A Framework for QoS- Enabled Semantic Routing in Industrial Networks: Overall Architecture and Primary Protocols

A Framework for QoS-Enabled Semantic Routing in Industrial Networks: Overall Architecture and Primary Protocols

Paolo Bellavista*, Mattia Fogli†, Luca Foschini*,
Carlo Giannelli†, Lorenzo Patera*, Cesare Stefanelli†
*University of Bologna, Bologna, Italy, {paolo.bellavista, luca.foschini, lorenzo.patera}@unibo.it
†University of Ferrara, Ferrara, Italy, {mattia.fogli, carlo.giannelli, cesare.stefanelli}@unife.it

***Abstract*—The manufacturing sector represents a notable use case of the Industry 4.0 revolution, heavily stressing the capability of plants to ensure the desired QoS. Currently, manufacturing plants are characterized by an increasing amount of non-mission-critical traffic, in addition to traditional mission-critical safety-related traffic, which is negligible in comparison. Since computing and networking capabilities are no longer as abundant as in the past, there is the need to properly manage available resources. To ensure challenging QoS requirements, we propose a novel protocol suite specifically designed for our QoS-enabled semantic routing framework. Such a framework adopts an architecture that fits the characteristics of modern manufacturing environments and exploits an overlay networking solution providing a semantic routing substrate that operates both at the application and network layers.**

***Index Terms*—Software-Defined Networking, Message-Oriented Middleware, Semantic Routing, Industry 4.0, Industrial Internet of Things, Quality of Service**

I. Introduction

Following the Industry 4.0 revolution, the industrial domain has been going through a profound metamorphosis. Traditionally, the traffic produced on the shop floor was limited to mission-critical information generated by fixed equipment. Such traffic was mainly about operational- and safety-related machine parameters and was a limited amount in comparison with available network capabilities. As a result, network resources available in loco were sufficient to carry out mission-critical messages both timely and reliably. The advent of the Industrial Internet of Things (IIoT) has radically changed the industrial domain. IIoT devices have been the key enablers for modern industrial applications (e.g., mobile asset tracking, online remote reconfiguration, and predictive maintenance), but at the price of ever-increasing complexity. Such complexity concerns the amount of traffic traversing the network, infrastructure heterogeneity, and cyber-security threats, among others.

Within the Industry 4.0 revolution, the manufacturing sector represents a notable use case. Typically, manufacturing environments consist of a shop floor, a plant, and an enterprise level. Specifically, the shop floor level is where industrial automation takes place. The primary components of the shop floor are programmable logic controllers, human-machine interfaces, industrial machines, automated guided vehicles, and IIoT devices. Then, the plant level is about managing manufacturing processes, whereas the enterprise level is about decision-making to run business operations. The critical components of the plant and enterprise levels are the manufacturing execution system and the enterprise resource planning, respectively. It is worth mentioning that current industrial guidelines for cyber security recommend splitting the network topology into several shop floor subnets (shop floor level components) and a single control room subnet (plant/enterprise level components), each subnet provided with a dedicated gateway. The industrial backbone connects such subnets through multiple communication channels to increase performance and fault tolerance. This means a multihop multipath topology providing several end-to-end connectivity opportunities with differentiated performance.

Nowadays, manufacturing plants not only produce a deluge of non-mission-critical traffic but also include mobile equipment. This leads to a challenging domain, where computing and networking resources within and among control and shop floor subnets are no longer as abundant as in the past and potentially adopt different (even proprietary) protocols. Therefore, the dynamic management of resources at service provisioning is crucial to ensure the desired QoS level. For instance, automated guided vehicles moving spare parts among the plant and the warehouse may abruptly migrate traffic flows across different network devices (e.g., switches, routers, and access points) based on their physical position. Additionally, the plethora of industrial protocols on the shop floor makes both intra- end inter-subnet machine-to-machine communications difficult. Ideally, industrial machines should keep interacting one each other seamlessly, notwithstanding their actual physical location or heterogeneous data exchange formats, while also ensuring QoS requirements.

This paper proposes a protocol suite for a framework for QoS-enabled semantic routing in industrial networks. Such a framework is based on an architecture that fits the characteristics of modern manufacturing environments, as outlined above. In particular, the proposed solution relies on overlay networking to provide a semantic routing substrate that operates both at the application and network layers. The application layer consists of a Message-Oriented Middleware (MOM) and

several Application Gateways (AGWs), whereas the network layer combines Software-Defined Networking (SDN) and In-Network Processing (INP). The overarching ambition is to enable i) loosely-coupled, asynchronous communications, ii) fine-grained traffic management, and iii) in-network traffic optimization.

The rest of this paper is structured as follows. First, Section II discusses the related work. Then, Section III lays out the requirements of modern manufacturing domains. Next, Section IV depicts the architecture and the protocols we propose. Lastly, Section V provides conclusive remarks.

II. Related Work

Industry 4.0 is pushing toward sophisticated industrial environments, where heterogeneous industrial machines interact in an articulated and flexible manner [1]. The SDN paradigm [2], [3] is gaining momentum, not only in static and wired scenarios but also in wireless ad hoc ones, where mobile nodes generate short-living connections [4]. Currently, OpenFlow [5] is the most significant implementation of an open interface between the control plane (i.e., a logically centralized entity that implements the control logic) and the data plane (i.e., network devices that behave as dictated by the control plane).

In the industrial domain, SDN has often been jointly applied together with edge computing and IIoT. For example, [6] adopts SDN to manage the edge-cloud interplay in IIoT environments. Specifically, the authors used SDN to handle big data streams in an energy-aware and QoS-guaranteed fashion. Similarly, [7] combines edge computing with SDN to support data streams characterized by different latency constraints among IIoT devices.

Recently, INP has been proposed to support the execution on network devices of software modules that typically run on end hosts [8], [9]. A notable first example of INP applied to the industrial domain is [10], which proposes an SDN-based adaptive transmission protocol to support time-critical services by activating in-network functions on-demand for in-path caching and retransmission.

Several proposals in the context of INP rely on Programming Protocol-independent Packet Processors (P4) [11], [12], an open-source domain-specific language for network devices. For example, [13] combines P4-enabled switches with edge computing. The authors offloaded critical lightweight (sub)tasks to network devices while keeping the others on edge-computing resources. [14] uses P4-enabled switches to aggregate packets sharing common headers into a single one and to disaggregate the aggregated packet before reaching the destination. [15] proposes to program P4-enabled switches to generate an alarm if MQTT messages convey values exceeding a given threshold.

In previous work, we provided principles and design guidelines for QoS-enabled semantic routing [16]. Semantic routing is based on additional semantics other than mere IP addresses. Specifically, semantic routing may also depend on policy coded in, configured at, or signaled to network devices [17]. This means network devices may operate in, above, and below the network layer. Accordingly, a network device is intended as an element that receives/transmits packets and performs network functions on them, such as forwarding, dropping, filtering, and packet header (or payload) manipulation. We then used those principles and guidelines to design a framework for QoS-enabled semantic routing in industrial networks [18]. In this work, we detail a protocol suite to rule how the framework components interact one each other.

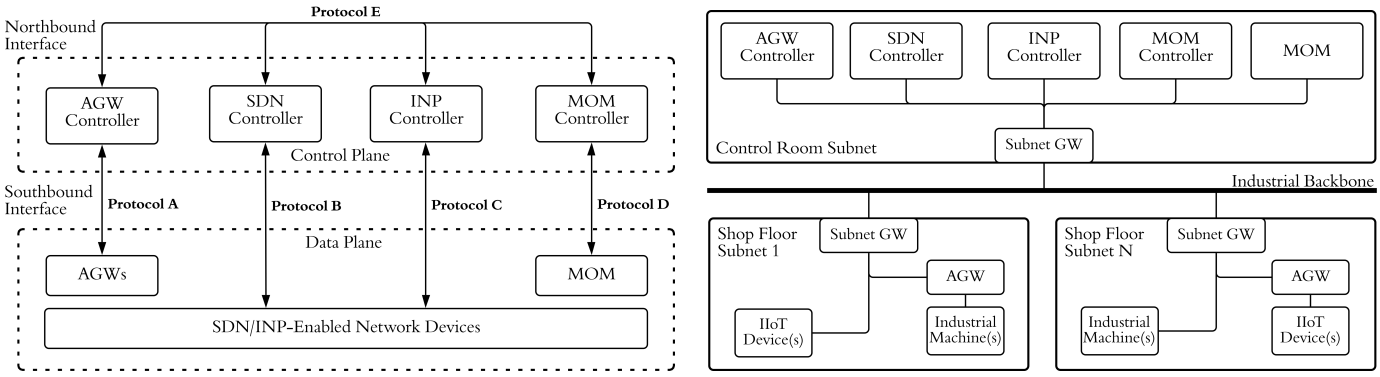
III. Requirements

Modern industrial environments have demanding communications requirements. In this regard, it is worth pointing out some constraints imposed by the industrial domain while seeking to satisfy such requirements. In particular, industrial machines typically tend to have an extremely long lifetime (i.e., decades) compared to the rate of innovation in computer science, high costs (i.e., ranging from thousands to millions of USD), and strict policies forbidding future updates unless provided by the manufacturer. Accordingly, a suitable communications substrate should live within the constraints of the industrial domain while satisfying the following requirements.

First, traffic flows traversing the network should be appropriately prioritized based on the information they carry. In this regard, it is paramount to distinguish between mission-critical and non-mission-critical traffic flows. For instance, mission-critical flows could notify human operators about (potential) hazardous circumstances, thus such flows should reach the destination as close to real-time as possible. At the opposite end of the spectrum, non-mission-critical flows carry non-urgent information. Note that non-mission-critical flows are typically way larger than mission-critical ones, meaning the former may take resources at the latter's expense if not adequately managed. This is where per-flow traffic management comes in. Specifically, traffic flows should be prioritized (SDN side) so that safety-related flows go first, then monitoring ones. However, flow prioritization may not be enough in the case of several huge flows traversing the network concurrently. Given that network resources are not as abundant as in the past compared to the traffic volume produced on the shop floor, network devices should be able to apply data filtering if needed and data aggregation whenever possible (INP side).

Second, shop floor components should communicate straightforwardly despite the plethora of protocols within the industrial domain. This requires connectors (AGWs) to normalize such protocols. Once normalized, intermediate nodes (i.e., network devices) can perform actions on packet headers and payloads according to a given semantic. For example, a network device may perform data aggregation by averaging consecutive temperature values or may perform data filtering by dropping values under a given threshold.

Lastly, shop floor components should seamlessly communicate on the move. This means the overall performance should not be affected while moving from one shop floor subnet to another. As a notable example, let us consider the case of an automated guided vehicle moving from a shop floor subnet to another while carrying spare parts. As the vehicle



(a) Functional/layered view.

(b) Network topology.

Fig. 1: Architecture overview.

changes subnet, so does its subnet gateway. As a result, the traffic flow(s) produced and consumed by the vehicle should be rescheduled accordingly. It is worth noting that flow rescheduling requires not only reconfiguring network devices dynamically but also doing so in compliance with other flows already in place along the new path(s).

IV. Architecture Overview and Protocols

As mentioned above, the solution we propose operates at both the application and network layers. The architecture design follows the SDN approach, clearly distinguishing the control plane from the data plane and identifying the interfaces (i.e., protocols) between such planes. In this regard, Fig. 1a provides a layered architecture overview.

The control plane comprises AGW, MOM, SDN, and INP controllers to monitor and configure AGWs, MOM, and SDN/INP-enabled network devices, respectively. The rationale behind the architectural components is the following. AGWs sit close to shop floor components that cannot support our protocols. For example, some legacy industrial machines may not support IP-based communications and, at the same time, cannot even be updated due to restrictive manufacturer policies. Then, the MOM decouples senders and receivers, sorts messages in topics of interest, and provides delivery semantics. Although the MOM enables critical features in message dispatching, it does not control how packets traverse the network. This is where SDN and INP come in. Specifically, SDN-enabled network devices put into action fine-grained traffic management (flow steering and prioritization), whereas INP-enabled network devices perform in-network traffic optimization (data filtering and aggregation).

Fig. 1b places control and data plane components within the industrial domain, as described above. Specifically, the controllers and the MOM are in the control room subnet (plant/enterprise level). SDN/INP-enabled network devices are both subnet gateways and industrial backbone elements. Lastly, AGWs are in the shop floor subnets, close to the shop floor entities not compliant with the architecture interfaces. Note that each shop floor subnet may be provided with one or more AGWs, and a single AGW may serve one or more shop floor entities.

The protocols used by the links and the interaction flows between the architectural components are the following. Starting from the lowest layer, in the data plane we find machine data packets enriched by the respective AGWs with the Data Header (Section IV-A), adopted to identify the data flow and the generating machine in a unique way. Protocols A to D (Sections IV-B to IV-E) implement the southbound interfaces of our architecture and command AGWs, SDN/INP-enabled network devices, and MOM. Lastly, Protocol E (Section IV-F) commands the infrastructure and configures the control plane, synchronizing all components on a unique shared state. For the sake of clearness, in the final subsection (Section IV-G), we report the sequence diagram in case of new topic subscription.

A. Data Header

The Data Header is attached to every packet traversing the system by the AGWs of our architecture. Each component that receives a new unforeseen Data Header in the packet forwards it to its controller and waits for routing/processing/flow rules to be set.

Table I shows an implementation of the Data Header, which keeps track of three fields:

- **flowID**: 16-bit user-defined integer that identifies the flow in a unique way in the system.
- **machineId**: 16-bit machine identifier. It can be logically set based on user necessities.
- **machineSerial**: String displaying the machine serial number. It can be dynamically read from machine registers or manually set on the AGWs.

TABLE I
Structure of Data Header

Field	Type
flowId	int16
machineId	int16
machineSerial	String

TABLE II
PROTOCOL A: AGW CONTROLLER TO AGWS

Field	Type
header	Data Header
crud	2bit
ttl	uint32
ipFrom	ipAddr
ipTo	ipAddr
destTopic	String
semanticDelivery	3bit
machineProtocol	String
machineUrl	String
pollingInterval	int8
geoPosition	geoURI [19]
applicationType	String

B. Protocol A

Protocol A is adopted between the AGW controller and AGWs. Its purpose is to pass information and configurations about which machine to connect for gathering data and with which frequency. Moreover, it contains the header field, to be put as header of every message gathered by the specific AGW.

Table II details Protocol A:

- **header**: the Data Header (Section IV-A), applied to each packet outgoing from the AGW.
- **crud**: 2 bit flags for identifying Create, Read, Update and Delete of a new or existent configuration.
- **ttl**:time to live [ms] of the configuration. After that time the AGW stops sending out new messages. If set to 0 the configuration is permanent.
- **ipFrom**: IP address of the AGW interface on which send out messages through the platform.
- **ipTo**: IP address of the destination MOM.
- **destTopic**: destination topic of the messages.
- **semanticDelivery**: 3 bits identifying the semantic of the flow in the MOM. Typically, it can be at-most-once, at-least-once, and exactly-once, but others can be defined based on the specific implementation of the MOM.
- **machineProtocol**: protocol for extracting data from the machine. Examples can be MODBUS [20], Profibus [21], EtherCAT [22], and OPC-UA [23].
- **machineUrl**: Url address of the machine supervised by the AGW.
- **pollingInterval**: interval in [ms] for polling data extraction.
- **geoPosition**: position in space of the machine supervised by the AGW expressed in compliance to RFC 5870 [19].
- **applicationType**: application type header attached to the message body by the AGW. Examples can be 'application/json' or 'application/xml'.

C. Protocol B

Open Networking Foundation - "OpenFlow Switch Specification" [24] is adopted as Protocol B to enable the interaction between SDN controller and the SDN-enabled network devices. The SDN approach decouples data from control plane access, making the introduction of new network functionalities simple and well structured on SDN-compliant hardware. A custom SDN controller uses the data shared with the Protocol E (Section IV-F) to control switches and to forward the data to the correct INP-enabled network device for processing and, lastly, to the MOM.

TABLE III
PROTOCOL D': MOM CONTROLLER TO MOM

Field	Type
header	Data Header
crud	2bit
ttl	uint32
semanticDelivery	3bit
forwardOnTopics	List<String>

D. Protocol C

We have selected the P4 [12] language to define data structures of this level. Unlike a generic language like C or Python, P4 is a domain-specific language with a set of constructs optimized for network data forwarding.

In our architecture, the P4 language is used to administrate the INP-enabled network devices and to create processing pipelines that digest the messages in the AGW-MOM path.

E. Protocol D

Protocol D is in charge of configuring and exchanging information between MOM controller and MOM. Protocol D is further split into two sub-protocols (i.e. D' and D") as follows.

Protocol D' configures the MOM to forward on a list of topics the data with the specified header and to exchange the semantic delivery QoS settings. The D' message is triggered every time that the MOM receives a new unforeseen flow in input and requires the MOM controller flow information.

Protocol D" is used to communicate the topic subscribers to the control plane to correctly set the network paths with the specified priority. The D" message is sent by the MOM to the controller every time there is a new subscription to the managed topics. It is necessary for guaranteeing the QoS also between the MOM and the subscribers and triggers an update on the SDN controller via the Protocol E (Section IV-F).

Table III reports Protocol D':

- **header**: the Data Header (Section IV-A), applied to each configuration for identifying the traffic on which to apply the rule.
- **crud**: 2 bit flags for identifying Create, Read, Update and Delete of a new or existent configuration.
- **ttl**: time to live [ms] of the configuration. After that time the MOM stops forwarding/prioritizing the flow. If set to 0 the configuration is permanent.
- **semanticDelivery**: 3 bits identifying the semantic of the flow in the MOM. Typically, it can be at-most-once, at-least-once, and exactly-once, but others can be defined based on the specific implementation of the MOM.
- **forwardOnTopics**: the topics on which to forward the specified flow.

Table IV reports Protocol D":

TABLE IV
PROTOCOL D": MOM TO MOM CONTROLLER

Field	Type
header	Data Header
crud	2bit
ipFromMom	ipAddr
topic	String
priority	3bit
subscribers	List<ipAddr>

- **header**: the Data Header (Section IV-A), applied to each configuration for identifying the traffic on which to apply the rule.
- **crud**: 2 bit flags for identifying Create, Read, Update and Delete of a new or existent configuration.
- **ipFromMom**: IP address of the MOM interface on which send out messages through the platform.
- **topic**: starting topic of the messages.
- **priority**: SDN priority of the flow outcoming the MOM.
- **subscribers**: a list of IP addresses subscribed to the specified topic. They are necessary for updating the SDN rules.

F. Protocol E

The northbound interface is managed in a uniform way between all entities by sharing information via Protocol E. Protocol E is configures the system and shares the state of the incoming data transmissions between all the controllers. It contains all the fields that we defined in the previous subsections together with the crud flags marking and differentiating new configurations from updates.

Table V describes Protocol E:

- **header**: the Data Header (Section IV-A), applied to each configuration for identifying the traffic on which to apply the rule.
- **crud**: 2 bit flags for identifying Create, Read, Update and Delete of a new or existent configuration.
- **ttl**: time to live [ms] of the configuration.
- **semanticDelivery**: 3 bits identifying the semantic of the flow in the MOM.
- **priority**: SDN priority of the flow outcoming the MOM.
- **ipFrom**: IP address of the AGW interface on which send out messages through the platform.
- **ipFromMom**: IP address of the MOM interface on which send out messages through the platform.
- **ipTo**: IP address of the destination MOM.
- **destTopic**: destination topic of the messages.
- **forwardOnTopics**: the topics on which to forward the specified flow.
- **subscribers**: a list of IP addresses subscribed to the specified topic.
- **inp**: a list of field coupled with the specific INP function to apply on.
- **applicationType**: application type header attached to the message body by the AGW.
- **geoPosition**: position in space of the machine supervised by the AGW expressed in compliance to RFC 5870 [19].

TABLE V
PROTOCOL E: CONTROLLER TO CONTROLLER

Field	Type
header	Data Header
crud	2bit
ttl	uint32
semanticDelivery	3bit
priority	3bit
ipFrom	ipAddr
ipFromMom	ipAddr
ipTo	ipAddr
destTopic	String
forwardOnTopics	List<String>
subscribers	List<ipAddr>
inp	List<(field: String, func: String)>
applicationType	String
geoPosition	geoURI [19]
machineProtocol	String
machineUrl	String
pollingInterval	int8

- **machineProtocol**: protocol used by AGWs for extracting data from the machine.
- **machineUrl**: Url address of the machine supervised by the AGW.
- **pollingInterval**: interval in [ms] for polling data extraction.

G. New Topic Subscription Sequence Diagram

Fig. 2 presents the sequence diagram in case of new topic subscription. By delving into finer details, the subscription of a new data consumer to the MOM triggers an interaction with Protocol D" (Table IV). Such interaction is handled by the MOM controller, which triggers a proper reconfiguration of the underlying SDN-enabled network devices.

Fig. 2 details the sequence diagram showing the interactions triggered by the addition of a new subscriber to a MOM topic. The MOM communicates the updated list of subscriber IPs to the MOM controller via Protocol D". MOM controller unleashes a synchronization phase (Protocol E) within the northbound interface to the SDN controller.

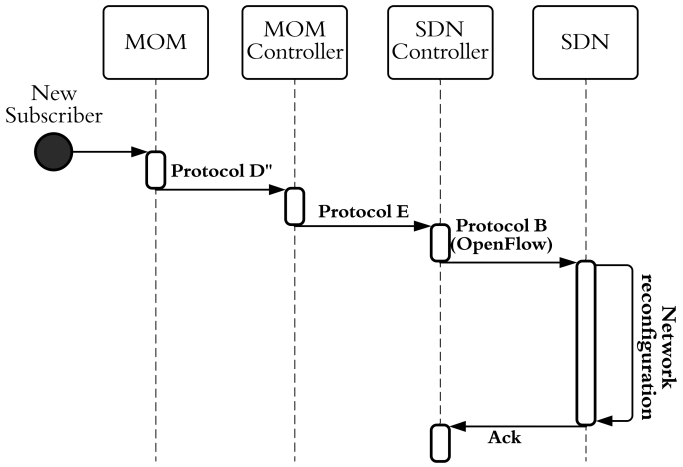


Fig. 2: New topic subscription sequence diagram.

The SDN controller has complete knowledge of the network components and thus chooses the best path to achieve the desired priority based on provided priority bits. Once the best path is calculated, the SDN controller configures the network equipment with Protocol B (OpenFlow) to route the packets accordingly from the MOM to the new subscriber, guaranteeing the correct QoS to the final data consumer.

V. Conclusions and Future Work

This paper extends our previous framework for QoS-enabled semantic routing in industrial networks, exploiting the SDN-INP interplay to enable the semantic routing of traffic flows within a MOM-based Industry 4.0 environment. In this work, we detailed how the architectural components interact based on a novel protocol suite designed to ensure challenging QoS requirements in terms of flow steering and prioritization, protocol normalization, and traffic optimization. We believe that the proposed protocol suite fits the characteristics of modern manufacturing environments, by providing an overlay networking solution that supports a semantic routing substrate operating both at the application and network layers. In the future, we intend to extend our solution by allowing the dynamic deployment and adoption of novel industrial machines and control room components seamlessly, also supporting the federation among different geographically distributed manufacturing plants.

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, 2019.

[2] E. Haleplidis, K. Pentikousis, S. Denazis, J. H. Salim, D. Meyer, and O. Koufopavlou, "Software-Defined Networking (SDN): Layers and Architecture Terminology," RFC 7426, Jan. 2015. [Online]. Available: https://www.rfc-editor.org/info/rfc7426

[3] 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.

[4] M. Fogli, C. Giannelli, and C. Stefanelli, "Software-defined networking in wireless ad hoc scenarios: Objectives and control architectures," *Journal of Network and Computer Applications*, p. 103387, 2022.

[5] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "Openflow: Enabling innovation in campus networks," *SIGCOMM Comput. Commun. Rev.*, vol. 38, no. 2, p. 69–74, mar 2008.

[6] 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.

[7] 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.

[8] "In-network computing," last visited in Apr. 2022. [Online]. Available: https://www.sigarch.org/in-network-computing-draft/

[9] D. R. K. Ports and J. Nelson, "When should the network be the computer?" in *Proceedings of the Workshop on Hot Topics in Operating Systems*, ser. HotOS '19. New York, NY, USA: Association for Computing Machinery, 2019, p. 209–215. [Online]. Available: https://doi.org/10.1145/3317550.3321439

[10] 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.

[11] 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.

[12] "P4 language specification," last visited in Apr. 2022. [Online]. Available: https://p4.org/p4-spec/docs/P4-16-v1.2.1.html

[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. . 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.

[15] 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.

[16] 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)*. IEEE, 2021, pp. 1–6.

[17] A. Farrel and D. King, "An Introduction to Semantic Routing," Internet Engineering Task Force, Internet-Draft draft-farrel-irtf-introduction-to-semantic-routing-03, Jan. 2022, work in Progress. [Online]. Available: https://datatracker.ietf.org/doc/html/draft-farrel-irtf-introduction-to-semantic-routing-03

[18] P. Bellavista, L. Foschini, L. Patera, M. Fogli, C. Giannelli, C. Stefanelli, and Z. Lou, "A Framework for QoS-Enabled Semantic Routing in Industrial Networks," Internet Engineering Task Force, Internet-Draft draft-bellavista-semantic-sdn-mom-00, Mar. 2022, work in Progress. [Online]. Available: https://datatracker.ietf.org/doc/html/draft-bellavista-semantic-sdn-mom-00

[19] C. Spanring and A. Mayrhofer, "A Uniform Resource Identifier for Geographic Locations ('geo' URI)," RFC 5870, Jun. 2010. [Online]. Available: https://www.rfc-editor.org/info/rfc5870

[20] "Modbus 101 - introduction to modbus," last visited in Apr. 2022. [Online]. Available: https://www.csimn.com/CSI_pages/Modbus101.html

[21] "Profibus," last visited in Apr. 2022. [Online]. Available: https://www.profibus.com/technology/profibus

[22] "Ethercat - the ethernet fieldbus," last visited in Apr. 2022. [Online]. Available: https://www.ethercat.org/en/technology.html

[23] "Unified architecture," last visited in Apr. 2022. [Online]. Available: https://opcfoundation.org/developer-tools/specifications-unified-architecture

[24] "Openflow switch specification," last visited in Apr. 2022. [Online]. Available: https://opennetworking.org/wp-content/uploads/2014/10/openflow-switch-v1.5.1.pdf