Joint Orchestration of Content-Based Message Management and Traffic Flow Steering in Industrial Backbones

M. Fogli, C. Giannelli and C. Stefanelli

2022 IEEE 23rd International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM), 2022

Cite this as

M. Fogli, C. Giannelli and C. Stefanelli, "Joint Orchestration of Content-Based Message Management and Traffic Flow Steering in Industrial Backbones," *2022 IEEE 23rd International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM)*, Belfast, United Kingdom, 2022, pp. 325-330, doi: 10.1109/WoWMoM54355.2022.00067.

Joint Orchestration of Content-Based Message Management and Traffic Flow Steering in Industrial Backbones

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*—The industrial internet of things has radically modified industrial environments, not only enabling novel services but also dramatically increasing the amount of generated traffic. Nowadays, a major concern within industrial plants is to support network-intensive services, such as real-time remote vibration monitoring of autonomous guided vehicles, while ensuring the prompt and reliable delivery of mission-critical safety-related messages among machines and the control room. To this purpose, we present a novel solution jointly orchestrating content-based message management and traffic flow steering: the former enables edge-powered in-network processing modules to process packet payloads as they traverse the industrial backbone, the latter supports dynamic (re)routing of traffic flows towards such processing modules. In particular, we exploit software-defined networking for flexible traffic flow (re)routing and Kubernetes for dynamic deployment on edge nodes of in-network processing modules for content-based message management. As demonstrated by performance results based on our working proof-of-concept prototype, our solution efficiently allows to manage industrial traffic flows in a coordinated fashion, by considering requirements of concurrently running industrial applications and the current state of the overall topology.**

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

I. Introduction

In the past, industrial environments had almost no practical issues concerning the amount of network traffic produced or consumed at the shop floor level. Such traffic was fairly limited to mission-critical information generated by fixed equipment, primarily carrying operational- and safety-related machine parameters. The network resources available on-site were (more than) adequate for managing it timely and reliably.

The advent of the Industrial Internet of Things (IIoT) is radically changing how modern industrial environments look like. On the one hand, IIoT devices have enabled sophisticated applications (e.g., mobile asset tracking, online remote reconfiguration, and predictive maintenance) while being low-cost. On the other hand, they have made the shop floor more articulated than ever, not only challenging the network infrastructure with an unprecedented amount of traffic but also posing security threats. In fact, IIoT devices rely on complex chains of software dependencies (e.g., third-party libraries), which make integrity mechanisms difficult to be guaranteed from both an operational and a security perspective.

It is worth noting that the deluge of messages sent out by IIoT devices may clog up the network infrastructure. In addition, modern industrial equipment, e.g., Automated Guided Vehicles (AGVs) moving spare parts among the plant and the warehouse, may abruptly migrate traffic flows across different Network Devices (NDs), e.g., switches, routers, and access points, in relation to their physical position. As a result, it is paramount to manage non-mission-critical and mission-critical traffic dynamically. Otherwise, the non-mission-critical traffic would take the network resources at the expense of the mission-critical one. This leads to the following. First, traffic flows should be properly prioritized: higher-priority operational flows go first, then non-mission-critical monitoring ones (flow prioritization). Second, messages should be discarded when carrying useless or not so relevant information (data filtering). Third, information should be aggregated whenever possible (data aggregation). By doing so, mission-critical traffic would be forwarded timely while optimizing the traffic traversing the network.

Accordingly, we propose a novel solution based on the joint orchestration of content-based message management and traffic flow steering in industrial backbones. Content-based message management enables decisions based on what is carried within packet payloads rather than only on packet headers (mere forwarding). This gives us the ability to look into (and potentially manipulate) the information carried within packets as they traverse the network (who produces/consumes the information is unaware of what is going on). However, since payload inspection and processing may introduce additional delays, content-based message management must be enforced as much as possible but without burdening mission-critical traffic. To this purpose, we adopt traffic flow steering to dynamically (re)route messages towards Edge Nodes (ENs) in charge of processing them, if and where required.

The rest of the paper is structured as follows. Section II introduces the related work. Section III outlines the traditional industrial layering. Section IV discusses the drivers of the proposal. Section V outlines the main architectural components of the proposed solution. Section VI provides implementation details, describes the testbed setup, and lays out performance results achieved on top of the implemented proof-of-concept prototype. Finally, Section VII provides conclusive remarks.

II. Related Work

Industrial scenarios adopt edge computing more and more frequently with the primary goal of bringing computation close to data sources, i.e., from the cloud to on the premises [1], [2]. This may improve, among others, data protection (by processing sensitive data on-premises rather than in the cloud) and real-time responsiveness (since latency at the edge is much lower than at the cloud). In this regard, [3] 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. [4] designs a multi-level computing architecture compliant with the MEC initiative enabling unified management of fog/edge-computing resources in Industry 4.0.

The adoption of edge computing and Software-Defined Networking (SDN) [5], [6] is gaining momentum. For instance, [7] adopts SDN to handle the edge-cloud interplay in IIoT environments. Specifically, the authors rely on SDN to handle big data streams in an energy-aware and QoS-guaranteed fashion. Similarly, [8] combines edge computing with SDN to support data streams with different latency constraints among IIoT devices.

Recently, In-Network Processing (INP), also known as in-network computing, has been proposed to support the execution on NDs of software modules that typically run on end-hosts [9], [10]. However, its adoption in industrial environments has not been yet widely investigated. A notable first proposal is [11], 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. An example of making work together edge computing and INP is [12], which offloads critical lightweight (sub)tasks to NDs while keeping the others on edge-computing resources.

Some INP-related proposals already provide forms of payload processing directly within NDs but with a limited degree of expressiveness. For instance, [13] proposes PayloadPark (based on P4 [14]), which parks payloads in programmable switch memory, sends header-only packets to network-function servers (that primarily make decisions based on headers), and resembles them as such packets come back. [15] uses P4-enabled switches to aggregate packets sharing common headers into one and then to disaggregate the aggregated packet before reaching the destination. [16] programs P4-enabled switches to generate an alarm if MQTT messages convey values exceeding a given threshold.

Inspired by the proposals outlined above, we enabled traffic flow steering and content-based message management in industrial environments via the SDN-edge-powered In-Network Processing (eINP) interplay. Such an interplay makes the SDN side (how traffic flows traverse the backbone) and the eINP one (how those flows are to be processed as they traverse the backbone) work synergically. Note that the eINP concept [17], [18] broadens the scope of traditional INP. Specifically, we envision backbone NDs with companion ENs (connected via high-speed links) to support the so-called eINP.

III. Modern Industrial Environments

Nowadays, manufacturing companies typically implement architectures based on three levels: shop floor, plant, and enterprise.

The shop floor level is mainly focused on industrial automation. As depicted in Fig. 1, the primary components of the shop floor are industrial machines, Programmable Logic Controllers (PLCs), Human-Machine Interfaces (HMIs), IIoT devices, and AGVs. Industrial machines tend to have extremely long lifetimes (in the order of decades) and may implement different (proprietary) protocols. In addition, software upgrades may not always be possible, since manufacturers usually forbid software upgrades for safety reasons, or industrial machines may not support them at all. Instead, IIoT devices are characterized by a substantially shorter lifetime and usually communicate via well-known protocols. Then, PLCs are tailor-made devices for controlling manufacturing processes while coping with the harsh conditions affecting industrial environments, such as temperature surges, vibrations, and electrical noise. Human operators interact with and receive feedback from industrial machines through HMIs, specifically designed for improving operator decision-making. In addition, the shop floor has been recently enriched by the deployment of AGVs, automating the process of moving products among different parts of the industrial environment.

The plant level regards the management of manufacturing processes. The critical component is the Manufacturing Execution System (MES). In particular, the MES receives instructions about how industrial machines should behave from operators, and then it transmits such instructions downwards, i.e., towards the shop floor. As the requested manufacturing process goes into production, the MES matches the instructions provided by the operators against the actual actions taken by industrial machines. If the current actions do not match the desired actions, the MES applies corrective measures.

The enterprise level is about making decisions on how to run business operations. In this regard, decision-makers rely on the Enterprise Resource Planning (ERP). The ERP collects information by the underlying business assets, such as supply chains, cash flows, customer orders, and production processes. Then, it processes such information using business analytics to provide decision-makers with an enterprise-wide picture of what is going on.

Up-to-date industrial guidelines for cyber security (e.g., IEC 62443 [19]) recommend splitting the network topology into several shop floor subnets and a control room subnet. As Fig. 1 shows, shop floor subnets involve the shop floor components and a subnet Gateway (GW). The control room subnet comprises both plant and enterprise components, plus a subnet GW. Note that the subnet GW may also work as WiFi access point. The network backbone connects such subnets through multiple communication channels to increase performance and fault tolerance. The outcome is a multihop multipath topology providing several end-to-end connectivity opportunities with differentiated performance.

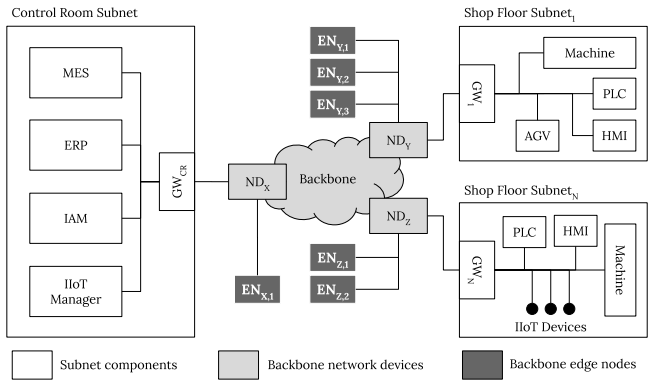


Fig. 1: Target environment.

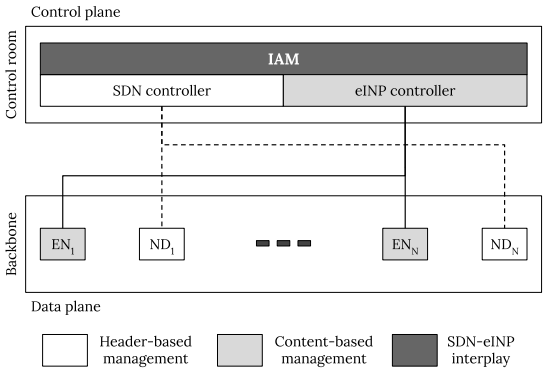


Fig. 2: Architecture overview.

IV. SDN-eINP Interplay for Traffic Management

The widespread deployment of IIoT devices has enriched industrial environments, delving into a quick shift towards modern industrial environments. For instance, the Industry 4.0 revolution has enabled monitoring applications generating a huge amount of data. In addition, the deployment of AGVs (and of relocatable industrial equipment in general) leads to dynamic and quick modifications of traffic flows, even at service provisioning time. Thus, there is the need to manage frequent and abrupt movements of industrial equipment, which generates multiple and unforeseen traffic flows if compared with former industrial environments usually crafting the same product(s) for long periods.

Modern industrial scenarios may involve multiple industrial applications concurrently running in the same environment and competing for shared resources. Such applications may generate traffic flows at differentiated priorities, e.g., highly relevant mission-critical operational start/stop messages (to be delivered promptly and reliably) and non-mission-critical messages for monitoring production processes (that may be delayed or even partially dropped). Therefore, it is paramount to support a priority-driven management of traffic flows, e.g., by rerouting high priority traffic flows towards high-performance paths while delaying low priority ones in case of bandwidth saturation. However, this might not be enough to manage industrial applications properly. In particular, in the case of huge traffic flows generated by multiple industrial applications, we claim there is the need to shape the content of transferred data, e.g., by processing packet payloads considering application-level characteristics and requirements. A typical example is the aggregation of multiple consecutive messages carrying temperature values in only one, the latter with the average temperature of aggregated messages.

In this regard, we propose to jointly exploit SDN and eINP. On the one hand, the adoption of the SDN approach allows to steer traffic flows based on the current state of the network from a logically centralized point of view. On the other hand, the deployment of eINP software modules on ENs tightly coupled with backbone NDs provides content-based message management at high degrees of expressiveness.

V. Architecture

This section lays out the architecture we designed to enable the SDN-eINP interplay described in Section IV. From a high-level perspective, the architecture consists of a control plane and a data plane (see Fig. 2). In the following, we will break down those planes into fundamental components.

A. Control Plane

The control plane is responsible for making decisions about i) how a flow traverses the backbone, ii) its priority along the path, and iii) how that flow is to be processed. The control room is where control plane architectural components are located. Such components consist of an Industrial Application Manager (IAM), an SDN controller, and an eINP controller.

The IAM is logically built on top of the SDN and eINP controllers. Its primary task is to bridge the SDN and eINP sides, steering traffic flows (SDN side) towards eINP modules for content-based message management (eINP side). To do so, the IAM takes advantage of the abstractions provided by each side, exploited as building blocks to enable the SDN-eINP interplay.

The SDN paradigm advocates for a clear-cut separation between who forwards packets and who makes routing decisions. The SDN controller is a logically (but not necessarily physically) centralized software entity that abstracts the underlying network infrastructure and provides the control logic to program the whole network. Specifically, it builds a network-wide view by gathering status statistics from NDs and dictates their behavior by pushing forwarding rules into them. Thus, NDs become dumb devices that make flow-based forwarding according to the instructions provided by the SDN controller. A flow is defined by rules matching a set of packet header values.

The eINP side abstracts computational resources made available by backbone ENs and orchestrates software modules (packaged as containers) across them. The eINP controller clusters those resources in zones. The scope of a zone is to seamlessly orchestrate containerized eINP modules only across those ENs directly connected to a given ND. For instance, zone "Y" (see Fig. 1) includes $EN_{Y,1}$, $EN_{Y,2}$, and

$EN_{Y,3}$, the only ENs directly connected to ND_Y. The rationale behind that is to bring eINP modules as close as possible to the NDs where the target traffic flow has been scheduled to pass through, avoiding messages traversing the backbone back and forth to be processed.

B. Data Plane

The data plane comprises backbone NDs and ENs. It is worth remarking that NDs and ENs pursue different purposes, although working synergically. On the one hand, NDs enforce steering policies on traffic flows. On the other hand, ENs apply processing functions to the data carried by messages belonging to such flows. From an operational point of view, both NDs and ENs provide a kind of processing. The difference lies in what they process. Specifically, the former process packet headers, whereas the latter packet payloads.

An EN is a cluster node that runs containerized eINP modules to perform content-based message management. As messages pass through an EN, the first step is payload de-serialization. On top of that, the EN can perform a broad spectrum of actions at different degrees of expressiveness, such as pattern-matching detection, message duplication, data aggregation, and encryption, among others. For instance, an EN may detect hazardous temperature levels and duplicate those messages to alert the affected operators.

Although message duplication may promptly improve information circulation, it also increases bandwidth usage. Strategies that may help in this regard are data filtering and aggregation. Filtering functions drop messages if the target field is evaluated as of little application interest, e.g., vibration values within a safe range. Data aggregation consists of, for example, averaging a target field carried in a certain number of consecutive messages that belong to the same flow and sending out a single message containing the average value. To do so, an EN queues incoming messages on a flow basis and performs the aggregation function as soon as the queue amounts to a given threshold. In addition to data aggregation, encryption is another form of payload manipulation. In this case, an EN may use a symmetric key to encrypt packet payloads of a target flow as they leave the sender, and then another EN may decrypt them before they reach the receiver. This allows protecting communications of machines that do not provide up-to-date (or even at all) security mechanisms.

C. Running Example

The starting point (see Fig. 3-up) describes a stationary situation, including two flows from different shop floor subnets. The AGV generates the so-called elephant flow (bold orange line) towards the MES, whereas an IIoT device produces a mice flow (dashed blue line) towards the IIoT manager. An eINP module providing data aggregation processes the elephant flow as close as possible to its source subnet. This allows saving backbone network resources since such a module dramatically lowers the size of the elephant flow. At the same time, even the mice flow goes through an eINP module for data filtering, dropping messages of little application interest.

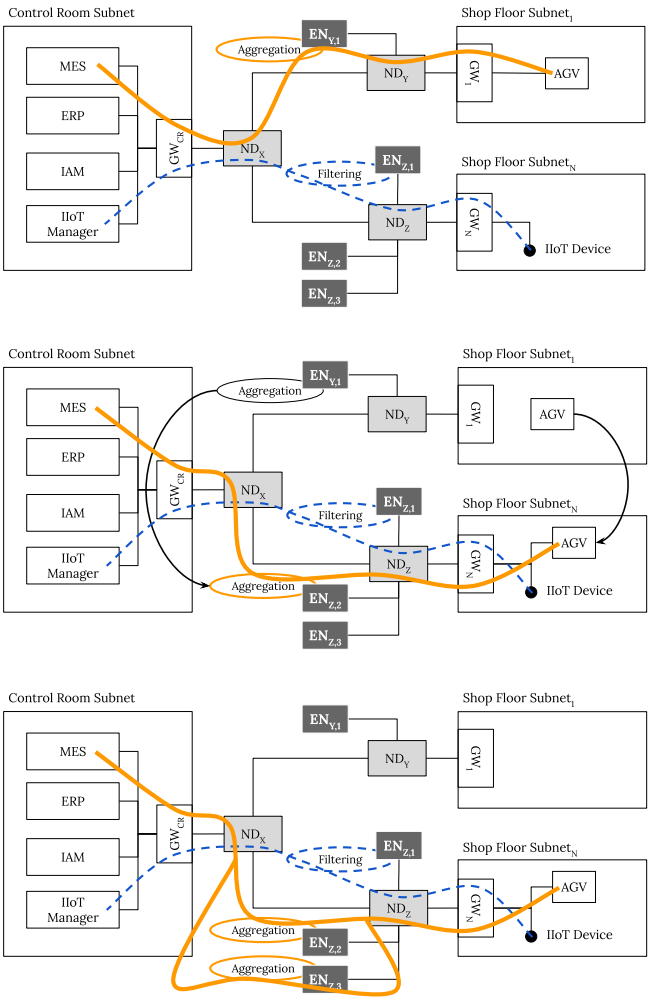
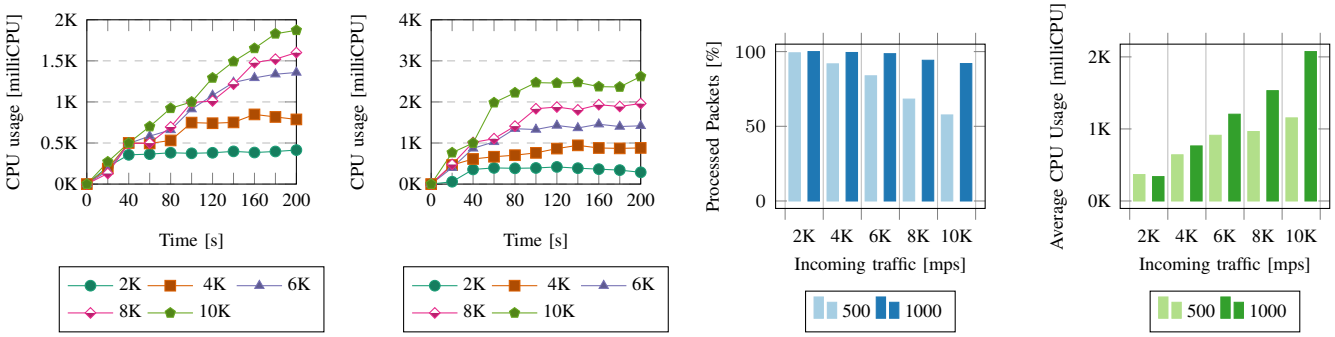


Fig. 3: Start-up (up), AGV movement (middle), and scaling out (down).

Then, the AGV starts moving from one shop floor subnet to another, changing the WiFi access point to which it is connected accordingly. As a result, the elephant flow also needs a new path towards the MES. However, an eINP module for data aggregation is not available along the new path. On the one hand, the elephant flow might go straight to the MES but with no aggregation in place. On the other hand, it might pass through the eINP module for data aggregation but via a longer path. In both cases, there is a waste of network resources. This triggers the IAM, in charge of checking for such harmful communication patterns. In this regard, the IAM decides to activate an eINP module for data aggregation along the shortest path possible (see Fig. 3-middle).

In this particular example, the cluster zone closest to the target subnet is resource-rich. This gives the capability to scale out and in module replicas dynamically. Since the elephant flow stresses the single replica available beyond a given threshold, the eINP controller starts a new module replica in the same cluster zone and thereby balances the incoming traffic accordingly (see Fig. 3-down).



(a) 500 milliCPU. (b) 1000 milliCPU. (c) 500 vs. 1000 milliCPU. (d) 500 vs. 1000 milliCPU.

Fig. 4: eINP while performing encryption with different milliCPU limit.

VI. Prototype Description and Evaluation

To assess the feasibility and efficiency of the proposed architecture, we developed and experimentally tested a proof-of-concept prototype that provides content-based message management at different degrees of expressiveness.

A. Testbed Setup

The testbed consisted of seven Virtual Machines (VMs) hosted on Amazon Web Services (AWS) Elastic Compute Cloud (EC2). Four of them made up a Kubernetes (v1.21.1) cluster involving a control plane and three workers. The workers acted as ENs while the control plane acted as the eINP controller. The ENs were split into two zones: "A" and "B." An additional VM mimicked an industrial machine, producing traffic flows at arbitrary rates. Another VM collected the messages generated by the industrial machine, acting as the MES. The last VM represented a technician laptop. In this scenario, the technician was interested in getting warning messages only when specific circumstances occur, such as hazardous vibrations detected by the industrial machine.

We assumed that i) zone "A" was the closest to the industrial machine, ii) the SDN controller steered the traffic flows generated by the industrial machine through zone "A", and iii) the eINP controller could not take advantage of resources available in zone "B" since not on the path. As the traffic grew, the control plane could decide to scale out horizontally (deploy more replicas of) the eINP module. Each cluster node ran Ubuntu 18.04 LTS and was equipped with two vCPUs and two GB of RAM. We chose CRI-O (v1.20.0) as the container runtime and Flannel (v0.14.0) as the Container Network Interface (CNI) plugin. The remaining VMs (industrial machine, MES, and technician's laptop) were based on Ubuntu 20.04 LTS and provided with one vCPU and one GB of RAM, respectively.

B. Experiments and Performance Results

The experiments varied along three dimensions. The first dimension was the rate the industrial machine sent out messages. In this regard, we set increasing rates from 2K to 10K Messages Per Second (MPS) in steps of 2K MPS. Each message was a UDP packet with a fixed payload size of 212 bytes.

The second dimension was the CPU usage to which the containerized eINP module was limited. Note that Kubernetes uses milliCPU as the base unit for expressing amounts of CPU and allows specifying both CPU requests and limits [20]. Specifically, a CPU request states the minimum amount of CPU a container needs to run, while a CPU limit indicates the maximum amount of CPU a container can take while running. For all the experiments, we fixed the CPU request to 500 milliCPU and tested the target eINP module with CPU limits of 500 milliCPU and 1000 milliCPU. When a container sets its CPU limit higher than its CPU request, it may benefit as follows. If a burst of messages causes a peak of computational demand, the container may successfully handle the incoming traffic by taking resources beyond its request. Otherwise, the container would run out of resources. Second, the horizontal autoscaler decides whether to scale replicas based on the ratio between the current CPU usage and the average of the given target values across running replicas, with a tolerance of 0.1 by default (please see [21] for more details). Given that a higher CPU limit allows a potentially higher CPU usage, the horizontal scaling is faster when peaks of computational demand occur. For all the experiments, the average target utilization was 80%, the minimum number of replicas was one, and the maximum number of replicas was four.

The third dimension was the kind of content-based message management. Specifically, we tested the following eINP modules:

- *Duplication & Aggregation (D&A)*: this module de/serialized incoming messages and detected mission-critical information with a probability of 5%. Such messages were duplicated and sent to the technician's laptop timely. Then, the module performed aggregation, reducing outgoing traffic by 90%;
- *Encryption*: this module de/serialized incoming messages, encrypted their contents with a symmetric algorithm, and forwarded them to the MES.

To statistically support our findings, each module has been tested by repeating the experiments five times, where each run of an experiment captured a time window of 200 s (in Fig. 4, each data point is the average value of five runs).

While performing D&A, the average target utilization we configured (80%, meaning 400 milliCPU) is not even triggered except for the worst-case scenario (10K MPS). It is worth mentioning that D&A is a very efficient form of content-based message management. In fact, aggregation dramatically decreases outgoing messages, cutting down JSON serialization and forwarding costs. The key takeway is that the eINP module correctly processed almost every packet at any incoming rate.

In contrast to D&A, encryption is a CPU-intensive task. This makes horizontal autoscaling critical, even at relatively low message rates. As Fig. 4a illustrates, the CPU usage grew to about 2000 milliCPU at 10K MPS and above 1500 milliCPU at 8K MPS. Given that Fig. 4a depicts the experiments while CPU limited to 500 milliCPU, it means Kubernetes scaled up to four replicas (i.e., the maximum allowed) to cope with such message rates. The comparison between Fig. 4a and 4b points out the system responsiveness with different CPU limits. In Fig. 4a, the lines about incoming rates of 6K, 8K, and 10K MPS still grow at the end of the time window (200 s), while all the lines reach a steady-state phase in Fig. 4b. This is because replicas available at a given time can take extra resources (over the CPU request) while horizontal scaling occurs. As Fig. 4c shows, replicas that can benefit from a higher CPU limit outperform those that cannot do it (500 milliCPU vs. 1000 milliCPU). For instance, the percentage of processed packets is 57.6% (500 milliCPU limit) against 92% (1000 milliCPU) at 10K MPS. Lastly, Fig. 4d reflects this point in terms of average CPU usage.

VII. Conclusions and Future Work

We implemented a working proof-of-concept prototype built on state-of-the-art technologies. Achieved performance results quantified its processing costs while performing content-based message management at different degrees of expressiveness. In particular, we explored horizontal scaling to cope with demanding forms of content-based message management, such as encryption.

Achieved performance results demonstrated that the adopted horizontal scaling solution with flexible resource cap is effectively able to properly enforce content-based message management in a dynamic fashion. Therefore, the joint orchestration of content-based message management and traffic flow steering may lay a solid foundation for properly shaping mission- and non-missional traffic flows concurrently traversing the industrial backbone.

Such promising results foster the research activity towards more sophisticated orchestration strategies using predictive algorithms to activate eINP modules proactively based on movement patterns of mobile industrial components. Also, we plan to explore additional content-based message management techniques, such as homomorphic encryption that allows manipulating encrypted payloads without decrypting them. Lastly, we will investigate how to make P4-enabled programmable switches and ENs work together.

References

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

[2] 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.

[3] 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.

[4] 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.

[5] 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.

[6] 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.

[7] 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.

[8] 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.

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

[10] 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.

[11] 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.

[12] 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.

[13] 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.

[14] 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.

[15] 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.

[16] 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.

[17] 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.

[18] M. Fogli, C. Giannelli, and C. Stefanelli, "Edge-powered in-network processing for content-based message management in software-defined industrial networks," in *2022 IEEE International Conference on Communications (ICC): Communication QoS, Reliability and Modeling Symposium*, vol. Submitted for Publication, 2022, pp. 1–6.

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

[20] "Resource management for pods and containers," https://kubernetes.io/docs/concepts/configuration/manage-resources-containers/, [Online; accessed 15-Jan-2022].

[21] "Horizontal pod autoscaling," https://kubernetes.io/docs/tasks/run-application/horizontal-pod-autoscale/, [Online; accessed 15-Jan-2022].