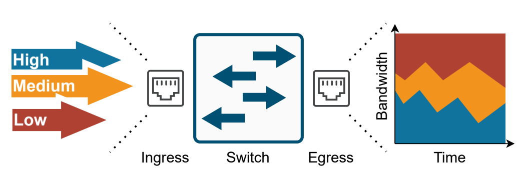


Fig. 1. Mixed traffic classes reaching the ingress port of a switch, forwarded out the egress port by a switch without any QoS configuration.

Contributions: To address the above-mentioned challenge, this paper presents a mechanism to detect faulty QoS configurations in *contemporary* industrial switches, thus enabling error detection to prevent costly failures. This mechanism is aimed at contributing to more reliable and fault-tolerant Ethernet-based ICS. We propose a distributed QoS supervisory agent, actively monitoring the industrial switches' egress queues from multiple points in the topology to detect and mitigate faulty QoS configuration in a heterogeneous industrial Layer 2 Ethernet network environment. The proposed technique could be implemented on networks based on switches, such as EtherNet/IP or PROFINET. The mechanism is designed with consideration of real-time requirements in the network, as will be explained later. We have also implemented and preliminary evaluated the mechanism, showing its feasibility and correctness in a small industrial use case. In our design, the supervisory agent polls the information from switches, enabling scheduling and calculation of worst-case execution time without jeopardizing the network operation.

II. Related work

Previous works propose various methods of managing QoS in different domains. A number of methods are proposed in both IT and Operational Technology (OT) is via SDN. One of the works within IT [6] proposed an open-source SDN-based QoS end-to-end monitoring tool, which adaptively polls switches from a centralized node, for TE based on various metrics. Another SDN-based approach in the OT domain [9] used a simulated network environment to manage the optimal network paths for two traffic flows according to their QoS requirements. To our best knowledge, SDN is yet not supported by today's industrial switches and is not commonly used in ICS use cases.

INT is another research area for near real-time network visibility. The framework tags the traffic with telemetry information, which in turn can be used to adjust the network configuration. The cost of INT is an additional overhead in the data plane. The work in [10] developed two accuracy-adaptive and lightweight INT-based mechanisms that were evaluated on network traces. Another work that addresses the overhead is [11], which developed a probabilistic framework that bounds the INT data for congestion control. The major downside of the INT-based solutions is that they require compatible hardware (Programming Protocol-independent Packet Processor (P4)) and are not deployable in today's switches with fixed-function Application Specific Integrated Circuits (ASICs).

Moreover, various works have dealt with QoS management through TSN. For example, the work in [7] proposes a specification for asynchronous TSN-compliant industrial network rings, using a centralized QoS management approach. The performance results are preliminary and have not yet been evaluated in hardware in an industrial use case. Similarly, the work in [8] proposes a centralized end-to-end QoS management approach for industrial applications over TSN, capable of handling link failures. This work was evaluated on TSN-compliant hardware and managed by Network Configuration Protocol (NETCONF), which none of our investigated industrial switches supported.

There are also a few works that propose distributed passive monitoring by mirroring network traffic to detect QoS violations, e.g., in [5]. These approaches interfere less with the ICS traffic, while they require additional hardware to perform (e.g., additional network taps, cables, and analyzing nodes).

III. QoS Supervisory Agent Design

This section first presents our investigation for methods that can access the network devices information, suitable for our supervisory agent solution. Then it presents the design of the supervisory agent.

A. Existing methods to access the network

To support various traffic class requirements, QoS is used to classify and act upon matching flows [4]. Typical actions include changing the priority of traffic flows, dropping traffic via policing, or smoothing peak bandwidth utilization via shaping. Two of the common approaches to prioritize and mark traffic flows are via Differentiated Services (DiffServ) and Class of Service (CoS). DiffServ uses the Internet Protocol (IP) Version 4 header's Differentiated Services Code Point (DSCP) field in Layer 3. CoS uses the Priority Code Point (PCP) field in the Virtual Local Area Network (VLAN) tag in Layer 2. DSCP allows 64 different priority levels, while PCP allows only eight. Network devices can commonly be configured to both trust and translate between Layer 2 and Layer 3 markings as needed. The challenge with using CoS for end-to-end prioritization is that it requires compatible Network Interface Cards (NICs) for interpreting the VLAN tags, which are otherwise stripped on the so-called access links such as (A) and (C) in Fig. 4. Nevertheless, the prioritization can either occur in the sending software application or the switch may be configured to (re-)prioritize the traffic. Once the prioritized traffic reaches the switch, it can use various scheduling algorithms, such as strict priority scheduling, to put the prioritized traffic to the associated egress queue on the port toward its destination. Switches support a defined number of egress queues per port, e.g., four or eight. Note that this limits the granularity of usable priority levels, making the 64 possible DSCP values excessive as only four or eight are needed.

One method of configuring a switch's QoS is through numbered Access Control Lists (ACLs) [4]. All traffic that reaches the switch is evaluated whether it matches the ACLs sequentially in ascending order. If a match is found, the ACLs can be configured to perform actions such as dropping the traffic or (re-)prioritizing it, subsequent ACLs are not checked for the same traffic. However, exactly what QoS configuration to apply depends on the environment. For example, a rule could be created to limit Transmission Control Protocol (TCP) traffic from a particular IP:Port to 500 kbps, another rule could down-prioritize User Datagram Protocol (UDP) traffic on a distinct physical port, or perhaps a rule could drop all traffic matching a protocol EtherType value. In other words, there are a vast number of configuration possibilities. General methods of remotely configuring a switch's QoS in-band include Command Line Interface (CLI) over Telnet or

Secure Shell Protocol (SSH), or through Simple Network Management Protocol (SNMP), NETCONF, and SDN.

QoS configuration adherence can be supervised by collecting and analyzing the network device's data, such as *statistical data* of the number of Received (Rx) bytes, or Transmitted (Tx) packets in a certain switch ports queue [4], [5]. This can be achieved through active or passive monitoring approaches. Active monitoring proactively and periodically polls the network devices and evaluates the responses. Passive monitoring reactively analyses network traffic. Consequently, active monitoring consumes more network resources while passive monitoring is more resource-intensive on the monitoring system having to analyze all traffic.

B. The supervisory agent design

Calculating an *expected traffic rate* in each queue is straightforward based on the previously mentioned statistical data and the predetermined QoS policy. This laid the foundation for the agent whose flowchart is depicted in Fig. 2. The agent polls a Tx or Rx queue counter at some interval to evaluate if the expected traffic rate threshold is approximately met, i.e., increasing as expected. If the environment is non-real-time, the measurements will be affected by jitter to a greater extent. Additionally, if the agent and switch are out of sync, the agent may fail to poll the most recent data. Therefore, this polling and subsequent evaluations are performed several times in a row to get better agent confidence in its decision-making, i.e., whether the agent should act upon unexpected flows or not. If the agent discovered that the threshold was less than or greater than expected for several evaluations, the QoS is adjusted by pushing appropriate ACLs (or other configurations, implemented examples found in the GitHub repository[1]) to the monitored switch. All CLI commands[1] are maximally shortened according to what the switch allows, to minimize the request overhead on the network. Note that we do not consider the switch queue depth and overflows for the agent, we instead use the static expected traffic rate.

IV. Preliminary Implementation and Results

We investigated up to ten industrial Layer 2 switch model datasheets from five vendors, Cisco, Red Lion, Moxa, Hirschmann, and Westermo. Based on this investigation, four switches were acquired as listed in Table I. We found that Telnet, SSH, and different versions of SNMP were the common denominators of remote management protocols to configure the devices. Configuration through Telnet and SSH involves using CLI commands specific to the switch model and its installed software. Configuration via SNMP requires vendor-specific Management Information Bases (MIBs) for data interpretation from an external agent. A subset of the investigated switches contained the statistical data as seen in Fig. 3. The approximate update frequency of this data was limited to every second on our switches. Our implementation details, switch configuration, and used CLI commands can be found in the GitHub repository[1].

[1]https://github.com/SebastianLeclerc/QoS-Adherence-Supervision

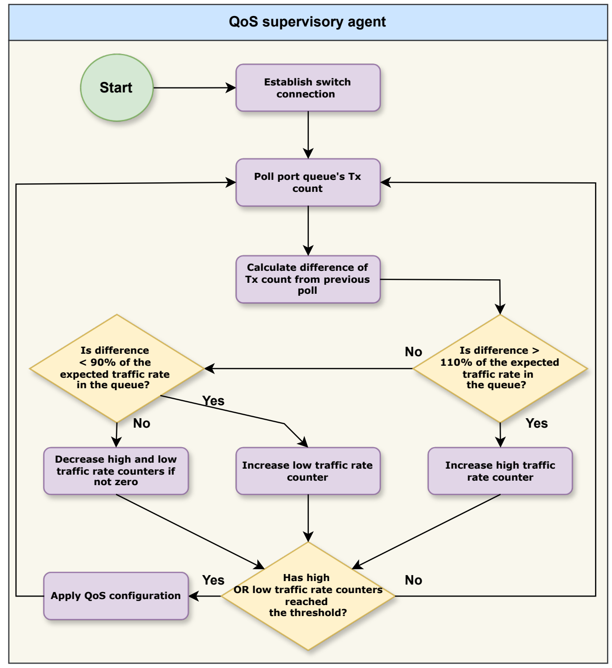
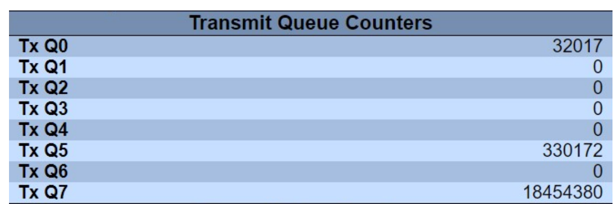


Fig. 2. Flowchart of the implemented QoS supervisory agent.

TABLE I
THE TESTED L2 INDUSTRIAL NETWORK SWITCHES.

Vendor	**Model**	**Operating software**
Cisco	IE-2000-8TC-L	IOS 15.2(7)E2
Hirschmann	BRS40 8TX-EEC	HiOS-2S-09.6.00
Red Lion	NT-4008-000-PN-M	Firmware 1.0.9
Westermo	NE810	NEOS 4.20.0-r0



Transmit Queue Counters	
Tx Q0	32017
Tx Q1	0
Tx Q2	0
Tx Q3	0
Tx Q4	0
Tx Q5	330172
Tx Q6	0
Tx Q7	18454380

Fig. 3. Tx queue counters from a Red Lion switch port.

Two non-real-time experimental testbeds were set up to emulate an ICS use case with mixed traffic classes as shown in Fig. 4. We emulated a redundant controller deployment, where one controller acted as the primary and the other as the warm standby secondary controller. The controllers were connected over an Ethernet network comprised of two switches: Red Lion (left) and Cisco (right). The primary controller's software sent High Priority (HP) UDP traffic by marking it with an HP DSCP value of 56. This value was chosen since its binary value of $0b111000$ is trivial to transform into an HP PCP value of seven, i.e., $0b111$ using the three most significant bits. The expected rate of HP traffic was estimated to 1050 packets/s. Medium Priority (MP) management traffic configuring the switch consisted of Telnet using its default port 23 for experimental purposes. The Low Priority (LP) best-effort

UDP traffic was generated at different rates with iPerf between the client and server from either the primary or in a separate node. A QoS policy was designed to assign HP traffic to the switches HP egress queue (Q7), MP management traffic in an MP queue (Q6), and LP traffic in an LP queue (Q0).

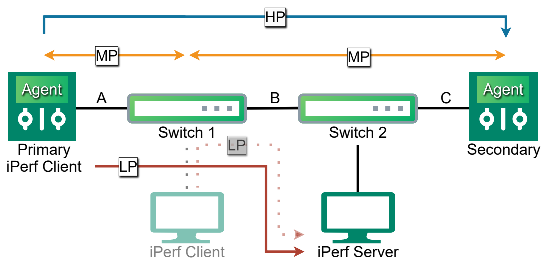


Fig. 4. The two different experimental testbeds where the QoS supervisory agent was implemented.

The agent was tested in both testbeds using three different QoS configurations, attempting to congest the link (A) and (B) when the iPerf Client was located in the primary or (B) when separated, seen in Fig. 4. The tested QoS configurations were as follows: (i) the expected configuration (ii) HP traffic incorrectly assigned to the LP queue, and (iii) LP traffic incorrectly assigned to the HP queue. Results show that it was possible to monitor and correct the two wrong QoS configurations in both testbeds while congesting the network by 949 Mbps over the 1 Gbps links.

Note that the implemented agent, test case, and testbeds were simplified by only managing three distinct traffic classes. Sufficient details about a particular traffic class are necessary to single it out through an ACL. However, it is possible to use "catch-all" ACLs to some extent which are evaluated last, matching non-HP traffic, and allocating it to an appropriate LP queue. Specifically, two ACLs could be constructed and placed at the end of the list, matching any other best-effort TCP or UDP traffic and marking them with a PCP value of 0 or dropping them. Nevertheless, what QoS configuration to apply is complex, error-prone, and dependent on the environment, i.e., one should first evaluate existing traffic and its requirements, and then apply relevant rules. Our concept works for IT and OT switches of today but must be adapted to specific switch models. It also requires statistical data, i.e., Rx or Tx queue counters, which half of the switches we investigated lacked. This makes monitoring QoS in today's network environments challenging and time-consuming.

V. Conclusions and Ongoing Work

We showed that our QoS supervisory agent could both detect and correct faulty configurations in an industrial Layer 2 Ethernet switch environment. Our preliminary results show that the agent could detect and resolve wrong QoS configuration while the network was stressed through iPerf up to 949 Mbps over the 1 Gbps links. If the network was congested more, the agent would get stuck in either polling the statistical data or pushing a new configuration. This state could be used to inform the supervisory system of a greater ongoing network issue. Distributing the agent over multiple nodes, however, increases the likelihood that one agent can detect and correct a switch's faulty configuration while also enabling agent awareness of missing switch replies. Although the switch polling frequency was limited to one second, the method can still be used to detect early symptoms of a wrongly configured network. The lack of standardization requires adapting the agent for each switch, but we show what is possible with suitable switches commonly deployed in today's industries. The ongoing work aims at developing the agent in more complex and scalable testbeds to investigate its overhead and performance in greater detail.

Acknowledgements

This work is supported by the Swedish Agency for Innovation Systems and the Knowledge Foundation via the ISECURE and SEINE projects.

References

[1] I. Álvarez, A. Ballesteros, M. Barranco, D. Gessner, S. Djerasevic, and J. Proenza, "Fault Tolerance in Highly Reliable Ethernet-Based Industrial Systems," *Proceedings of the IEEE*, vol. 107, no. 6, pp. 977–1010, May. 2019, doi: 10.1109/JPROC.2019.2914589.

[2] T. Carlsson, "Industrial network market shares 2023," *hms-networks.com*, May., 2023. [Online]. Available: https://www.hms-networks.com/news-and-insights/news-from-hms/2023/05/05/industrial-network-market-shares-2023. (accessed 2024-04-04).

[3] B. Leander, B. Johansson, T. Lindström, O. Holmgren, T. Nolte, and A. V. Papadopoulos, "Dependability and security aspects of network-centric control," in *2023 IEEE 28th International Conference on Emerging Technologies and Factory Automation (ETFA)*, 2023, pp. 1–8, doi: 10.1109/ETFA54631.2023.10275344.

[4] T. Szigeti, R. Barton, C. Hattingh, and J. K. Briley, *End-to-End QoS Network Design Quality of Service for Rich-Media & Cloud Networks, Second Edition*. Indianapolis, IN: Cisco Press, 2014.

[5] Y. Won, M.-J. Choi, B. Park, and J. W.-K. Hong, "An approach for failure recognition in IP-based industrial control networks and systems," *International Journal of Network Management*, vol. 22, no. 6, pp. 477–493, 2012, doi: 10.1002/nem.1804.

[6] N. L. M. van Adrichem, C. Doerr, and F. A. Kuipers, "OpenNetMon: Network monitoring in OpenFlow Software-Defined Networks," in *2014 IEEE Network Operations and Management Symposium (NOMS)*, 2014, pp. 1–8, doi: 10.1109/NOMS.2014.6838228.

[7] A. Mifdaoui, J. Lacan, A. Dion, F. Frances, and P. Leroy, "FactoRing: Asynchronous TSN-compliant Network with low bounded Jitters for Industry 4.0," in *2021 26th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, 2021, pp. 1–8, doi: 10.1109/ETFA45728.2021.9613631.

[8] A. Garbugli, A. Bujari, and P. Bellavista, "End-to-end QoS Management in Self-Configuring TSN Networks," in *2021 17th IEEE International Conference on Factory Communication Systems (WFCS)*, 2021, pp. 131–134, doi: 10.1109/WFCS46889.2021.9483600.

[9] N. N. Josbert, W. Ping, M. Wei, M. S. A. Muthanna, and A. Rafiq, "A Framework for Managing Dynamic Routing in Industrial Networks Driven by Software-Defined Networking Technology," *IEEE Access*, vol. 9, pp. 74 343–74 359, May. 2021, doi: 10.1109/ACCESS.2021.3079896.

[10] S. R. Chowdhury, R. Boutaba, and J. François, "LINT: Accuracy-adaptive and Lightweight In-band Network Telemetry," in *2021 IFIP/IEEE International Symposium on Integrated Network Management (IM)*, 2021, pp. 349–357.

[11] R. Ben Basat, S. Ramanathan, Y. Li, G. Antichi, M. Yu, and M. Mitzenmacher, "PINT: Probabilistic In-band Network Telemetry," in *Proceedings of the Annual Conference of the ACM Special Interest Group on Data Communication on the Applications, Technologies, Architectures, and Protocols for Computer Communication*, 2020, pp. 662—-680, doi: 10.1145/3387514.3405894.