Towards Intelligent Data Protocols for the Edge

Praveen Kumar Donta *Senior Member, IEEE,* and Schahram Dustdar *Fellow, IEEE*

Distributed Systems Group, TU Wien, Vienna 1040, Austria.
{pdonta,dustdar}@dsg.tuwien.ac.at

***Abstract*—The computing continuum is growing because multiple devices are added daily. Edge devices play a key role in this because computation is decentralized or distributed. Edge computing is advanced by using AI/ML algorithms to become more intelligent. Besides, Edge data protocols are useful for transmitting or receiving data between devices. Since, computation efficiency is possible when the data is received at the Edge timely, and it is possible only when the data protocols are efficient, reliable and fast. Most edge data protocols are defined with static set of rules and their primary purpose is to provide standardized and reliable data communications. Edge devices need autonomous or dynamic protocols that enable interoperability, autonomous decision making, scalability, and adaptability. This paper examines the limitations of popular data protocols used in edge networks, the need for intelligent data protocols, and their implications. We also explore possible ways to simplify learning for edge devices and discuss how intelligent data protocols can mitigate challenges such as congestion, message filtering, message expiration, prioritization, and resource handling.**

***Index Terms*—Data protocols; Intelligent protocols; Edge Computing; Computing Continuum; Edge Intelligence; Autonomous decision-making**

I. Introduction

In the computing continuum, many diverse devices are connected, including tiny sensors, actuators, edge systems, and cloud systems. In such an architecture, computations shift from centralized to distributed and decentralized, processing data in real-time and enabling faster decision-making [1]. Furthermore, Edge devices are a key storage and processing element in the computing continuum. Although they are close to the data source, they are tiny in size, limited in storage and processing capacity, and low in power. But, efficient utilization of edge devices improves privacy, reliability, and scalability while minimizing latency, cost, and bandwidth requirements [2]–[4]. In the recent years, machine learning (ML) and artificial intelligence (AI) have helped transitioning edge computing to edge intelligence (EI) by providing advanced analytics and smart and autonomous decision-making systems. In the recent years, EI has become a frequent research topic because of its growing use in various intelligent applications and services [5]. Currently, EI is limited to computation or decision-making. As per our concern, EI necessarily consider several other parameters (e.g., how smartly it can receive data, or forward data or make decisions) in addition to the current intelligence mechanisms [6]. Data protocols allow devices to communicate with each other, so making them intelligent is one way to add more intelligence to the Edge [7].

In general, protocols are a set of rules or standards defining how two devices are connected and how they communicate their data. Protocols are needed at the edge to communicate with sensors and actuators, the cloud, or the data center [8]. These protocols are categorized into network and data protocols [9]. Network protocols enable connections between devices (edge-to-edge, edge-to-cloud, or edge-to-things, or vice versa). Most popular network protocols include Bluetooth, WiFi, ZigBee, LoRa, LoRaWAN, or NB-IoT [10], [11]. Data protocols control data exchange between devices. The majority of these data protocols follow request-response or publish-subscribe models [7]. In the request-response model, a client interacts with the server in a synchronous communication pattern, which means client wait for the server to respond before continuing. The most popular request-response protocols for the edge are Hypertext Transfer Protocol (HTTP) and Constrained Application Protocol (CoAP). In the publish-subscribe model, publishers send their information to a message broker in an asynchronous communication pattern, so they do not wait for the broker's response. Most popular publish-subscribe protocols include Advanced Message Queuing Protocol (AMQP), Message Queue Telemetry Transport (MQTT), and Data Distribution Service (DDS) [12]. The primary focus of this paper is on data protocols, and the rest of the article focuses on them (further discussion on these protocols is available in Section II).

Currently, available data protocols are composed of static rules, and the primary goals of these protocols are efficient and accurate data transmissions, reliability, scalability, and achieving optimized speed in data delivery. The computing continuum is ever-growing rapidly with a huge variety of computing devices connecting every day, the traditional protocols continually upgrade their rules according to the newly added devices or the devices to be manufactured. Embedding intelligence into data protocols makes the edge more intelligent with efficient and reliable communications, adaptable to environments or applications, predictive maintenance, autonomous decision-making, and interoperability. Recent advances in AI/ML can be used to achieve these goal. In this context, this paper discusses intelligent data protocols for the next-generation intelligent edge from a bird-eye-view perspective. The outline of this paper is as follows:

1) Initially, we will discuss currently available popular data protocols, their general structures or working models, and their limitations. We summarize the different challenges of these protocols.
2) We discuss the possible ways or focuses to make existing protocols intelligent, and the learning algorithms that can

be used to achieve it.

3) We summarize the challenges of using different learning strategies on resource-constrained devices along with possible future research directions.

This paper is organized as follows. Section II presents existing data protocols for the edge along with their working model and advantages. Section III highlights the various challenges associated with traditional data protocols. Section IV discusses the use of intelligent data protocols in next-generation Edge computing and explores the implications of adding intelligence to data protocols. Additionally, this section examines possible ways to simplify learning for constrained edge protocols. Finally, Section V concludes the paper.

II. Existing Data Protocols for Edge

There are several data protocols in the literature with their own advantages and limitations. But, very few developments have taken place in recent years due to constraints such as the availability of resources, the complexity of designing protocols for ever-growing edge networks (scalability and applications), challenges associated with interoperability, and lack of standardization. In this section, we discuss the most popular data protocols and their recent enhancements.

A. HTTP

It is one of the most commonly used protocols for devices connected over the Internet. It is a stateless protocol that does not store or use its history in any way [13]. When an edge device (client) sends a message to another edge/cloud (server), servers respond according to the message [14]. Since it does not use or store history, it is very difficult to apply AI/ML or make this protocol intelligent. HTTP supports several formats including plain text, documents, images, audio, and video. Since HTTP supports larger file formats, it needs more time to transfer, more buffer availability (temporary storage), consumes more battery life,, and takes a high response or turnaround time compared to other data protocols. It is a reliable data delivery protocol, but not secure. In the past decade, HTTP has improved version HTTPS with the security enabled feature [15], [16].

B. CoAP

CoAP is a lightweight request-response protocol for constrained devices like the IoT or the Edge. This protocol supports binary data and Extensible Markup Language (XML). CoAP uses limited memory, energy, bandwidth, and computational requirements compared to the HTTP protocol, and CoAP uses User Datagram Protocol (UDP). A CoAP ensures data confidentiality, availability, and integrity as it traverses the various nodes (Edge/Fog/Cloud) and gateways in a computing continuum. In CoAP, an edge device can store a copy of sent data locally until the server receives it. This feature mitigates the risk of missing or lost data in any transmission or further benefits analytics and decision-making. Considering the low bandwidth and buffer, and the high number of devices connected and exchanging data, this feature can cause a high congestion level. Several enhancements are made in the literature to mitigate congestion in the CoAP protocol [17]. Figure 1 summarizes a general congestion control model.

From Fig. 1, we notice that most CoAP congestion control strategies use minimum and maximum Retransmission Time (RTT) and Retransmission Timeout (RTO) from history to generate new RTO for future data transmissions. These strategies include Binary Exponential Backoff (BEB) [18], Variable Backoff Factor (VBF) [19], Probabilistic Backoff Function (PBF) [20], and Fibonacci Pre-Increment Backoff (FPB) [21]. Several approaches have recently been developed using machine learning [22] and Deep Reinforcement Learning (DRL) [17] methods to determine efficient RTOs. In the end, these protocols control the congestion, but they are not able to predict or mitigate it completely. There are several other approaches that focus on enhancing the CoAP protocol further by adding features such as security, remote accessibility, multimedia data transmission, and cache management. But, there is huge research needed on CoAP according to recent advancements in the computing continuum. These advancements include computing dynamic RTO to avoid congestion and latency, privacy and security mechanisms, working according to the dynamic condition of the environment or applications, and prioritizing data delivery. On the other hand, the developments in 5G and 6G ensure solutions for bandwidth problems in CoAP.

C. MQTT

MQTT is a publish-subscribe data protocol for low-constrained devices. This is one of the most popular protocols for IoT because of several benefits including reliable data delivery, low bandwidth, lightweight (headers and control messages are tiny in size), efficiency (fast delivery), and secure [23]. MQTT enables the communication between the entire computing continuum, for e.g., sensor nodes (MQTT-SN [24])-to-Edge-to-Cloud and vice versa. The general working model for MQTT is summarized in Figure 2. The primary components of the MQTT protocol are the client (either Publisher or Subscriber), and the broker. Any device in the computing continuum (IoT/Edge/Fog/Cloud) can act as a client, and its key responsibility is to send a message (called a publisher) to a broker or receive a message from a broker (called a subscriber). MQTT broker is a central entity, which authenticates clients and authorizes their messages, receives messages from publishers, filters messages according to the topics (topic is a keyword used to filter messages), and sends these messages to the appropriate subscriber. Each MQTT broker holds a message according to its expiry time (called message expiry), and this time varies depending on the application. It is also responsible for deciding the priority of each message before it is delivered. MQTT supports a range of formats, including plain text, XML, and JSON formats. Since the broker is a central entity responsible for most work, its failure causes major damage.

Recently, researchers have added several features to MQTT protocols for delay minimization [25], device location iden-

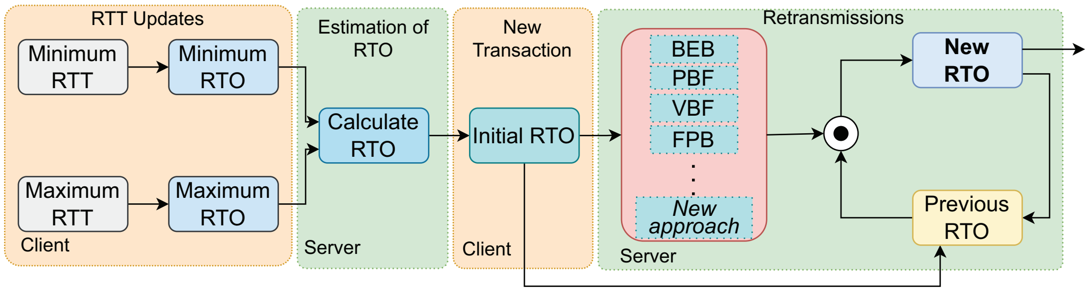

Fig. 1: RTO estimation of CoAP and its enhancements to control high congestion

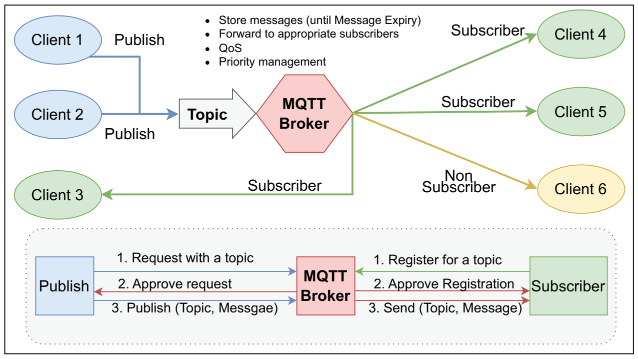

Fig. 2: General message exchange model of MQTT protocol

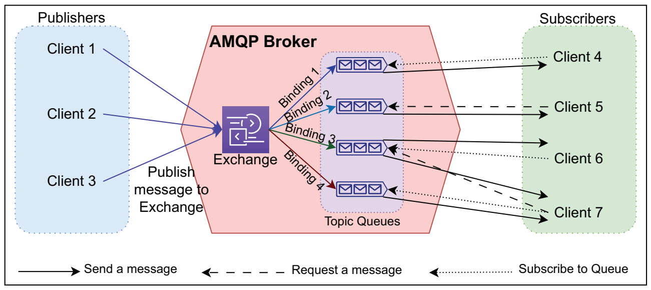

Fig. 3: General message exchange model of AMQP protocol

tification [26], as well as several improvements to message queues. The message queues include RabbitMQ, Mosquitto, modified Mosquitto, HiveMQ, ZeroMQ, VerneMQ, HornetQ, MicrosoftMQ, AmazonMQ, ApacheQPID, and OpenMQ [7]. RabbitMQ and Mosquitto are the most popular message queues for the majority of applications because of their flexibility. Although there have been many advancements in MQTT over the past few years, no one has added intelligence to it. To ensure efficient and reliable data delivery using the MQTT protocol, there is a need for more development to make it more intelligent, especially on dynamic priority and message expiry depending on the dynamic conditions of the computing continuum.

D. AMQP

AMQP is a publish-subscribe protocol for point-to-point message exchange between two devices. It is designed to provide a flexible and scalable way to send and receive messages. It supports different message exchange patterns and messaging topologies [27]. AMQP's general message transmission structure is similar to MQTT, but it contains multiple queues to maintain each topic as shown in Figure 3. The AMQP structure is composed of clients (both publishers and subscribers), an AMQP broker, and multiple queues. The publisher can send a message to the AMQP broker, specifying the destination and any other relevant metadata. The AMQP broker stores messages temporarily, filters them, and pushes them to the appropriate topic queues. Clients can subscribe to a specific topic queue to receive messages in a timely manner. Clients can also request messages even if they are not subscribed to a specific topic queue. AMQP is widely used in industrial applications because of its interoperability support. AMQP provides a reliable message delivery and a scalable message architecture. It also supports secure and high-quality data exchange between devices [28]. Like MQTT, AMQP supports binary data, plain text, XML, and JSON formats.

AMQP supports different message queues including RabbitMQ, Apache ActiveMQ, RedHat AMQ, and Apache Apollo MQ. Because AMQP delivers high-quality messages, it uses more memory and computation. In addition, this process also increases delay. This protocol is feasible for a large number of devices, and not suitable for small-scale devices. This protocol is highly scalable but not cost-effective. There is huge potential for integrating intelligence in this protocol in large environments like the computing continuum.

E. DDS

DDS is a data-centric topic-based publish-subscribe protocol that does not use message brokers for data exchange. It delivers high Quality of Service (QoS) messages using a multicasting approach. The message exchange process of the DDS protocol is shown in Figure 4. Instead of message brokers, DDS uses a global databus that shares space or is distributed between all topics, which helps to avoid single points of failure. But, like other publish-subscribe protocols, clients play subscriber or publisher roles. Both publishers and

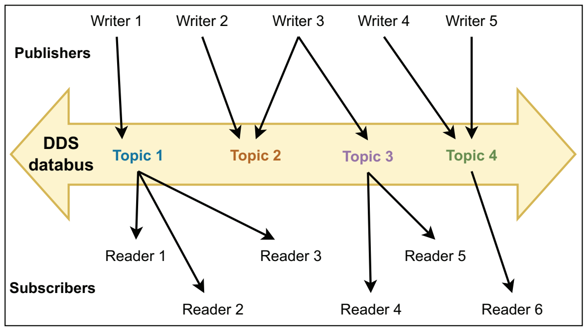

Fig. 4: General working model of DDS protocol

subscribers share the DDS databus dynamically and can join or leave anytime [29]. A DDS protocol can be used for a variety of applications, ranging from IoT to Edge to Fog to Cloud. It transmits data at ultra-high speeds (low latency), manages thousands of devices, and is highly available and secure. It can handle small to large systems. Security in DDS is distributed across the continuum without affecting the performance of the system. Data can be stored in the DDS databus until all the subscribed clients receive it, so the publisher does not need to store it locally with the application.

DDS is the first protocol developed with interoperability features. Several other factors make this protocol chosen by a vast number of computing continuum applications. There are limited non-standard developments of DDS protocols focused on security, and message filtering approaches. *White et al.* [30], embedded a security mechanism for DDS protocols which is applicable in industrial IoT, and supports identifying malicious heartbeat messages and denial of service attacks. In general, the standardized DDS supports Simple Discovery Protocol (SDP), but it is inefficient when the application grows with many devices in the computing continuum. There are a few researchers who tried to overcome this challenge. A content-based topic filtering strategy was introduced in [31] which takes limited memory and less computing time with higher filtering efficiency and timely message delivery. *Mustaque et al.* implemented a message filtering strategy based on causality to achieve higher availability and efficiency when multiple similar services are connected [32]. To improve DDS SDP scalability, a Bloom Filters (BFs) strategy is introduced in [33], and they use the Hash function to represent the space-efficient probabilistic datasets.

III. Challenges of Existing Data Protocols

There are several challenges associated with existing data protocols, which are summarized using Table I, and discussed below:

A. Latency

Low latency is extremely critical for constrained devices such as IoT or Edge. Several factors contribute to high latency in the data protocols, including waiting for an acknowledgment to send the packets to the server, delay at message brokers, inefficient queue management, inefficient message expiry time, and starvation due to prioritization. In some cases, high latency causes further problems, such as congestion and inefficient bandwidth use, which further complicate upcoming packets. HTTP and AMQP result in high latency because of waiting for the server response and using high-quality data transmissions, respectively. These two protocols support several data formats that are larger. CoAP is better in terms of latency than HTTP and AMQP. But CoAP cause a high congestion rate that sometimes improves retransmissions count for a lost packet. This re-transmitted packets cause higher latency. DDS results in low latency because it does not use message brokers like MQTT. Because of its tiny headers, MQTT results in low latency in comparison with HTTP, CoAP, and AMQP.

B. Scalability

As the computing continuum grows, more devices are participating in computations. It is one of the most critical metrics because of the complications caused by factors such as connectivity, authorization or certification, or handling timely updates from software firms. So, it is necessary for any data protocols to support scalability, so that newly added devices are adaptable and easily communicated. From the literature, HTTP and CoAP are scalable depending on the application, network constraints, and specific use cases. In contrast, these two protocols are not scalable for large systems like computing continuum or large numbers of edge devices. MQTT and AMQP are scalable because newly added devices need to register their participation as publishers or subscribers. DDS protocols are more flexible and scalable because the devices can join or quit anytime in the network, making it very easy and adoptable for newly added devices.

C. Security

Data protocols are designed to exchange data between two devices, so it is necessary to protect data before it is delivered to the destination. Although each protocol in the literature provides different levels of security, HTTP is not a secure protocol. There is a high chance of denial of service attacks on the HTTP protocol. So, the HTTP protocol has been enhanced with security, namely HTTPS. HTTPS and CoAP can provide security through Transport Layer Security (TLS) to provide secure data exchange between devices. MQTT uses authentication and authorization mechanisms to deliver data securely, but this does not meet security standards. AMQP provides security through authentication and authorization using the Simple Authentication and Security Layer (SASL) and supports TLS for secure data exchanges. DDS supports security through access control, authentication, integrity, and encryption mechanisms.

D. Complexity

Complexity, in this context, refers to implementing and managing data protocols efficiently and effectively to manage data exchanges, memory, and computations. HTTP is a traditional and standard protocol that simplifies the deployment

TABLE I: Summary on Challenges associated with Existing Data Protocols

Protocol	High Latency	Scalability	Security	High Complexity	Interoperability	Reliability
HTTP	✓	✗	✗	✗	✓	✓
CoAP	✗	✓	✗	✗	✗	✓
MQTT	✗	✗	✓	✗	✗	✗
AMQP	✓	✓	✓	✓	✓	✗
DDS	✗	✓	✗	✓	✓	✓

and management of applications. But, this protocol increases complexity and security issues in large-scale systems. The complexity of CoAP is similar to HTTP. Although MQTT is efficient for the Edge, it increases complexity when dealing with complex data formats. AMQP is one of the more complex protocols due to multiple features such as multiple queues, dynamic routing, interoperability, and security. But, these features make this protocol extremely relevant for many industrial applications. Similarly, the DDS protocol is also embedded with several features that are difficult to maintain and complex. Furthermore, these features make the DDS protocol more adoptable for many applications. Still, several advancements are needed in these protocols to make them more adaptable and intelligent.

E. Interoperability

It is the ability of two or more heterogeneous devices to work together and achieve a common goal. In the computing continuum, several heterogeneous devices are participating and exchanging their data. Achieving interoperability among these devices and coordinating efficiently and timely is necessary. HTTP supports different data formats and interoperable through specifying multiple `Content-Type` in the headers. But, CoAP protocols need a data format or CoAP request to be understandable by the server or receiver. So, it does not facilitate interoperability. MQTT does not support interoperability by default, but uses different message queues such as Apache ActiveMQ, or some set of configuration setups it can be supported. These changes in the protocols may affect the actual protocol performance. AMQP supports a high level of interoperability. DDS is the first protocol designed with interoperability features, and it is highly interoperable and suitable for edge devices.

F. Reliability

The reliability of a data protocol is a crucial metric for assessing its QoS. It ensures that the message is delivered accurately and on time. Data protocols are designed to ensure reliable message exchanges because they also play an important role in decision-making. HTTP and CoAP follow a request-response communication pattern, meaning each message is acknowledged and retransmitted in failure scenarios. So, they are highly reliable. The MQTT protocol is unreliable due to the possibility of a single point of failure at the message broker or because there is no retransmission policy if the message expires or is lost. Publishers and senders do not know whether their messages reach subscribers. As AMQP also uses a message broker, there is a chance of a single point of failure. However, it maintains multiple queues, so the loss is low compared to MQTT because failure in one queue does not cause failure in another. DDS achieves high QoS and is highly reliable [34]. DDS does not use message brokers, so there is no chance for a single point of failure.

IV. Intelligence for Data Protocols

This section discusses intelligence in data protocols and its implications for possible solutions. In this section, we explain how existing data protocols can be advanced through learning and enhanced in a way that enables intelligent decision-making. This further addresses several challenges such as congestion, message filtering, message expiry issues, resource control and dynamic prioritization. Next, we discuss various implications associated with incorporating intelligence into data protocols. Further, we discuss several ways to simplify learning algorithms with respect to constrained devices and protocols. Finally, we summarize all the discussion about the challenges.

A. Intelligence in Data Protocols

According to *Friston et al.* [35], ”intelligence is the ability to generate evidence for their own experience”. In data protocols, intelligence indicates the ability to generate efficient and dynamic decisions (in message queues, clients or servers), through past experiences or data patterns. There are several ways to embed intelligence in data protocols depending on their rules and structures. For example,

- CoAP protocol congestion control is one of the major challenges, and several mathematical or probabilistic models are used to control it. Adding intelligence in this module helps to identify congestion before it happens (Prediction), so that possible data failures or delays can be avoided. For instance, generating a dynamic RTO helps to identify the appropriate data delivery time for each data packet, so that data packets do not overlap and interrupt other packets or channels. In this way, congestion is completely mitigated.
- Protocols like MQTT and AMQP use a central message broker to exchange data between clients. And, a message broker is responsible for multiple operations including store and forward, prioritization, and QoS. The message broker is developing an intelligent mechanism that can determine the importance or sensitivity of data, and then determine its priority according to that. This method can be used to analyze message patterns through structure or representation learning strategies. But, it is also necessary to keep privacy and security in mind. It is also challenging

to assign a dynamic priority to each data packet. However, we can consider the history of data or business of the channel or receiver device, and a dynamic message expiry is generated for each data packet. This way, intelligence is beneficial to achieve high-level of intelligence in data protocols. When developing dynamic message expiry, it is necessary to treat low constrained devices like Edge. A low message timer expiry increases the packet loss rate and a high message expiry may cause overflow or congestion situations at the message broker.

- Currently, multiple computing devices connect through multiple applications, and edge devices play computing roles for multiple applications. So, it is necessary to assign priorities to messages based on how important they are. Since both MQTT and AMQP support message priority (AMQP priority ranges from 0-9, and MQTT priority from the perspective of QoS is 0-3) through their message headers. Adding intelligence to these protocols helps them become smart and reliable. However, it is a highly challenging decision, but it can help improve the performance of the Edge devices.
- DDS-like protocols replace message brokers with message filtering strategies. However, these strategies are unintelligent. Added intelligence to message filtering through natural language processing (NLP), AI/ML, or deep learning will significantly improve these protocols. For example,
  - NLP techniques such as *word2vec* [36] can identify the underlying topic or themes in messages, which can be useful for topic modeling.
  - NLP further extracts keywords from messages, which create topics dynamically. Similarly, due to the accuracy of AI/ML or deep learning in classification, dynamic message filtering can be performed based on the categories identified in messages.
  - AI/ML or deep learning techniques help identify anomalous messages in DDS.
  - From the literature, *Mustaque et al.* [32] used causality to filter messages and achieved increased efficiency and availability.

  There are several other strategies that will greatly help to achieve high-level intelligence in DDS protocol, including Markov blankets, free energy principle, structure or representation learning algorithms.

B. Implications of intelligent data protocols

Most data protocols designed for the edge and IoT are tiny in size, use minimal resources, and limited in configuration settings. For example, MQTT is very restricted in message formats, priority ranges (for example QoS0-QoS3) and sizes. So, it is necessary to use lightweight learning strategies, or learning models that learn through limited data. As a result, these protocols are also limited in memory and other resources. DDS and AMQP protocols' complexity is high due to several features, and it is necessary to trade off complexity before adding intelligence. For example, when AI/ML algorithms generate more topics during message filtering in DDS, it is very challenging for the subscribers to find the appropriate subscribers for the messages, and increases latency. Maintaining topic queues in AMQP is also challenging. To make this protocol more efficient and smart, efficient learning algorithms are very significant. Since these two protocols are resource-intensive, and AI/ML is resource-intensive, it does not become overloaded on existing models. CoAP's major challenge is congestion, according to *Donta et al.*'s DRL [17] and *Demir et al.*'s ML model [22] tries to add intelligence to it. But adding intelligence to the system will require higher resources or limit the number of devices at the edge. Since CoAP uses previous information such as RTT and RTO, keeping all information in memory for training models is an additional burden. So, efficient buffer management, lightweight ML approaches, AI/ML which produces efficient learning models through tiny data are appropriate for CoAP protocol, to make it intelligent and efficient.

C. Ways to simplify learning complexity

Most AI/ML algorithms are highly computational, and require huge computational resources to produce an efficient and accurate learning model. Despite this, data protocols work on constrained devices, such as the IoT or the Edge. It is difficult to keep large amounts of data to train learning models because of limited storage space. So, it is necessary to choose AI/ML algorithms for data protocols carefully. Such chosen algorithms must be able to generate accurate and efficient learning models using limited data and low computational resources. The following techniques are useful to solve computational challenges for data protocols when adding intelligence to data protocols.

1) AutoML: Automated machine learning (AutoML) is becoming more popular because of its advantages in data analytics. It simplifies the ML model building process through automatic feature selection, hyperparameter tuning, and model selection [37]. However, it is not popular for small datasets, but it is still capable of producing accurate and efficient models [38]. So, there is a high chance of enabling intelligence in data protocols through AutoML techniques.

Most scenarios or applications, it is difficult to predict or identify the best learning model to fit or resolve the problem, in such cases AutoML is one possible solution. Depending on the availability of data or computational resources, it can choose the most appropriate algorithm/s to learn the models. AutoML is more efficient at classifying supervised or unsupervised data, which is further applied in DDS for message filtering. Extending this towards multi-label classification (MLC) [39] enables more intelligence concerning topic filtering in the DDS, further improves reliable message delivery. Similarly, MLC can also be applied to AMQP to push published messages to appropriate topic queues dynamically. AutoML can also be applied to identify common patterns that can be used to route messages. Using these predictions, DDS or AMQP can support more resources to mitigate message drops or achieve high reliability on these routes. AutoML can help MQTT or AMQP identify message priorities according to message context or subscribers' behaviours.

2) Representation learning: It extracts meaningful information, latent features or underlying exploratory factors from the input datasets. Representation learning (ReL) further extracts internal structures or patterns from the data. There are several ReL algorithms in the literature with their benefits and drawbacks, or low to high computations [40]. The advantages of ReL algorithms are prominent for data protocols for edge to make them intelligent. It is also embedded in autoML, so that model selection or learning happens automatically. The possible challenges to be addressed through ReL are discussed subsequently.

MQTT brokers can consider ReL to learn a model from historical message expiry data, and extract patterns and relationships between various features such as message content, topic, and subscriber behavior, and analyze this information to predict when a message is likely to expire. Deciding or predicting a dynamic message expiry for published messages further helps to optimize message delivery and prevent unnecessary message loss. Representation learning algorithm is useful in CoAP protocols to identify the right time to transmit a packet. This is so that it is delivered in a reliable manner and does not cause congestion. This is possible through the analysis of the structure of the network, and message data. The ReL is also useful in DDS protocol to predict the route and enable needed resources for the data while the data is being delivered. By analyzing the history of data transmissions between publishers and subscribers, DDS can identify the busy routes in the network.

3) Complexity Minimization: There are several ways to minimize the complexity of learning algorithms and we listed some of them as follows.

- *Dimensionality reduction and coarse-gaining* are helpful in simplifying the datasets for further analysis. The dimension reduction is used to reduce the number of variables, whereas coarse-gaining simplifies the complex data to simple representations.
- *Move from big data to smart data:* Big data refers to highly complex and large unvalidated data, and consists of both structured and unstructured data that requires high-level processing units to run [41]. Smart data refers to data with meaningful information, validated, and well-defined, allowing information to be processed more efficiently. Smart data can be run on constrained devices with high accuracy and efficiency.
- *Disentangled*: refers to the process of separating different factors or components of a complex system, such as separating variables or features in a data set. These disentangled representations are simple to analyze and further improve model interpretability.

D. Summary

Overall, this paper uses intelligence not only to make protocols dynamic and smart, but also to mitigate several challenges. In the Table II following subsections, we summarize possible challenges able to address through learning while making the protocols intelligent.

TABLE II: Summary of possible challenges able to address through learning in different data protocols

Protocol	HTTP	CoAP	MQTT	AMQP	DDS
Congestion Control	✓	✓	✗	✗	✓
Message Filtering	✗	✗	✗	✓	✓
Message Expiry	✗	✗	✓	✓	✓
Prioritization	✗	✗	✓	✓	✗
Control on Resources	✗	✗	✓	✓	✓

1) Congestion control: In CoAP, congestion control is one of the primary challenges. It can be overcome through efficient RTO computations. It is promising to use ML or ReL algorithms since they can analyze the history and patterns of data packets to determine the most appropriate time to send them to the server or receiver. This allows them to achieve higher reliability. Embedding intelligence into CoAP will also help to predict the situation before it occurs, so that QoS can be increased.

2) Message filtering: Message or domain filtering is another challenge that is addressed through AI/ML and easily embedded to DDS or AMQP protocols. AMQP uses multiple message queues, and dynamic message filtering which helps to increase message queue utilization. Domain filtering in DDS performs an efficient delivery to appropriate subscribers and increases reliability. ML and ReL have the potential to make these protocols more intelligent, because of their ability to classify and identify patterns quickly and easily in their data sets.

3) Message expiry: As these protocols are designed for devices with limited buffer capacity, message expiration plays a crucial role in MQTT and AMQP's message queues. A minimal message expiration will cause additional unnecessary message drops, which further affects the delay caused by a retransmission. In case of a long message expiry, the messages remain alone in the message queue, thereby contributing to congestion and data overflow. It is therefore necessary to trade off message expiration. It is more reasonable if it can be dynamically generated based on message size and available resources. By analyzing historical data loss or timely delivery of messages, and their routes, AI/ML can generate dynamic message expiration dates.

4) Prioritization: MQTT and AMQP messages have a prioritization algorithm embedded in their headers, and they include message priority. MQTT supports three priorities, and AMQP supports 0-9, and is added once a message is created. It is recommended to decide the priority of each message by the message broker. This is to increase the reliability of the delegate based on the condition of the subscriber or channel through its history data analysis. Dynamic prioritization must improve the reliability of the protocols.

5) Control on resources: AI/ML algorithms predict busy routing paths, so it is easy to control or make them available on that path to avoid problems. This further improves protocol availability and reliability. As CoAP and HTTP require strict connection establishment before communication, they reserve the necessary resources before communicating. Similarly, protocols like MQTT, AMQP, and DDS enable dynamic routing with increased resource availability because of predicting routing information through history data analysis.

V. Conclusion

Data protocols are the key elements to move data from one device to another device for computation or storage (temporary or permanent storage). However, traditional protocols are a static set of rules, while recent advances enforce dynamic functionalities in these protocols, due to advancements in connected computing devices, and the scale of applications. In this context, this paper examines how existing protocols can be extended to enforce intelligence and mitigate challenges. Initially, we present popular data protocols for the edge and computing continuum, and study their benefits and limitations. We discuss various challenges associated with these protocols when extended further with additional features. As a result, we explore a number of research questions, including the need for intelligence for data protocols, the challenges associated with embedding intelligence into existing protocols, and possible ways to simplify their complexity. We summarize the findings in the paper while considering the most appropriate challenges to make the protocols intelligent.

Acknowledge

Research received funding from the EU's Horizon Europe Research and Innovation Program under Grant Agreement No. 101070186. EU website for TEADAL: https://www.teadal.eu/.

We thank Dr. Victor Casamayor Pujol (Distributed Systems Group, TU Wien, Austria) for helping with proofreading.

References

[1] S. Dustdar, V. C. Pujol, and P. K. Donta, "On distributed computing continuum systems," *IEEE Transactions on Knowledge and Data Engineering*, vol. 35, no. 4, pp. 4092–4105, 2023.

[2] D. Xu, T. Li, Y. Li, X. Su, S. Tarkoma, T. Jiang, J. Crowcroft, and P. Hui, "Edge intelligence: Architectures, challenges, and applications," *arXiv preprint arXiv:2003.12172*, 2020.

[3] V. Casamayor Pujol, P. K. Donta, A. Morichetta, I. Murturi, and S. Dustdar, "Distributed computing continuum systems–opportunities and research challenges," in *Service-Oriented Computing–ICSOC 2022 Workshops: ASOCA, AI-PA, FMCIoT, WESOACS 2022, Sevilla, Spain, November 29–December 2, 2022 Proceedings*. Springer, 2023, pp. 405–407.

[4] C. K. Dehury, P. K. Donta, S. Dustdar, and S. N. Srirama, "CCEI-IoT: Clustered and cohesive edge intelligence in internet of things," in *2022 IEEE International Conference on Edge Computing and Communications (EDGE)*. IEEE, 2022, pp. 33–40.

[5] S. Deng, H. Zhao, W. Fang, J. Yin, S. Dustdar, and A. Y. Zomaya, "Edge intelligence: The confluence of edge computing and artificial intelligence," *IEEE Internet of Things Journal*, vol. 7, no. 8, pp. 7457–7469, 2020.

[6] V. Casamayor Pujol, A. Morichetta, I. Murturi, P. Kumar Donta, and S. Dustdar, "Fundamental research challenges for distributed computing continuum systems," *Information*, vol. 14, no. 3, p. 198, 2023.

[7] P. K. Donta, S. N. Srirama, T. Amgoth, and C. S. R. Annavarapu, "Survey on recent advances in IoT application layer protocols and machine learning scope for research directions," *Digital Communications and Networks*, vol. 8, no. 5, pp. 727–744, 2022.

[8] O. Ali, M. K. Ishak, M. K. L. Bhatti, I. Khan, and K.-I. Kim, "A comprehensive review of internet of things: Technology stack, middlewares, and fog/edge computing interface," *Sensors*, vol. 22, no. 3, 2022.

[9] Y. Sharma, M. G. Khan, A. Al-Dulaimy, M. A. Khoshkholghi, and J. Taheri, "Networking models and protocols for/on edge computing," *Edge Computing: Models, Technologies and Applications*, vol. 33, p. 77, 2020.

[10] S. Al-Sarawi, M. Anbar, K. Alieyan, and M. Alzubaidi, "Internet of things (IoT) communication protocols," in *2017 8th International conference on information technology (ICIT)*. IEEE, 2017, pp. 685–690.

[11] D. N. Jha, K. Alwasel, A. Alshoshan, X. Huang, R. K. Naha, S. K. Battula, S. Garg, D. Puthal, P. James, A. Zomaya *et al.*, "IoTSim-Edge: a simulation framework for modeling the behavior of internet of things and edge computing environments," *Software: Practice and Experience*, vol. 50, no. 6, pp. 844–867, 2020.

[12] T. M. Tukade and R. Banakar, "Data transfer protocols in IoT—an overview," *Int. J. Pure Appl. Math*, vol. 118, no. 16, pp. 121–138, 2018.

[13] M. Seufert, S. Egger, M. Slanina, T. Zinner, T. Hoßfeld, and P. Tran-Gia, "A survey on quality of experience of HTTP adaptive streaming," *IEEE Communications Surveys & Tutorials*, vol. 17, no. 1, pp. 469–492, 2015.

[14] M. A. da Cruz, J. J. Rodrigues, P. Lorenz, P. Solic, J. Al-Muhtadi, and V. H. C. Albuquerque, "A proposal for bridging application layer protocols to HTTP on IoT solutions," *Future Generation Computer Systems*, vol. 97, pp. 145–152, 2019.

[15] F. Callegati, W. Cerroni, and M. Ramilli, "Man-in-the-middle attack to the HTTPS protocol," *IEEE Security & Privacy*, vol. 7, no. 1, pp. 78–81, 2009.

[16] D. Naylor, A. Finamore, I. Leontiadis, Y. Grunenberger, M. Mellia, M. Munafò, K. Papagiannaki, and P. Steenkiste, "The cost of the" s" in https," in *Proceedings of the 10th ACM International on Conference on emerging Networking Experiments and Technologies*, 2014, pp. 133–140.

[17] P. K. Donta, S. N. Srirama, T. Amgoth, and C. S. R. Annavarapu, "iCoCoA: intelligent congestion control algorithm for CoAP using deep reinforcement learning," *Journal of Ambient Intelligence and Humanized Computing*, vol. 14, no. 3, pp. 2951–2966, 2023.

[18] C. Bormann, A. P. Castellani, and Z. Shelby, "CoAP: An application protocol for billions of tiny internet nodes," *IEEE Internet Computing*, vol. 16, no. 2, pp. 62–67, 2012.

[19] A. Betzler, C. Gomez, I. Demirkol, and J. Paradells, "CoCoA+: An advanced congestion control mechanism for CoAP," *Ad Hoc Networks*, vol. 33, pp. 126–139, 2015.

[20] V. Rathod, N. Jeppu, S. Sastry, S. Singala, and M. P. Tahiliani, "CoCoA++: Delay gradient based congestion control for internet of things," *Future Generation Computer Systems*, vol. 100, pp. 1053–1072, 2019.

[21] C. Suwannapong and C. Khunboa, "Congestion control in CoAP observe group communication," *Sensors*, vol. 19, no. 15, p. 3433, 2019.

[22] A. K. Demir and F. Abut, "mlCoCoA: a machine learning-based congestion control for CoAP," *Turkish Journal of Electrical Engineering and Computer Sciences*, vol. 28, no. 5, pp. 2863–2882, 2020.

[23] D. Dinculeană and X. Cheng, "Vulnerabilities and limitations of MQTT protocol used between IoT devices," *Applied Sciences*, vol. 9, no. 5, p. 848, 2019.

[24] A. Stanford-Clark and H. L. Truong, "MQTT for sensor networks (MQTT-SN) protocol specification," *International business machines (IBM) Corporation version*, vol. 1, no. 2, pp. 1–28, 2013.

[25] J.-H. Park, H.-S. Kim, and W.-T. Kim, "DM-MQTT: An efficient MQTT based on SDN multicast for massive IoT communications," *Sensors*, vol. 18, no. 9, p. 3071, 2018.

[26] R. Bryce, T. Shaw, and G. Srivastava, "MQTT-G: A publish/subscribe protocol with geolocation," in *2018 41st international conference on telecommunications and signal processing (TSP)*. IEEE, 2018, pp. 1–4.

[27] J. L. Fernandes, I. C. Lopes, J. J. Rodrigues, and S. Ullah, "Performance evaluation of RESTful web services and AMQP protocol," in *2013 Fifth international conference on ubiquitous and future networks (ICUFN)*. IEEE, 2013, pp. 810–815.

[28] F. Iqbal, M. Gohar, H. Karamti, W. Karamti, S.-J. Koh, and J.-G. Choi, "Use of QUIC for AMQP in IoT networks," *Computer Networks*, vol. 225, p. 109640, 2023.

[29] G. Pardo-Castellote, "OMG data-distribution service: Architectural overview," in *23rd International Conference on Distributed Computing Systems Workshops, 2003. Proceedings.* IEEE, 2003, pp. 200–206.

[30] T. White, M. N. Johnstone, and M. Peacock, "An investigation into some security issues in the DDS messaging protocol," in *Australian Information Security Management Conference*, 2017, pp. 1–8.

[31] K. An, A. Gokhale, D. Schmidt, S. Tambe, P. Pazandak, and G. Pardo-Castellote, "Content-based filtering discovery protocol (CFDP) scalable and efficient OMG DDS discovery protocol," in *Proceedings of the 8th ACM International Conference on Distributed Event-Based Systems*, 2014, pp. 130–141.

[32] M. Ahamad, M. Raynal, and G. Thia-Kime, "An adaptive protocol for implementing causally consistent distributed services," in *Proceedings. 18th International Conference on Distributed Computing Systems (Cat. No. 98CB36183)*. IEEE, 1998, pp. 86–93.

[33] J. Sanchez-Monedero, J. Povedano-Molina, J. M. Lopez-Vega, and J. M. Lopez-Soler, "Bloom filter-based discovery protocol for DDS middleware," *Journal of Parallel and Distributed Computing*, vol. 71, no. 10, pp. 1305–1317, 2011.

[34] H. Otter, J. Martin, K. Bäsell, C. von Heymann, O. V. Hein, P. Böllert, P. Jänsch, I. Behnisch, K.-D. Wernecke, W. Konertz *et al.*, "Validity and reliability of the DDS for severity of delirium in the ICU," *Neurocritical care*, vol. 2, pp. 150–158, 2005.

[35] K. J. Friston, M. J. Ramstead, A. B. Kiefer, A. Tschantz, C. L. Buckley, M. Albarracin, R. J. Pitliya, C. Heins, B. Klein, B. Millidge *et al.*, "Designing ecosystems of intelligence from first principles," *arXiv preprint arXiv:2212.01354*, 2022.

[36] K. W. Church, "Word2vec," *Natural Language Engineering*, vol. 23, no. 1, pp. 155–162, 2017.

[37] X. He, K. Zhao, and X. Chu, "AutoML: A survey of the state-of-the-art," *Knowledge-Based Systems*, vol. 212, p. 106622, 2021.

[38] F. Conrad, M. Mälzer, M. Schwarzenberger, H. Wiemer, and S. Ihlenfeldt, "Benchmarking AutoML for regression tasks on small tabular data in materials design," *Scientific Reports*, vol. 12, no. 1, p. 19350, 2022.

[39] M. Wever, A. Tornede, F. Mohr, and E. Hüllermeier, "AutoML for multi-label classification: Overview and empirical evaluation," *IEEE transactions on pattern analysis and machine intelligence*, vol. 43, no. 9, pp. 3037–3054, 2021.

[40] P. K. Donta and S. Dustdar, "The promising role of representation learning for distributed computing continuum systems," in *2022 IEEE International Conference on Service-Oriented System Engineering (SOSE)*. IEEE, 2022, pp. 126–132.

[41] P. K. Donta, S. Boris, C. P. Victor, and D. Schahram, "Governance and sustainability of distributed continuum systems – A big data approach," *Journla of Bigdata*, vol. 1, no. 1, pp. 1–30, 2023.