Edge-Powered In-Network Processing for Content-Based Message Management in Software-Defined Industrial Networks

M. Fogli, C. Giannelli and C. Stefanelli

ICC 2022 - IEEE International Conference on Communications, 2022

Cite this as

M. Fogli, C. Giannelli and C. Stefanelli, "Edge-Powered In-Network Processing for Content-Based Message Management in Software-Defined Industrial Networks," *ICC 2022 - IEEE International Conference on Communications*, Seoul, Korea, Republic of, 2022, pp. 1438-1443, doi: 10.1109/ICC45855.2022.9838863.

Edge-Powered In-Network Processing for Content-Based Message Management in Software-Defined Industrial Networks

Mattia Fogli*, Carlo Giannelli†, Cesare Stefanelli*,
* *Department of Engineering, University of Ferrara, Italy*
† *Department of Mathematics and Computer Science, University of Ferrara, Italy*
{mattia.fogli, carlo.giannelli, cesare.stefanelli}@unife.it

***Abstract*—Traditional industrial networks were characterized by flat topologies, where industrial equipment exchanged a limited number of messages. In sharp contrast, modern manufacturing plants are evolving towards articulated environments generating an ever-increasing amount of network traffic. Such emerging environments prevent the adoption of traditional solutions based on end-to-end dispatching of few messages in reliable networks with bandwidth availability much greater than needed. This leads to the need to adopt proper message management strategies as close as possible to industrial equipment to avoid overwhelming the industrial network with non-mission-critical traffic at the expense of mission-critical one. This paper originally proposes edge-powered in-network processing to i) transparently manage messages sent by industrial equipment, ii) support a broad spectrum of message management strategies, ranging from efficient header-based solutions to expressive content-based ones, and iii) fulfill the application-dependent requirements demanded by industrial environments nowadays. Achieved performance results based on a proof-of-concept prototype demonstrate that the proposed solution efficiently provides content-based message management at the edge, even considering edge nodes with limited hardware capabilities.**

***Index Terms*—Content-Based Message Management, Edge Computing, In-Network Processing, Industrial Internet of Things, Software-Defined Networking**

I. Introduction

Traditionally, industrial networks only dispatched a few messages in flat topologies to monitor primary operational- and safety-related machine parameters. However, the advent of Industrial Internet of Things (IIoT) has enabled novel applications, such as online remote reconfiguration and predictive maintenance [1], not only greatly increasing the traffic but also widening the set of nodes interested in receiving messages generated by machines. The adoption of proper solutions able to manage such demanding scenarios is particularly challenging since industrial equipment cannot be easily upgraded to fit novel requirements. In fact, modifying the software within machines could be either not feasible (e.g., machines do not allow firmware updates) or not legit (e.g., manufacturer warranties would be void in case of modifications).

This paper proposes an architecture enabling content-based message management at the edge of software-defined industrial networks. The objective is to promote content-based message management while not burdening time-critical applications, which might not tolerate the additional delay due to payload inspection and processing. Given that legacy networking devices can only inspect packet header details (e.g., source/destination address, protocol, and checksum), we deploy Edge Nodes (ENs) close to networking devices via high-speed links to support more expressive content-based message management strategies. By doing so, we enrich networking devices with edge-powered In-Network Processing (eINP) capabilities, thus enabling message management decisions based on what is carried in payloads rather than in headers. In addition, we adopt the Software-Defined Networking (SDN) paradigm to ensure fine-grained control over how messages traverse the network, also considering application-dependent requirements (e.g., time-sensitive messages are promptly forwarded by circumventing ENs). Whenever feasible, messages from industrial equipment to monitoring/management services are rerouted towards ENs, enforcing content-based message management in a transparent, cost-effective, and application-aware fashion. As better detailed later, our solution is based on an Industrial Application Manager (IAM), which orchestrates the interplay between an SDN controller (that dictates how messages traverse the network) and an eINP controller (that manages eINP capabilities on ENs).

The rest of this paper is structured as follows. Section II introduces the current literature related to edge computing, SDN, and In-Network Processing (INP) in industrial environments. Section III outlines the traditional industrial layering, while Section IV presents how industrial environments are evolving, their main issues, and the design guidelines we have identified to address such issues. Then, Section V and Section VI present the proposed solution, provide implementation details, and describe achieved performance results based on our prototype. Finally, Section VII provides conclusive remarks.

II. Related Work

Nowadays, edge computing is adopted in many industrial scenarios with the primary goal of bringing computation closer to data sources, i.e., from the cloud to on the premises [2], [3]. In this manner, it is possible to improve data protection (by computing sensitive data on on-premises nodes at the edge rather than sending them to the cloud) and real-time responsiveness (since latency at the edge is much lower than at the cloud). In this regard, [4] proposes an architecture to

support time-sensitive tasks and real-time data analysis at the edge, while big data analytics tools reside in the cloud. [5] designs a multi-level computing architecture compliant with the multi-access edge computing initiative enabling unified management of fog/edge-computing resources in Industry 4.0.

Edge computing is frequently adopted together with the SDN paradigm [6], [7]. For instance, [8] adopts SDN to handle the edge-cloud interplay in IIoT environments. The authors take advantage of SDN to handle big data streams in an energy-aware and Quality of Service (QoS)-guaranteed fashion. Similarly, [9] combines edge computing with SDN to support data streams with different latency constraints among IIoT devices.

Recently, INP, also known as in-network computing, has been proposed to support the execution on networking devices of software modules that typically run on hosts [10], [11]. However, its adoption in industrial environments has not been yet widely investigated. A notable first example is [12], which proposes an SDN-based adaptive transmission protocol to support time-critical services by on-demand activating in-network functions for in-path caching and retransmission. In addition, [13] combines edge computing with INP to offload critical lightweight (sub)tasks to networking devices while keeping the others on edge-computing resources.

Some INP-related proposals already provide forms of payload processing directly within networking devices but with a limited degree of expressiveness. For instance, [14] proposes PayloadPark (based on P4 [15]), which parks payloads in programmable switch memory, sends header-only packets to network-function servers (that primarily make decisions based on headers), and finally resembles them as such packets come back. [16] uses P4-enabled switches to aggregate packets sharing common headers into one and then to disaggregate the aggregated packet before reaching the destination. [17] programs P4-enabled switches to generate an alarm if MQTT messages convey values exceeding a given threshold.

In contrast to the INP-related proposals outlined above, we broaden the scope of INP by adopting it on ENs [18]. Note that the traditional definition of INP only includes devices already used for networking. The rationale behind the so-called eINP is that networking devices cannot (still) provide a high degree of expressiveness at a line rate. Accordingly, we enable content-based message management at different degrees of expressiveness in a transparent, cost-effective, and application-aware manner by connecting networking devices to ENs via high-speed links.

III. Industrial Environments: A Layered View

To better present the proposed solution and the benefits it brings, let us point out that the typical industrial architecture adopted nowadays by manufacturing companies is composed of three primary levels: shop floor, plant, and enterprise.

The **shop-floor level** resides at the bottom layer and mainly focuses on industrial automation. Traditionally, a shop floor consists of manufacturing machinery equipped with sensors (e.g., temperature and pressure) and actuators (e.g., drills and presses). Programmable Logic Controllers (PLCs) are tailor-made devices to cope with harsh conditions typically affecting industrial environments (e.g., extremely high/low temperatures, massive vibrations, and tremendous electrical noise) and control manufacturing processes. Human operators interact with and receive feedback from manufacturing machinery through Human-Machine Interfaces (HMIs), specifically designed for improving operator decision-making.

The **plant level** resides at the middle layer and focuses on the management of manufacturing processes. The Manufacturing Execution System (MES) allows information flowing upstream and downstream between the shop-floor level (where manufacturing machinery and human operators produce goods) and the enterprise one (where managers make decisions), assuming integrated systems along the way. More specifically, the MES receives instructions describing what machinery should accomplish from MES operators. Then, the MES transmits such instructions downwards, i.e., to HMIs and PLCs (shop-floor level). As the requested manufacturing process goes into production, the MES checks that the instructions provided by the MES operators match with the actual actions taken by manufacturing machinery. If needed, the MES applies corrective actions accordingly.

The **enterprise level** resides at the top layer, where decision-makers rely on the Enterprise Resource Planning (ERP) for running business operations. The ERP provides an up-to-date enterprise-wide view based on the data coming from the underlying business activities and resources, such as supplies, cash, customer orders, and production processes. The ERP mainly relies on a database management system to store data and provides business analytics to interpret them. As a result, the ERP plays a crucial role in enabling information circulation both upwards and downwards (shop floor - plant - enterprise) and also among enterprise departments ancillary to manufacturing, such as accounting and sales.

IV. Content-based message management: challenges and solution guidelines

The widespread deployment of IIoT (enriching industrial environments with networked devices equipped with sensing and computing capabilities) is quickly changing modern industrial environments. For instance, manufacturing companies benefit from IIoT by tracking assets, monitoring equipment, and optimizing processes, overall pushing towards cost-saving. However, IIoT devices make shop floors more articulated than ever, raising important concerns also in terms of QoS and cyber security. The high amount of traffic generated by IIoT devices (if compared with the traffic of traditional industrial equipment) pushes for the deployment of novel software components (along with MES and ERP) able to monitor and manage traffic flows to maximize the QoS as well as to ensure the safety of human operators. In addition, while legacy shop floors typically consist of vendor-supplied hardware and software, IIoT devices present complex chains of software dependencies (e.g., third-party libraries), making integrity mechanisms challenging to be guaranteed

[19]. Let us also note that the typical lifetime of industrial equipment is between 10 and 15 years (also longer in some cases), and its software cannot be easily updated, either since there is no access to its firmware or it is forbidden by the manufacturer. For this reason, the industrial environment is typically characterized by highly heterogeneous devices, ranging from industrial devices with minimal computing and memory capabilities running tailor-made network applications to up-to-date commodity server technology.

Based on these considerations, we claim that typical end-to-end communications adopted in traditional manufacturing environments must evolve towards message management strategies able to adapt to ever-changing conditions of the network and to requirements of industrial applications. So far, industrial machines are configured to receive control messages from the control room (by the MES above all) and send back a few information about their current state (e.g., number of crafted and faulty products). Sporadically, industrial machines also exchange messages with each other, e.g., machines in the same production line share information about the rate of crafted products. In any case, once industrial machines and companion control servers are deployed, their dynamic reconfiguration is not possible, e.g., requiring to stop production to reroute messages towards a different control server. On the contrary, modern industrial environments need to dynamically manage message flows, even at service provisioning time and without imposing any pause to the production process. For instance, since servers can be easily replicated and migrated while industrial machines relocated, it is not possible anymore to support only static topologies.

Our purpose is to make both more flexible and network-efficient the interconnection between shop-floor equipment and control-room services for industrial monitoring/management. In this regard, we recognize that in fulfillment of modern industrial guidelines for cyber security (e.g., IEC 62443 [20]), the overall industrial topology should be split into several shop-floor subnets and a control-room subnet. As Fig. 1 shows, we consider shop-floor subnets involving an EN, a Gateway (GW), and the typical shop-floor equipment (e.g., industrial machines, PLCs, HMIs, and IIoT devices), whereas the control-room subnet includes a GW and plant/enterprise-level services (e.g., MES/ERP and IIoT/Security Manager).

To provide a more flexible message management solution fitting modern industrial environments, we claim that industrial technicians should be able (via the IAM) to decide if and how messages should be inspected or even manipulated close to industrial equipment, thus moving part of the traffic management know-how at the edge of shop-floor subnets. In this manner, content-based message management can be done as soon as possible, without imposing centralized management on servers in the control room (as it is in traditional solutions). Note that a centralized approach not only might represent a single-point-of-failure and a potential bottleneck, but it also imposes additional transmission delays since messages must be sent back and forth to control-room servers to be managed. In addition, by leveraging on content-based message

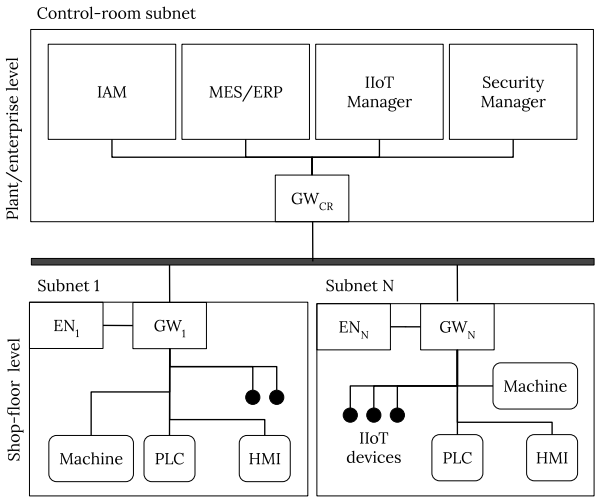

Fig. 1: Target network topology for industrial environments.

management at the edge, it is possible to easily deploy novel management strategies without requiring to modify the software on industrial machines, thus increasing flexibility and dynamicity even in the (not so unusual) case of old machines that cannot be modified or replaced. Accordingly, we identify the following primary guidelines that should be followed to support content-based message management in modern industrial environments:

- **Transparent.** Senders and receivers can be even unaware (e.g., by ignoring the existence of ENs) that content-based message management is in place, thus increasing the flexibility of the industrial environment;
- **Cost-effective.** Content-based message management must not impose an overhead cost beyond what senders and receivers can tolerate;
- **Application-aware.** Content-based message management must be aware that multiple industrial applications, even potentially demanding conflicting requirements, might concurrently run in the same plant.

V. Architecture Overview and Prototype Insights

To fulfill the solution guidelines provided above, we have designed, implemented, and experimentally verified an architecture combining two networking paradigms: SDN and eINP.

We believe that SDN-eINP is the enabling interplay to provide content-based message management in a transparent, cost-effective, and application-aware fashion (see Section IV). As Fig. 2 shows, we consider tightly coupled GWs and ENs, where the former act as mere forwarding elements while the latter provide eINP capabilities to process payloads in situ. From a networking perspective, the main difference between GWs and ENs is their knowledge of (and the range of actions they can apply to) messages flowing through them. Specifically, GWs inspect headers while ENs inspect payloads. Therefore, messages opaquely traverse GWs, which make header-based forwarding decisions according to the rules

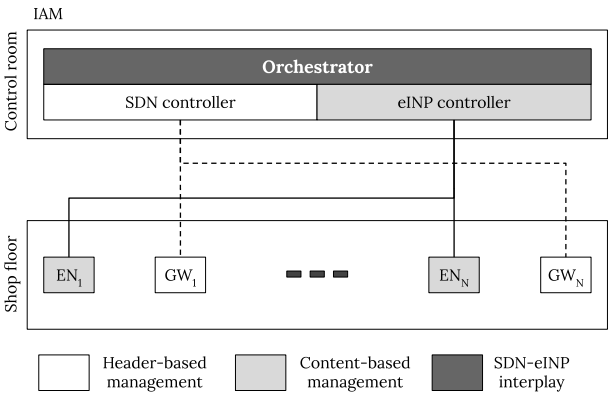

Fig. 2: IAM architecture overview.

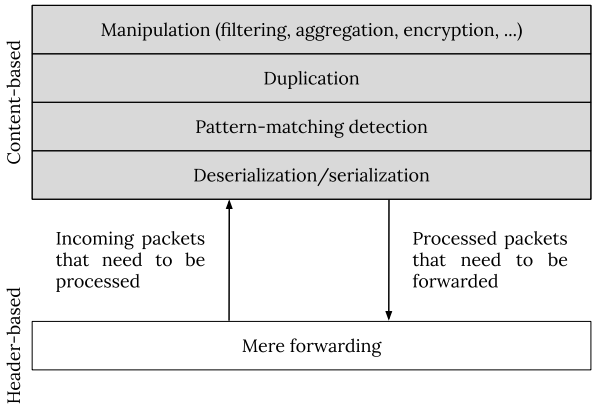

Fig. 3: Header- vs. content-based message management.

pushed by the SDN controller, whereas ENs make content-based management decisions, which depend on the information included in payloads.

Our architecture design aims to integrate the logically centralized nature of SDN with the decentralized one of eINP for content-based message management. To do so, we take advantage of the IAM, which includes an SDN controller, an eINP controller, and an orchestrator. As in traditional SDN architectures, the SDN controller maintains an up-to-date network-wide view and pushes forwarding rules to GWs. Similarly, the eINP controller gathers updates from ENs about resource usage (e.g., CPU and memory) and de/activates eINP capabilities on ENs as needed. Lastly, the orchestrator oversees the SDN-eINP interplay. Specifically, such an orchestrator uses the SDN controller to redirect target messages passing through GWs towards nearby ENs (see Fig. 1). In turn, ENs provide content-based message management and send back processed messages to the GWs. In addition, the orchestrator balances the incoming message rates on ENs to avoid them running out of resources (otherwise, ENs might become bottlenecks).

As Fig. 3 shows, an eINP-enabled EN first needs to deserialize payloads to make content-based decisions. On top of that, the EN can perform a plethora of actions, such as pattern-matching detection, duplication, and manipulation. For instance, the EN can detect patterns in messages flowing through it and deliver such messages to actors interested in getting them. A typical use case regards messages that potentially represent cyber security threats. Accordingly, the EN can promptly duplicate such messages and deliver them to actors that are not the actual destination but need to receive them, such as the Security Manager in the control room. Note that security modeling and analysis (whose definition is beyond the scope of this work) to determine whether the Security Manager should receive a given message can be dynamically deployed as modules on ENs.

It is important to note that although message duplication improves situation awareness, it intrinsically increases network traffic. In this regard, aggregation and filtering could be valuable strategies to mitigate it. An example of aggregation is averaging given field values carried in a certain number of consecutive messages that belong to the same flow and sending out a single message containing the computed value. To do so, the EN needs to queue incoming messages on a flow basis and perform the aggregation function. This dramatically decreases outgoing traffic but with potential costs in terms of processing overhead and increased delay. Instead, filtering functions drop messages if the target field is evaluated as of little application interest, e.g., a vibration within a safe range. In addition to aggregation and filtering (but without impacting traffic volume), encrypting is another example of message manipulation, which allows protecting communications of machines that do not provide up-to-date (or even at all) security mechanisms.

As a proof of concept, we developed a software prototype to implement an eINP module for ENs. This prototype is an initial step towards transparent, cost-effective, and application-aware message management at the edge in industrial environments. First, it deserializes JSON messages. Such messages follow a pre-defined schema that dictates a set of recommended tags for providing information about the data carried within messages, e.g., whether a field is aggregable or not and, if so, the aggregation function to use. Then, our eINP module can duplicate and even encrypt messages using Fernet [21], an implementation of symmetric authenticated cryptography. Lastly, we implemented an aggregation function that sorts incoming messages in queues according to the flow they belong to. As soon as a queue grows up to, e.g., ten elements, the developed module computes the average value and sends out a single message containing the computed value.

VI. Experimental Evaluations

To assess the feasibility and efficiency of the proposed solution, we tested how the prototype outlined in Section V performs while providing content-based message management with different degrees of complexity and expressiveness. In the following, we describe testbed setup and testing framework in Section VI-A, and we detail experiments and analyze performance results in Section VI-B.

A. Testbed Setup and Testing Framework

We built a testbed consisting of 5 Virtual Machines (VMs) running on Amazon Web Services Elastic Compute Cloud, each based on Ubuntu Server 20.04 LTS and equipped with 1 vCPU and 1 GB of RAM. The first VM acted as an industrial machine, sending messages based on pre-defined templates at arbitrary messages per second (mps) rates. The second VM represented the laptop of a maintenance technician interested in receiving warning messages under specific circumstances, such as an IIoT device detecting a hazardous noise level. The third VM acted as an eINP-enabled EN (see Section V). The fourth VM collected messages from machinery as an MES/ERP would do in real-world scenarios. The fifth VM was a dedicated server used by the IIoT/Security Manager (see Fig. 1) to further inspect/analyze network traffic. The industrial machine, technician laptop, and eINP-enabled EN resided on the shop floor, while the MES/ERP and IIoT/Security Manager were in the control room.

We also developed a testing framework to run experiments in a consistent and reproducible manner. In particular, we used Ansible, a configuration management tool, to configure the testbed over SSH from a remote control node. Also, we used `cpulimit` [22] and `psrecord` [23] to limit and monitor the CPU usage of a target process (i.e., the eINP module running on the EN), respectively. Then, we captured incoming and outgoing network traffic flowing through the eINP-enabled EN with `tcpdump`. Lastly, we used `capinfos` to extract statistics about capture files.

B. Experiments and Performance Results

We varied the experiments along three dimensions. The first one was the mps rate at which the industrial machine generated data. Specifically, we set increasing mps rates from 2K to 16K in steps of 2K, where each message was a UDP packet with a fixed payload size of 212 bytes. The second one was the percentage of CPU exploitable by the eINP module running on the EN. In fact, since the EN might execute other processing tasks, we explored how the eINP module performs in constrained conditions by limiting CPU availability to 50% and 25% of the total. The third one was the expressiveness degree adopted on the EN for content-based message management. In particular, we tested the eINP module while performing:

- *Mere forwarding*: The module forwarded incoming messages to the MES/ERP, acting as a forward proxy;
- *JSON de/serialization*: The module deserialized and then serialized incoming messages without taking any further action but forwarding;
- *Encryption*: The module de/serialized incoming messages, encrypted their contents with a symmetric key, and forwarded them to the MES/ERP;
- *Duplication & Aggregation (D&A)*: The module detected a blacklisted pattern with 5% probability and mission-critical information with 1% probability. In the former case, messages were duplicated and sent to the IIoT/Security Manager. In the latter case, messages were duplicated and sent to the technician laptop. Also, the eINP module performed aggregation (see Section V) by reducing the traffic volume towards the MES/ERP by 90%.

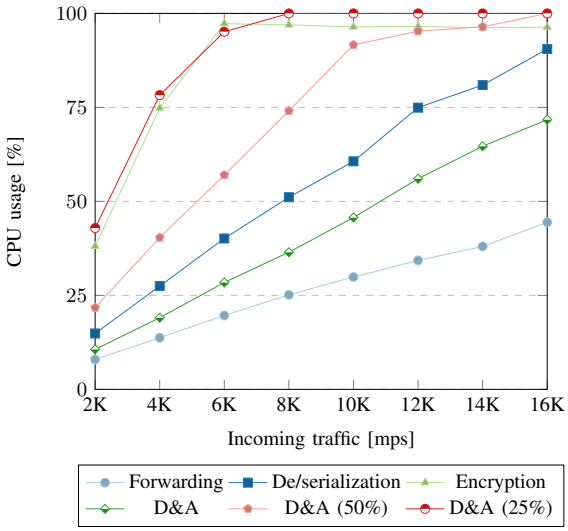

Fig. 4: CPU overhead.

We collected the results by running each experiment for 100 seconds. As Fig. 4 shows, the eINP module successfully handled incoming traffic up to 16K mps while performing mere forwarding, JSON de/serialization, and D&A with a CPU usage up to 44.5%, 90.5%, and 71.7%, respectively, thus demonstrating the feasibility and efficiency of the proposed solution. However, the module reached CPU saturation at just 6K mps while performing encryption, a very CPU-intensive task. It is interesting to note that D&A is much more efficient than encryption. In fact, D&A was still affordable up to 10K mps even when the CPU availability was limited to 50%. In particular, the CPU usage while performing encryption mirrored D&A when the CPU availability was limited to 25%, meaning the former is roughly four times more expensive than the latter. While this may seem counter-intuitive (since D&A actually includes JSON de/serialization), it is important to note that aggregation allows reducing outgoing messages, dramatically decreasing the CPU usage due to forwarding. Therefore, aggregation not only helps in reducing the imposed traffic but also the CPU consumption by balancing the cost of JSON de/serialization.

Along with processing costs, the eINP module also introduces delays. In particular, Fig. 5 shows the per-message processing time, measuring the elapsed time between when the eINP module receives a message and when it checks for the next one. Such delays remained roughly constant with any incoming traffic rate for mere forwarding (15.75 μs $\pm$ 1.74 μs), JSON de/serialization (49.59 μs $\pm$ 2.83 μs), and D&A (31.19 μs $\pm$ 1.12 μs). Note that the per-message processing time due to JSON de/serialization was 32% higher than that due to D&A. As mentioned above, this is because aggregation

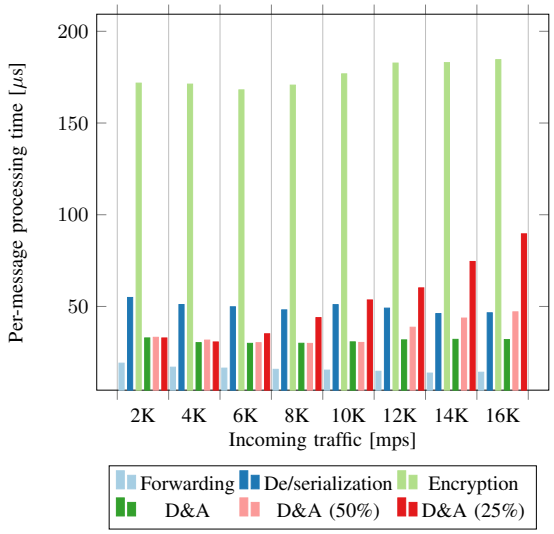

Fig. 5: Per-message processing time overhead.

allows forwarding less, which is a costly operation per se. While CPU availability was limited to 50% and 25%, the delay due to D&A tended to be longer as incoming traffic rates increased. By delving into details, per-message processing time started to increase when CPU reached saturation, i.e., at incoming traffic rates of 10K mps (CPU limited to 50%) and 6K mps (CPU limited to 25%), leading to longer delays. Specifically, the per-message processing time was 30.39 μs at 10K mps (CPU limited to 50%) and 35.11 μs at 6K mps (CPU limited to 25%). Instead, such time was 47.07 μs (CPU limited to 50%) and 89.60 μs (CPU limited to 50%) at 16K mps. Lastly, it is worth mentioning that encryption (176.09 μs $\pm$ 6.53 μs) took ten, three, and five times longer than mere forwarding, JSON de/serialization, and D&A, respectively.

VII. Conclusions

This paper proposes to adopt eINP coupled with SDN to support the transparent, cost-effective, and application-aware management of messages at the edge of industrial environments. Achieved performance results demonstrate that the proposed eINP fits the demanding requirements of industrial environments. In particular, the imposed processing costs and delays introduced by the eINP module are limited, thus allowing expressive and articulated message management techniques even on edge nodes with poor hardware capabilities. Such encouraging results foster the research activity towards developing additional message management techniques, such as homomorphic encryption to manipulate encrypted payloads without decrypting them. Also, we plan to enrich the target scenario by considering a network backbone providing multi-hop multi-path connectivity among subnets and P4-enabled programmable switches to which offload simple (sub)tasks of the content-based message management. Lastly, we intend to experimentally assess the SDN-eINP interplay, which primarily includes the joint orchestration of content-based message management (eINP side) and traffic flow steering (SDN side).

References

[1] A. Corradi, L. Foschini, C. Giannelli, R. Lazzarini, C. Stefanelli, M. Tortonesi, and G. Virgilli, "Smart appliances and rami 4.0: Management and servitization of ice cream machines," *IEEE Transactions on Industrial Informatics*, vol. 15, no. 2, pp. 1007–1016, 2019.

[2] W. Shi and S. Dustdar, "The promise of edge computing," *Computer*, vol. 49, no. 5, pp. 78–81, 2016.

[3] P. Bellavista, C. Giannelli, D. D. P. Montenero, F. Poltronieri, C. Stefanelli, and M. Tortonesi, "Holistic processing and networking (hornet): An integrated solution for iot-based fog computing services," *IEEE Access*, vol. 8, pp. 66 707–66 721, 2020.

[4] B. Chen, J. Wan, A. Celesti, D. Li, H. Abbas, and Q. Zhang, "Edge computing in iot-based manufacturing," *IEEE Communications Magazine*, vol. 56, no. 9, pp. 103–109, 2018.

[5] D. Borsatti, G. Davoli, W. Cerroni, and C. Raffaelli, "Enabling industrial iot as a service with multi-access edge computing," *IEEE Communications Magazine*, vol. 59, no. 8, pp. 21–27, 2021.

[6] D. Kreutz, F. M. V. Ramos, P. E. Veríssimo, 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, 2015.

[7] E. Haleplidis, K. Pentikousis, S. Denazis, J. H. Salim, D. Meyer, and O. Koufopavlou, "Software-defined networking (sdn): Layers and architecture terminology," *RFC 7426*, 2015.

[8] K. Kaur, S. Garg, G. S. Aujla, N. Kumar, J. J. P. C. Rodrigues, and M. Guizani, "Edge computing in the industrial internet of things environment: Software-defined-networks-based edge-cloud interplay," *IEEE Communications Magazine*, vol. 56, no. 2, pp. 44–51, 2018.

[9] X. Li, D. Li, J. Wan, C. Liu, and M. Imran, "Adaptive transmission optimization in sdn-based industrial internet of things with edge computing," *IEEE Internet of Things J.*, vol. 5, no. 3, pp. 1351–1360, 2018.

[10] "In-network computing," https://www.sigarch.org/in-network-computing-draft/, [Online; accessed 22-Oct-2021].

[11] D. R. K. Ports and J. Nelson, "When should the network be the computer?" ser. HotOS '19. New York, NY, USA: Association for Computing Machinery, 2019, p. 209–215.

[12] J. Chen, Q. Ye, W. Quan, S. Yan, P. T. Do, W. Zhuang, X. S. Shen, X. Li, and J. Rao, "Sdatp: An sdn-based adaptive transmission protocol for time-critical services," *IEEE Network*, vol. 34, no. 3, pp. 154–162, 2020.

[13] T. Mai, H. Yao, S. Guo, and Y. Liu, "In-network computing powered mobile edge: Toward high performance industrial iot," *IEEE Network*, vol. 35, no. 1, pp. 289–295, 2021.

[14] S. Goswami, N. Kodirov, C. Mustard, I. Beschastnikh, and M. Seltzer, "Parking packet payload with p4," in *Proceedings of the 16th International Conference on Emerging Networking EXperiments and Technologies*, ser. CoNEXT '20. New York, NY, USA: Association for Computing Machinery, 2020, p. 274–281.

[15] P. Bosshart, D. Daly, G. Gibb, M. Izzard, N. McKeown, J. Rexford, C. Schlesinger, D. Talayco, A. Vahdat, G. Varghese, and D. Walker, "P4: Programming protocol-independent packet processors," *SIGCOMM Comput. Commun. Rev.*, vol. 44, no. 3, p. 87–95, Jul. 2014.

[16] S. . Wang, J. . Li, and Y. . Lin, "Aggregating and disaggregating packets with various sizes of payload in p4 switches at 100 gbps line rate," *Journal of Network and Computer Applications*, vol. 165, 2020.

[17] A. Atutxa, D. Franco, J. Sasiain, J. Astorga, and E. Jacob, "Achieving low latency communications in smart industrial networks with programmable data planes," *Sensors*, vol. 21, no. 15, 2021.

[18] P. Bellavista, M. Fogli, L. Foschini, C. Giannelli, L. Patera, and C. Stefanelli, "Qos-enabled semantic routing for industry 4.0 based on sdn and mom integration," in *2021 IEEE 22nd International Conference on High Performance Switching and Routing (HPSR)*, 2021, pp. 1–6.

[19] F. Maggi and M. Pogliani, "Attacks on smart manufacturing systems," Trend Micro Research: Shibuya, Japan, Tech. Rep., 2017.

[20] "Iec 62443: Industrial network and system security," International Electrotechnical Commission, Tech. Rep.

[21] "Fernet (symmetric encryption)," https://cryptography.io/en/latest/fernet/, [Online; accessed 07-Feb-2022].

[22] "Cpu usage limiter for linux," https://github.com/opsengine/cpulimit, [Online; accessed 22-Oct-2021].

[23] "Record the cpu and memory activity of a process," https://github.com/astrofrog/psrecord, [Online; accessed 22-Oct-2021].