Contents lists available at ScienceDirect

Digital Communications and Networks

journal homepage: www.keaipublishing.com/dcan

Survey on recent advances in IoT application layer protocols and machine learning scope for research directions

Praveen Kumar Donta [a,c], Satish Narayana Srirama [b,c,*], Tarachand Amgoth [a], Chandra Sekhara Rao Annavarapu [a]

[a] Department of Computer Science and Engineering, Indian Institute of Technology (Indian School of Mines), Dhanbad, 826004, India
[b] School of Computer and Information Sciences, University of Hyderabad, Hyderabad, 500046, India
[c] Mobile & Cloud Lab, Institute of Computer Science, Faculty of Science and Technology, University of Tartu, 50090, Estonia

ARTICLE INFO

Keywords:
Internet of things
Machine learning
Application layer
Request-response protocols
Publish-subscribe protocols

ABSTRACT

The Internet of Things (IoT) has been growing over the past few years due to its flexibility and ease of use in real-time applications. The IoT's foremost task is ensuring that there is proper communication among different types of applications and devices, and that the application layer protocols fulfill this necessity. However, as the number of applications grows, it is necessary to modify or enhance the application layer protocols according to specific IoT applications, allowing specific issues to be addressed, such as dynamic adaption to network conditions and interoperability. Recently, several IoT application layer protocols have been enhanced and modified according to application requirements. However, no existing survey articles focus on these protocols. In this article, we survey traditional and recent advances in IoT application layer protocols, as well as relevant real-time applications and their adapted application layer protocols for improving performance. As changing the nature of protocols for each application is unrealistic, machine learning offers means of making protocols intelligent and is able to adapt dynamically. In this context, we focus on providing open challenges to drive IoT application layer protocols in such a direction.

1. Introduction

In recent years, a large variety of devices have become interconnected over the Internet at an unexpected rate by using the Internet of Things (IoT). Increasing interest in the IoT has driven many areas (such as agriculture, hospitals, manufacturing, security surveillance [1–3], air and water quality, transport, and domotics) to adopt IoT features [4,5]. The primary goal of these applications is to either exchange data and control signals among devices or to use cloud-based communication protocols. In the layered architecture of the IoT, the application layer provides various communication protocols and acts as an interface between desired application and end-users [6–8]. As the number of IoT applications increases, so does the need to modify or introduce new protocols that address issues, such as dynamic adaption to the network conditions and interoperability. However, updating or providing new protocols for each and every application is a challenging task. Machine Learning (ML) can be used to make these protocols dynamic and intelligent.

ML extracts complex models through analysis of (or learning from) the experiences without being explicitly programmed [9–11]. It is used in several applications, such as Wireless Sensor Networks (WSNs) [12], medical data analysis [13,14], and social networks [15,16], and also helps to mitigate several issues in IoT, such as intrusion detection [17], improving wireless communication [18], and data analytics [19,20]. Applying ML to an IoT application layer not only produces intelligent protocols, but also reduces redesign overheads and limits human intervention [12]. For instance, data-sensitive real-time applications (e.g., medical, industrial, and smart city apps) frequently and quickly require data from IoT devices, but getting the most recent data quickly from edge nodes is difficult due to network delays and connection failures. To address these issues, ML is used extensively in the IoT for predicting values in situations where real-time data cannot be acquired [21]. Other advantages of using ML in IoT applications are summarized as follows:

- IoT applications require less human intervention and redesign in the dynamic environments.

* Corresponding author.
E-mail addresses: praveeniitism@gmail.com (P.K. Donta), satish.srirama@uohyd.ac.in (S.N. Srirama), tarachand@iitism.ac.in (T. Amgoth), acsrao@iitism.ac.in (C.S.R. Annavarapu).

https://doi.org/10.1016/j.dcan.2021.10.004
Received 3 April 2020; Received in revised form 26 September 2021; Accepted 8 October 2021
Available online 13 October 2021

- ML–based clustering algorithms are efficient and accurate [12], such as those used by message brokers (e.g., Apache Qpid, VerneMQ, and HornetMQ).
- ML used for data pre-processing and feature selection reduces traffic overheads and optimizes the energy consumption of IoT devices

However, ML does have certain limitations that require consideration when applied to IoT applications and their protocols. ML is non-deterministic, has few available datasets, and requires prediction validation. Over the past decade, there have been several surveys of the IoT, its applications, applicable technologies, architectures, IoT privacy, IoT security, and so on [4,5]. Articles [18,19] were written about ML approaches to, among other subjects, IoT applications and challenges as well as fifth-generation communication and security. In Refs. [6–8,22], the authors covered various IoT protocols supporting the application, network, and physical layers. *Al* et al. [6] included some traditional protocols, such as Constrained Application Protocol (CoAP), Hypertext Transfer Protocol (HTTP), Extensible Messaging and Presence Protocol (XMPP), Message Queue Telemetry Transport (MQTT), and Data Distribution Service (DDS). In Ref. [7], *Sethi* et al. briefly compared HTTP, CoAP, and MQTT protocols with other layer protocols. An editorial column [22] discussed the importance of the IoT application layer protocols. In Ref. [8], *Salman* et al. made a simple comparison between some application layer protocols and other layer protocols.

The CoAP and HTTP were compared in Ref. [23]. This article discussed the benefits of these protocols, but did not make any open challenges to enhance the research. A comparison of MQTT and CoAP in terms of error and delay prone links was performed in Ref. [24], which stated that MQTT performs better with the IoT. IoT protocols were studied for cognitive Machine-to-machine (M2M) communication by Ref. [25]. In that article, *Aijaz* et al. restricted their focus to application layer protocols, specifically CoAP. In Ref. [26], the authors studied various CoAP security issues while failing to consider other application layer protocols or provide significant open issues for future research [27]. summarized the benefits and features of various traditional IoT application layer protocols (such as HTTP, CoAP, XMPP, Advanced Message Queuing Protocol (AMQP), MQTT, and DDS). However, it did not list any limitations or future enhancements. The computational performances of CoAP, MQTT, and WebSocket were compared across various scenarios and network conditions by Ref. [28].

In [29], *Saritha* et al. compared various traditional application layer protocols. This article briefly discusses these protocols and has not provided sufficient limitations to enhance the research in this area. The interoperability issues of various application layer protocols such as HTTP, CoAP, MQTT and AMQP are covered in Ref. [30]. In Ref. [31], *Safaei* et al. discussed reliability side effects related to application layer protocols, stating that MQTT works well for IoT among other protocols. Similarly, empirical studies on MQTT and CoAP were performed in Refs. [32,33]. In Ref. [34], experimental evaluations were conducted on CoAP, HTTP, XMPP, WebSocket, MQTT, and AMQP using various conditions and network scenarios. A performance evaluation of IoT application layer protocols was performed in Ref. [35]. [36] discussed application layer coding for IoT along with its implementation aspects, benefits, and limitations. A performance analysis of the HTTP, CoAP, and MQTT was performed in Ref. [37] under various circumstances, proving that the MQTT was lightweight and faster. Table 1 compares these surveys on IoT application layer protocols.

In the past decade, several IoT application layer protocols have been introduced and modified according to rapid changes in the network properties of the IoT. Existing surveys have not covered the advancements made for traditional application layer protocols. This motivates us to study recent advancements in application layer protocols and relevant applications used by these protocols. These enhancements are covered in this article, along with their benefits, limitations, and further research scope. Existing surveys have also neglected to discuss the scope of ML in IoT application layer protocols, despite a broad scope existing for making IoT application layer protocols intelligent. The primary focus of this article is shown in Fig. 1. Considering the gaps in existing surveys, this article contributes:

- A study of previous and recent advancements in IoT application protocols (e.g., request-response method, publish–subscribe (PubSub) pattern).
- A survey of various IoT application layer protocols and the benefits of using them in real-time applications, including industrial IoT, smart cities and homes, video and security surveillance, healthcare, mobility management, and the Web of Things (WoT).
- Open challenges to make IoT application layer protocols intelligent and dynamic by employing ML approaches. These challenges highlight the major issues of application layer protocols, such as congestion control, message expiry, end-to-end delay, energy-efficiency, and resource management.

Most of the abbreviations we use throughout the article are either standard or defined when first used, and for the reader's convenience, we listed the most used acronyms in Table 2. The remaining sections of this article are arranged as follows. In Section 2, we provide a survey on previous and recent advancements in IoT application layer protocols and list their limitations. In Section 3, we list various IoT applications that use application layer protocols and their evaluations. In Section 4, ML's adaptability to request-response and PubSub for further research is discussed. Finally, in Section 5, we conclude our discussions.

2. Recent advances in IoT application layer protocols

The Constrained RESTful Environments (CoRE) group under the Internet Engineering Task Force (IETF) and International Telecommunication Union-Telecommunication works on application layer protocol development in the IoT. These protocols are mainly categorized into request–response (e.g., client/server), PubSub, push-pull, and exclusive-pair communication protocols. Of these, request–response and PubSub protocols have been well received in the literature. This section describes historical and recent advances of these protocols, along with their benefits and limitations. The taxonomy of conventional and current application layer protocols of IoT are shown in Fig. 2, and the comparisons are summarized in Table 3.

2.1. Request-response protocols

The request-response model is one of the basic stateless and bidirectional communication models on the Internet; it is also used in a constrained IoT. In this model, two or more end parties (e.g., clients/servers) exchange their data asynchronously. In the IoT, Representational State Transfer (REST)ful HTTP, XMPP, CoAP, and WebSocket use the request-response communication pattern. In this subsection, we discuss recent advancements in each of these request-response protocols.

2.1.1. RESTful HTTP

HTTP was initially developed by Tim Berners-Lee [39] and later enhanced jointly by IETF and World Wide Web (WWW) Consortium for web-based messaging. It commonly uses a Transmission Control Protocol (TCP) for reliable transmissions over the Internet [40]. This protocol is not recommended for the IoT, as it uses existing web standards instead of developing services or components for constrained IoT applications. Its limitations with respect to the IoT are highlighted as follows.

- *Scalability:* The IoT is composed of a large number of nodes, and each node that connects with a server requires an underlying persistent connection, as the limited capability of holding requests by an HTTP server precludes it from holding connection requests from all nodes. Besides, each node in the network connects with multiple nodes, creating heavy loads on constrained IoT devices. These heavy

Table 1
Comparison of recent and related surveys on IoT application layer protocols.

Reference	Year	CoAP	HTTP	XMPP	WebSocket	MQTT	MQTT-SN	AMQP	WS–N	STOMP	DDS	Message Queues	Others	Remarks
[23]	2012	✓	✓t	×	×	×	×	×	×	×	×	×	×	1. Covered CoAP and HTTP along with their benefits 2. No open issues discussed in this article
[24]	2014	✓	×	×	×	✓	×	×	×	×	×	×	×	Compared MQTT and CoAP in terms of error and delay prone links
[6]	2015	✓	✓	✓	×	✓	✓	×	×	×	✓	×	×	1. Discussed no open issues related to application layer protocols 2. Failed to cover the latest advances in the application layer protocols and message queues
[25]	2015	✓	×	×	×	×	×	×	×	×	×	×	×	Focused on non-application layer protocols
[26]	2015	✓	×	×	×	×	×	×	×	×	×	×	×	1. Covered CoAP security issues 2. Focused on no other application layer protocols
[27]	2016	✓	✓	✓	✓	✓	×	✓	×	×	✓	×	×	1. Covered traditional application layer protocols only 2. Discussed no recent advancements
[28]	2016	✓	×	×	✓	✓	×	×	×	×	×	×	×	Compared CoAP, MQTT and WebSocket protocols using various scenarios for performance computation
[7]	2017	✓	✓	×	×	✓	✓	×	×	×	×	×	×	1. Covered application layer protocols briefly 2. Provided no open challenges related to said protocols
[29]	2017	✓	✓	✓	✓	✓	✓	✓	×	✓	×	×	×	1. Described traditional protocol benefits 2. Provided no open issues or recent advancements of application layer protocols
[30]	2017	✓	✓	×	×	✓	×	✓	×	×	×	×	×	Described interoperability issues for application layer protocols
[31]	2017	✓	✓	×	×	✓	×	✓	×	×	✓	×	×	Discussed reliability side effects for application layer protocols
[32]	2017	✓	×	×	×	×	×	×	×	×	×	×	×	1. Studied MQTT and CoAP empirically 2. Failed to study other application layer protocols
[34]	2017	✓	✓	✓	✓	✓	×	✓	×	×	×	×	×	Performed an experimental evaluation of CoAP, HTTP, XMPP, WebSocket, MQTT, and AMQP using various conditions
[33]	2017	✓	×	×	✓	×	×	×	×	×	×	×	×	Compared the CoAP and MQTT using various IoT conditions
[22]	2018	×	✓	×	×	×	×	×	×	×	×	×	×	Covered basic information about communication protocols
[37]	2018	✓	✓	×	✓	×	×	×	×	×	×	×	×	

(continued on next page)

Table 1 (*continued*)

Reference	Year	CoAP	HTTP	XMPP	WebSocket	MQTT	MQTT-SN	AMQP	WS–N	STOMP	DDS	Message Queues	Others	Remarks
														Performed comparative study on HTTP, CoAP, and MQTT
[35]	2018	✓	×	✓	×	✓	×	✓	×	×	×	×	×	Evaluated performance of application layer protocols
[8]	2019	✓	×	✓	×	✓	×	×	×	×	✓	×	×	1. Covered application layer protocols briefly 2. Discussed no open challenges
[38]	2020	✓	✓	✓	×	✓	×	✓	✓	×	✓	×	×	Focuses on research efforts for smart farming using IoT application layer protocols
Our Survey	–	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	Discusses recent advances, relevant applications, and the scope of ML in IoT application layer protocols

resource constraints force HTTP servers to consume equivalent power, preventing them from providing scalability and energy-awareness with IoT applications.

- *Uni-directional:* Due to its large number of connections, the IoT needs to communicate with multiple devices simultaneously. By contract, HTTP protocol can only process a single request or response at a time. This uni-directional communication is not a recommended solution for IoT applications.
- *Responsiveness:* HTTP requires large bandwidth because of its weighty message sizes that increase latency and energy consumption. The synchronization feature of HTTP leads to latency during data transmission.

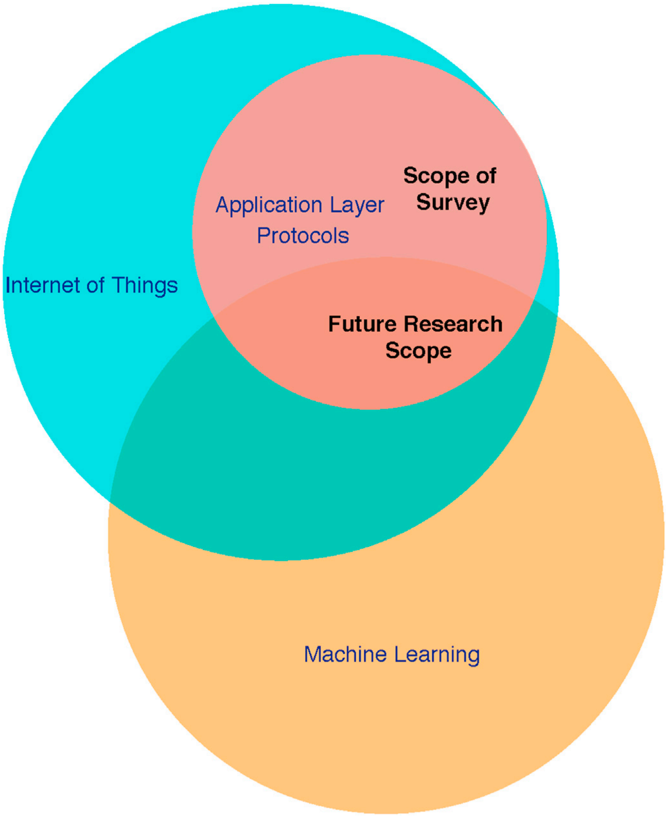

Fig. 1. Survey scope and future research directions.

Table 2
Most used acronyms.

Acronym	Abbreviation
ACK	Acknowledgement
AMQP	Advanced Message Queuing Protocol
BEB	Binary Exponential Backoff
CAIA	Centre for Advanced Internet Architectures
CBOR	Concise Binary Object Representation
CDG	CAIA Delay-Gradient
CoAP	Constrained Application Protocol
CoCoA	Congestion Control/Advanced
CoCO-RED	Congestion Control Random Early Detection
CoEP	Constrained Extensible Protocol
CoRE	Constrained RESTful Environments
DDS	Data Distribution Service
DTLS	Datagram Transport Layer Security
EAP	Extensible Authentication Protocol
FPB	Fibonacci Pre-increment Backoff
HTTP	Hypertext Transfer Protocol
ICA	Independent Component Analysis
IETF	Internet Engineering Task Force
IoT	Internet of Things
k-NN	*k*-Nearest Neighbors
M2M	Machine-to-Machine
MQTT	Message Queue Telemetry Transport
MSSQ	Microsoft Message Queue
OASIS	Organization for the Advancement of Structured Information Standards
PBF	Probabilistic Backoff Function
PCA	Principal Component Analysis
PDR	Packet Delivery Ratio
PubSub	Publish – Subscribe
RELOAD	Resource Location and Discovery
REST	Representational State Transfer
RFC	Request for Comments
RF	Random Forest
RL	Reinforcement Learning
RTO	Retransmission TimeOut
RTT	Retransmission Time
SOA	Service Oriented Architecture
STOMP	Simple/Streaming TextOriented Messaging Protocol
SVD	Singular Value Decomposition
SVM	Support Vector Machine
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
VBF	Variable Backoff Factor
WWW	World Wide Web
WSNs	Wireless Sensor Networks
WS–N	Web Services-Notification
XML	Extensible Markup Language
XMPP	Extensible Messaging and Presence Protocol
XoR	XMPP over RELOAD
XSF	XMPP Standards Foundation

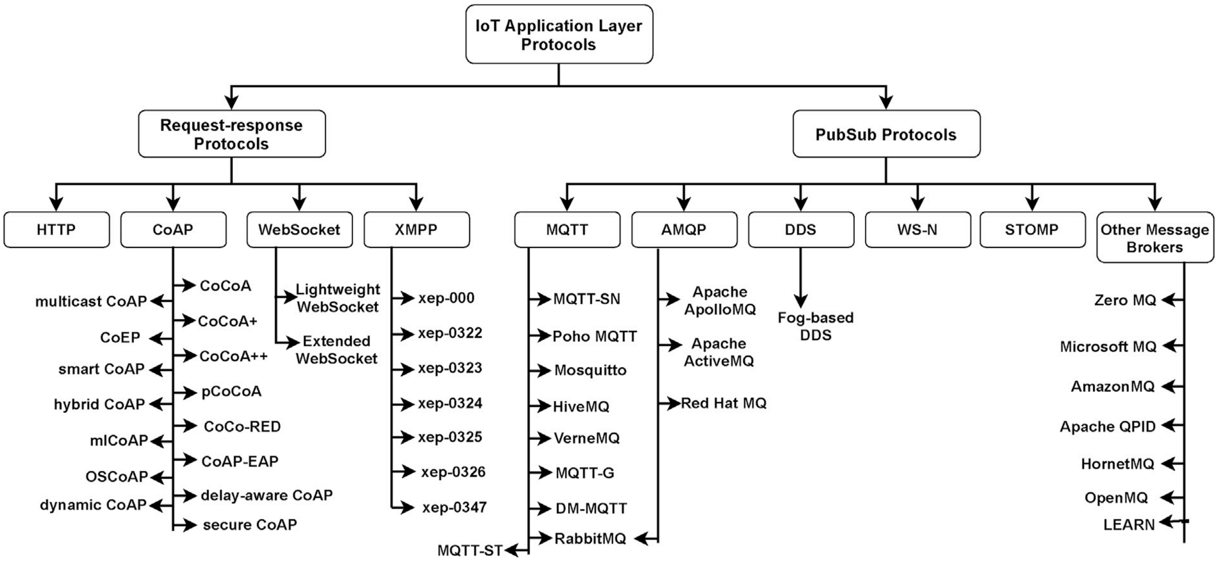

Fig. 2. Taxonomy of conventional and recent advances in IoT application layer protocols.

Table 3
Summary of traditional and recent advances in IoT application layer protocols.

Protocol/Broker	Transport	Reliability	Security	Flow Control	Clustering	QoS	Interoperability	Architecture
HTTP	TCP/UDP	Yes	Yes	Yes	No	Yes	No	Req/Res
CoAP	UDP	Yes	Yes	Yes	No	Yes	No	Req/Res
CoCoA/+/++	TCP/UDP	Yes	Yes	Yes	No	Yes	No	Req/Res
XMPP	TCP	Yes	No	Yes	No	No	No	Both
WebSocket	TCP	Yes	No	Yes	No	No	No	Both
MQTT	TCP	Yes	No	Yes	No	Yes	No	PubSub
MQTT-SN	UDP	Yes	No	No	No	Yes	No	PubSub
Mosquitto	TCP	Yes	No	No	No	Yes	Yes	PubSub
HiveMQ	TCP	Yes	No	Yes	No	Yes	Yes	PubSub
VerneMQ	TCP	Yes	Yes	Yes	Yes	Yes	Yes	PubSub
Paho MQTT	TCP	Yes	No	No	No	Yes	Yes	PubSub
AMQP	TCP	Yes	Yes	No	No	Yes	Yes	PubSub
Rabbit MQ	TCP	Yes	Yes	No	No	Yes	Yes	Both
ActiveMQ	TCP	Yes	Yes	Yes	Yes	Yes	No	PubSub
RedHat AMQ	TCP	Yes	Yes	Yes	Yes	Yes	No	PubSub
WS–N	–	Yes	Yes	No	No	No	No	PubSub
STOMP	TCP	Yes	No	Yes	No	No	Yes	PubSub
DDS	TCP/UDP	Yes	No	No	No	No	Yes	PubSub
ZeroMQ	TCP	No	Yes	No	No	Yes	Yes (only)	–
MicrosoftMQ	UDP	Yes	Yes	Yes	No	Yes	Yes	–
Amazon MQ	TCP	–	Yes	No	No	Yes	No	–
Apache QPID	TCP	–	No	Yes	Yes	Yes	No	–
HornetMQ	TCP/UDP	Yes	Yes	Yes	Yes	Yes	No	Both

- *Point-to-point Communication:* In the IoT, a large number of nodes collect and transmit data to base stations simultaneously. As stated previously, HTTP cannot handle multiple requests simultaneously, as it can only communicate with two point-to-point (one-to-one) devices (e.g., nodes, servers).
- *No Event-driven Nature:* Most IoT applications are event-driven, responding only when they identify an event. HTTP design is based on the request–response pattern, which is not event-driven and requires unnecessary energy consumption.

2.1.2. *Extensible messaging and presence protocol*

XMPP was developed by the IETF for the Internet, initially for heterogeneous networks [41] to avoid interoperability issues. It also provides efficient communication between devices while supporting both request–response and PubSub. These features are attractive to IoT users to implement various applications. Extended versions of the XMPP have been published through the XMPP Standards Foundation (XSF), and the recent versions are listed below.

- *xep-0000:* This version provides an efficient multicast data transmission between low-powered devices and uses the PubSub communication pattern. It also includes an interoperability interface over heterogeneous networks.
- *xep-0322, xep-0323, xep-0324:* The major extensions in these versions are about packet size optimization and memory issues. The xep-0323 version performs basic operations on sensor data, and xep-0324 incorporates the access privilege management approaches.
- *xep-0325, xep-0326:* These versions control servers, devices, and actuators in IoT infrastructures.
- *xep-0347:* This version defines localization and discovers needs for the deployment or removal of any device in a network.

The Lightweight Directory Access Protocol Request For Comments(RFC)8284 was introduced recently over XMPP using white page objects rather than Jabber IDs [42]. A Lightweight XMPP (LXMPP) was introduced in Ref. [43] for resource-constrained IoT. This protocol performs periodic data transmissions by using a sleeping mechanism. The

Table 4
Summary of recent advances in XMPP.

Protocol	Features
xep-0000	Multicast data transmissions, Interoperability
xep-0322	Compress XML files and fragments, decreasing packet size
xep-0323	Sensor data exchange over IoT
xep-0324	The management of access privileges and provisioning
xep-0325	Controlling actuators
xep-0326	Handling multiple Things
xep-0347	Location discovery
LADP [42]	Use of white pages instead of Jabber IDs
LXMPP [43]	Adopting duty-cycle mechanism
XoR [45]	Peer-to-Peer communication instead of routing
μXMPP [46]	Lightweight and low power to deploy in Sensor nodes

development of this protocol inherited the merits of *xep-0060* with the support of PubSub architecture [44]. In this approach, sleeping clients prolong battery lifespan. In Ref. [45], *Khramtsov* et al. proposed an XMPP over RELOAD (XoR) protocol based on XMPP. In it, XMPP clients establish peer-to-peer streams without routing to XMPP servers. Some other XMPP projects, such as mbed XMPPClient and μXMPP [46], use the LXMPP to allow the deployment of sensor nodes. The XSF team encouraged developers to modify or incorporate additional services in to XMPP. A summary of recent XMPP advances is presented in Table 4.

2.1.3. CoAP and its recent advances

CoAP (RFC-7252) is a low-powered, low-bandwidth, and lightweight constrained protocol for the IoT. This protocol was developed by the IETF CoRE group and inspired by HTTP over User Diagram Protocol (UDP). Due to lowbandwidth and higher network traffic, CoAP faces problems when handling the congestion and delay increases because of simple Binary Exponential Backoff (BEB) [47,48]. Congestion leads to network retransmission, increasing energy consumption, latency, and packet loss, while reducing throughput and the Packet Delivery Ratio (PDR) [49]. There are several recent CoAP advances and extensions summarized in Table 5. An end-to-end congestion control mechanism called Congestion Control/Advanced (CoCoA) was developed based on the CoAP [50]. The major enhancement from CoAP to CoCoA is the Retransmission Timeout (RTO). The CoAP uses a fixed RTO, whereas CoCoA uses a variable RTO based on the Retransmission Time (RTT) calculation used in TCP. In Ref. [51], *Betzler* et al. performed experimental testbeds on CoAP and CoCoA in various conditions with different nodes. These testbeds indicated that CoCoA produces a PDR 14–45% better than the CoAP. An extension of CoCoA, CoCoA+, introduces a Variable Backoff Factor (VBF) in place of the BEB. The VBF is calculated based on the initial RTO, avoiding the successive retransmissions over a short period [52]. CoCoA and CoCoA+ require weak and strong RTOs to determine their RTOs. Congestion Control Random Early Detection (CoCo-RED) was developed by Ref. [53] using a Fibonacci Pre-increment Backoff (FPB). In Ref. [54], *Akpakwu* et al. introduced a context-aware congestion control mechanism for CoAP, which computes RTO based on the weak and strong RTTs.

Further extensions for CoCoA+ with an optimized RTO were introduced in Precise CoCoA (pCoCoA) [55] and CoCoA++ [56]. The pCoCoA uses only one RTO (a smooth RTO) rather than maintaining two RTOs. It additionally uses the retransmission count at Acknowledgement (ACK) during the retransmission of the Confirmable (CON) message to regulate the number of packet retransmissions. It has less computational overhead compared with CoCoA+ or CoCoA. CoCoA++ maintains a single RTO, and computes a new RTO by integrating with the CAIA Delay-Gradient (CDG) and Probabilistic Backoff Function. CDG gets congestion information from TCP's congestion window. CoCoA++ replaces the VBF with a PBF during RTO computation, and it does not use per-packet RTT like other protocols. Overall, most CoAP advancements were made to prevent congestion by computing an optimal RTO. The general RTO estimation strategies of these protocols are illustrated in Fig. 3.

Table 5
Summary of recent advances in CoAP.

Protocol	Features	Advantages	Limitations
CoCoA/+ [52]	PBF and VBF introduced	Congestion control, packet loss minimization	End-to-end delay
CoCoA++ [56]	CAIA Delay-Gradient and PBF introduced	Congestion control, packet loss minimization	End-to-end delay and heavy computations
pCoCoA [55]	ACK maintained, single RTO used for new RTO computation	Congestion control, packet loss minimization	End-to-end delay
CoCo-RED [53]	FPB introduced	Congestion control, packet loss minimization	End-to-end delay
multicast CoAP [57]	Max-Age approach to maintain the cache	Congestion control, secure, energy-efficient, and delay-aware	Heavyweight
CoEP [58]	Lightweight security protocol	Authentication, confidentiality, and integrity	End-to-end delay
OSCoAP [59]	CMM-based security in the cross-layer	Security	Packet loss and high latency
secure CoAP [60]	Group key management and access control mechanisms	DTLS handshake mechanism	High latency
smart CoAP [61]	ML-based spoofing vulnerability attacks	Remote server access, fake message identification	High latency and packet loss
EDHOC [62]	Used Ephemeral Diffie-Hellman for security	End-to-end security	High packet loss
hybrid CoAP [63]	Switched dynamically between local and central executions	Low-cost, faster, scalable	Heavyweight
dynamic CoAP [64, 65]	Latency-aware data delivery using supervised classification methods	Minimum end-to-end delay	High computational
dynamic CoAP [66]	Multimedia streaming data transmissions	Traffic control	Heavyweight
delay-aware CoAP [67]	Delay-aware data communications in CoAP	Categorizes data deliveries according to demand	Handling the delay times with in the messages
CoAP-EAP [68]	Lightweight authentication mechanism	Supports authentication, flexibility, scalability, and identity federation	Packet loss
mlCoCoA [69]	Used ML to compute dynamic RTO times	Throughput improvement	Computationally heavy

In [70], *Ishaq* et al. proposed a CoAP group communication approach by allowing them to monitor resources. The parallel operation of group communication and observation in the CoAP is an incessant task, but it enables both operations to be performed while making the protocol intelligent at the data source. *Larmo* et al. [71] tested the performances of the CoAP and MQTT on a narrowband IoT. The data transmissions between the sensor nodes and the cloud were faster in the UDP-based CoAP when compared with the TCP-based MQTT in terms of system capacity, coverage, and latency, even with a massive network load. The MQTT performs well only under low network loads. The authors of [72] developed an efficient proxy for IoT to estimate congestion. Their approach also adopted the Max-Age approach to maintain the cache. CoAP cache management was also performed in Ref. [57] focusing on multicast CoAP to update cache information at the proxy. This approach also focused on efficient energy management and delay-aware data transmissions by preventing congestion. In Ref. [69], *Demir* et al.

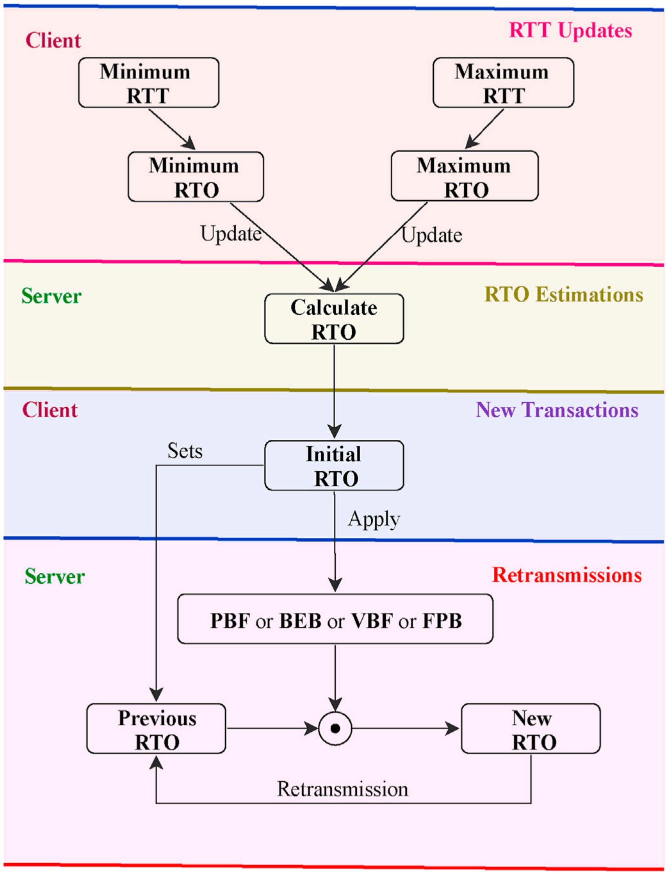

Fig. 3. General RTO estimation strategies for CoAP and its recent enhancements.

proposed an ML-based CoAP protocol with an RTO computation strategy that adopted ML-based technique.

Several articles [58–62,73–75] have focused on CoAP protocol security. A Constrained Extensible Protocol (CoEP) was introduced in Ref. [58] for a secure and lightweight IoT application layer protocol, embedding authentication, confidentiality, and integrity for efficient security-based data transmissions. A lightweight secure protocol using Datagram Transport Layer Security (DTLS) was is introduced in Ref. [73]. *Bhattacharyya* et al. [74] extended this approach by using DTLS with a pre-shared key to exchange encrypted data between IoT nodes. Similarly, a cross-layer approach of the Object Security of CoAP (OSCoAP) with a Cipher-block Chaining Message was proposed in Ref. [59] for media access control layer security in the IoT. The authors experimentally proved that this approach was more energy efficient (≈10%) and 37% faster than existing protocols. A patent from Ref. [76] enhanced the CoAP for group communication with selective responses in the IoT. In Ref. [61], *Roselin* et al. used ML to mitigate spoofing vulnerability attacks while supporting remote server access. This approach efficiently controlled fake requests from attackers to the remove servers. A security framework for CoAP was developed by Ref. [60] that included group key management and access control mechanisms as well as a pairwise symmetric key to avoid the DTLS handshake mechanism. Generic-bootstrapping architecture-based authentication and security mechanisms were introduced for CoAP in Ref. [75]. In Ref. [62], *Perez* et al. implemented a CoAP security mechanism using Ephemeral Diffie-Hellman Over COSE (EDHOC), an alternative to the DTLS handshake for an end-to-end security mechanism.

In [64–66], dynamic CoAP was introduced to control an IoT network. In Refs. [64,65], *Herrero* et al. focused on latency awareness to guarantee packet delivery using supervised classification on the data packets. This method also optimized power usage by minimizing network retransmissions. In Ref. [66], *Krawiec* et al. focused on multimedia streaming data transmission using the CoAP. This approach adjusted network parameters to improve traffic efficiency in the IoT. Hybrid CoAP was introduced in Ref. [63] that automatically switched between local devices and centralized server. This approach also detected and removed resources based on availability, making it scalable, cheaper, faster, and able to adapt to a dynamic system. The CoAP-Extensible Authentication Protocol (EAP) is a lightweight CoAP that was introduced as an authentication mechanism in Ref. [68], using bootstrapping services (such as the architecture) with entities, interfaces, and the flow operation. This protocol also supports authentication, flexibility, scalability, and identity federation.

2.1.4. WebSocket

WebSocket is a bidirectional, asynchronous, low-latency, full-duplex protocol developed for constant data transmission between two devices using a single TCP channel. This protocol was inspired by HTTP with advances that included event-driven communication in real-time IoT applications [77]. A WebSocket session can also start without using a request-response or PubSub communication pattern. Wong [78] developed a server system for WebSocket without using a master. Another WebSocket enhancement is communication clustering in master-slave servers. However, security is a major issue for this protocol. An extended WebSocket protocol was introduced in Ref. [79] that allowed control messages to be included in the frame without interruption, whereas the existing WebSocket protocol failed to get control messages.

2.1.5. Summary of request-response protocols

As pointed out earlier in Subsection 2.1, most CoAP advances have been introduced at the RTO computations, using static mathematical approaches or TCP retransmission strategies, and focusing on minimizing the number of retransmissions. These methods use only the RTT to estimate RTOs, yet RTTs are sometimes noisy [56]. These methods can control, but not prevent, congestion. The CoAP is embedded with the security features, but all the techniques discussed in the literature are heavyweight. XMPP, by contrast, is decentralized with redundant protocols that create excessive traffic, making it unsuitable for larger applications. XMPP is also unable to transmit unmodified binary data over the network.

2.2. Publish–subscribe model

The PubSub model is an asynchronous and loosely coupled model for data exchange between two devices that uses an eventbus or message broker. In this model, endpoints do not know about each other, but the broker knows about them. The broker makes a bridge between publishers and subscribers during data transmissions. The basic PubSub communication pattern is illustrated in Fig. 4. The most commonly used PubSub protocols (such as MQTT, AMQP, Web Services-Notification (WS-N), Simple/Streaming TextOriented Messaging Protocol (STOMP), and DDS) are described as follows.

2.2.1. MQTT and its recent advances

Andy Stanford-Clark and Arlen Nipper initially developed MQTT at IBM in 1999, and it was standardized in 2013 by the Organization for the Advancement of Structured Information Standards (OASIS). MQTT is a lightweight PubSub messaging protocol widely used by many web applications, including the IoT [80,81]. MQTT connection uses M2M communication with a many-to-many routing mechanism. MQTTv3.1 adopted the feature of message expiry (i.e., discarding unreceived messages after a set time period). Recently, several modified versions of MQTT clients, servers, and brokers have evolved; among these versions, MQTT-SN, Mosquitto, hiveMQ, verneMQ, and Paho-MQTT have been well received. Their benefits and limitations are summarized in Table 6. MQTT for Sensor Networks (MQTT-SN) protocol is specially designed for WSNs; currently, it is renamed as MQTT-SN [82]. The primary

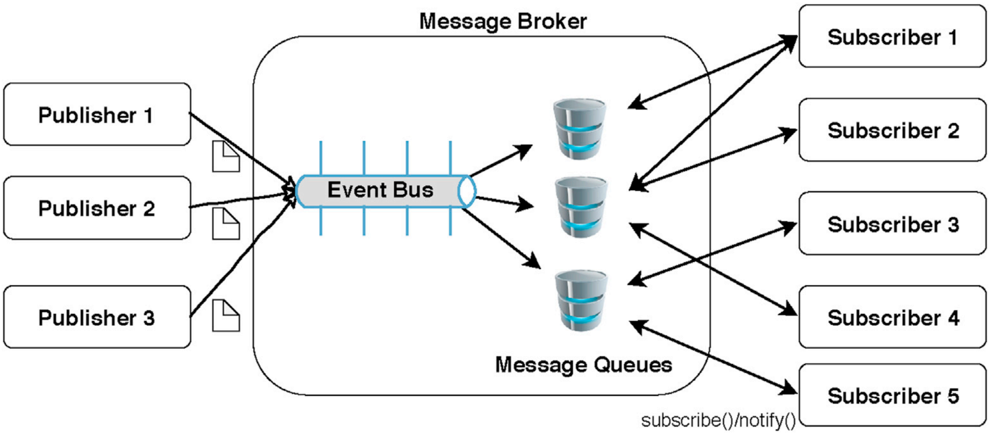

Fig. 4. Basic SubPub communication pattern.

Table 6
Summary of recent advances in MQTT.

Protocol	Features	Advantages	Limitations
MQTT-SN [82]	Designed for WSNs	Lightweight with low payload size	Works between sensor nodes or gateways
MQTT-G [86]	Search and rescue improvement	Packet loss reduction	Application specific, high delay
DM-MQTT [87]	Minimized data transmission delay in MQTT	Bidirectional communication and centralized broker	Heavy power consumption
MQTT-ST [88]	Enhanced MQTT with bidirectional communications	Quick response on failure messages and embedded message expiry	Heavyweight and high power consumer
lightweight MQTT [89]	Modified MQTT with limited features	Lightweight and quicker	No security
Paho MQTT [90]	Cost-effective and open source MQTT message broker	Reliable and interoperable	Not scalable
Mosquitto [91]	Lightweight message broker/server for MQTT	Maximizes bandwidth, reliable, and scalable	Limited message size, no cross-platform support
Modified Mosquitto [92]	Message priority embedded instead of FIFO	On-demand message delivery	High latency
RabbitMQ [93]	MQTT broker that uses Mosquitto	Scalable and reliable	Limited message size
HiveMQ [94]	MQTT message broker for M2M communication	Scalable and reliable	Large packet size
VerneMQ [95]	Distributed master-less clustering MQTT broker for reliable, highly scalable, and available	Low-latency and fault-tolerant	Lack of security

modifications performed in MQTT-SN prevent permanent connections through UDP and reduce payload sizes. This lightweight protocol consumes less power than MQTT. In Ref. [83], *Roy* et al. proposed gateway-to-gateway message transmissions using MQTT-SN. This approach performs communications efficiently between sensor nodes and gateway. Datasets used to evaluate the MQTT protocol with various ML algorithms are available in Refs. [84,85].

Geolocation-based MQTT (MQTT-G) was introduced in Ref. [86], an extension of the MQTT protocol. This protocol advertises the specific range of locations as a notification to clients and provides search and rescue improvements. However, this application can only be used for a particular application, not for all categories. *Kumar* et al. [96] integrated MQTT with quick UDP Internet connections to reduce connection overhead during the message exchange between IoT devices or other devices and servers. MQTT with TCP takes the additional burden to make the handshake during the transmission. This approach reduces latency up to 55%, while processor and memory usage are lowered by 80% and 50% respectively when compared to MQTT. The Spanning Tree-based MQTT (MQTT-ST) protocol was introduced in Ref. [88] with a bidirectional communication pattern. In addition to this feature, it also enhances functions that include message expiry enhancement, optimal route selection, routing path with minimal RTT, run-time message tracking, and early reaction on failure messages. However, when these functions are enhanced, the protocol becomes heavyweight and consumes additional power to perform them.

Park et al. [87] proposed a Direct Multicast-MQTT (DM-MQTT) protocol for efficient network resource utilization by preventing data transmission delay. This approach uses a bidirectional multicast mechanism between publishers and subscribers without using a centralized broker. This approach performs 58% and 65% better than MQTT in network usage and transmission delay, respectively. A lightweight MQTT was proposed in Ref. [89], using IEEE 1451 to modify MQTT's existing architecture and network features. This method is not a secured one, but it can be extended to allow for security. The Mosquitto is an open source lightweight message broker/server for low-powered IoT devices; it implements MQTT/MQTT-SN using C programming and is maintained by the Eclipse Foundation [91]. It does not have cross-platform support and does not categorize message priorities; instead, it follows a first-come first-served approach. The message priority feature included in modified Mosquitto [92], separates urgent and regular messages, assigning critical messages higher priority and serving them first.

RocketMQ is an MQTT-based message broker introduced for MQTT protocol in Ref. [93]. This broker controls the message push server with the help of Mosquitto by using the producer-consumer approach. HiveMQ is a client-based MQTT broker for reliable, efficient, and high-speed data transmission protocols on IoT devices. It is highly secured and provides continuous real-time data processing for brokers. This protocol works on MQTTv3.1 and all subsequent versions [94]. VerneMQ is a distributed, reliable, high-performance, master-less clustered messaging protocol developed in Erlang for MQTT brokers [95]. This protocol includes many features, such as flow-control mechanism, message expiration, and shared subscriptions. The Paho-MQTT is a cost-effective open source client-based MQTT messaging protocol for constrained M2M and IoT applications [90].

2.2.2. AMQP and its recent advances

The AMQP is an open source protocol initially designed for business transactions exchanging messages between two parties (i.e., point-to-point communication) [97,98]. The messaging structure of the AMQP is quite different in internal design and has more overhead compared to

MQTT. The AMQP maintains multiple queues, storing messages in them temporarily before subscribers receive them. Thus, it supports interoperability between brokers and clients, enabling communication between heterogeneous connected systems [99,100]. The AMQP is also suitable for applications that require safe, high-quality, reliable, and rapid message delivery. However, it is not completely suitable for constrained applications because of its substantial features.

RabbitMQ is a server initially implemented using Erlang programming [101]. It is an open source hybrid message broker for MQTT, AMQP, STOMP, and WebSocket. It sorts messages according to message properties, and ensures efficient and reliable delivery while managing component relationships well. However, this protocol is heavyweight, high-latency, and slow. It requires high computation, deployment, and maintenance costs. Apache ActiveMQ is an open source, asynchronous java-based multi-protocol message server for constrained applications [102–104]. It manages and allocates resources very efficiently, achieves high throughput, and also possesses interoperability. ActiveMQ also supports flow-control, message expiration, and message persistence by default. Its major limitations are memory limits per queue, and the heap is not used by default. An ActiveMQ broker is also limited by its architecture in terms of reliability, robustness, and scalability. Red Hat AMQ is an open source, lightweight, fast, and secure Java-based message protocol for large-scale Internet business applications, which was inspired by ActiveMQ. This protocol does not require any administrative costs, installation, or configurations. Apache Apollo is a new core for ActiveMQ that includes thousands of concurrent connections and a large multi-core server. Apache Apollo is a faster MQ that does not include all the features of ActiveMQ.

2.2.3. *Web Services-Notification*

The WS-N protocol is standardized by OASIS, and it exchanges messages in a coalition environment using predetermined notifications. It transmits packets simultaneously to all registered clients and supports interoperability well between middleware providers. This protocol also takes care of security during data transmissions, subscriptions, and notifications using various key exchange mechanisms. It is a loosely-coupled architecture that uses Extensible Markup Language (XML) and follows Service-oriented Architecture (SOA) principles. This protocol was initially developed for resource-constrained applications, but it is resource-heavy compared with other protocols due to SOA restrictions, XML, and multiple queues. Thus, it is not ideal for all types of IoT applications.

2.2.4. *STOMP*

The STOMP is a lightweight, secure, reliable text-based messaging protocol similar to HTTP that uses the PubSub pattern [105]. It has a straightforward implementation at the client level and also supports interoperability. Similar to AMQP, it uses frames for message exchanges between clients and servers by using a message broker. It does not use any comprehensive Application Programming Interfaces (APIs); instead, it uses simple, commonly used message operations with ACKs. The server implementation of STOMP is complicated, but it uses various existing server protocols (RabbitMQ, ActiveMQ, Apache Apollo, etc.).

2.2.5. *DDS*

DDS was developed by the Object Management Group for real-time, lightweight IoT applications and is available for both open source and commercial implementations. It uses the PubSub model for reliable and scalable multicast message exchanges, whereas the multicast improves the QoS for real-time IoT applications. DDS has a broker-less architecture, unlike MQTT or AMQP, and requires no additional memory for a message queue. DDS is inexpensive, smart, secure, fast, interoperability capable, and simplifies the deployment, integration, development, and management of IoT applications. Fog-based DDS architecture was used between fog devices and cloud centers in Ref. [106]. The fog-based DDS architecture classifies data by considering the minimum storage cost and latency. This model was specially designed for fog-cloud communication, so it is not useful for constrained IoT devices. In Ref. [107], the DDS protocol was used as middleware between software and hardware devices for microgrid applications. It is a centralized approach used to identify and clear faults optimally. The DDS protocol is currently used in real-time large-scale IoT applications, such as air-traffic control, smart grid, healthcare, robotics, industrial integration, military [108–110].

2.2.6. *Summary of PubSub protocols*

In the PubSub model, there are still some open challenges. For instance, there is no consistency check in the message queue or the brokers. As the model depends entirely on its brokers, a broker crash can render the whole system useless. Should a message crash while a client is handling it, the message cannot be recovered (e.g., Mosquitto allows the client to control the message). Most of these protocols use multiple message queues when providing transmission between publishers and subscribers, yet the queues work based on predetermined rules in static conditions, but not work in dynamic conditions. Message expiry time management is also challenging. PubSub model performance can be determined based on its constraints (such as message size, required queues, and clients accessing a queue simultaneously). However, determining these parameters on-the-fly is very difficult under conventional PubSub protocols. Finally, the MQTT protocol does not support multicast messaging and is also unsuitable for service discovery auto configurations [111].

2.3. *Other protocol/message brokers*

There have been several other recent advancements for message brokers. Some popular brokers are summarized in Table 7. *ZeroMQ (i.e., 0MQ, zMQ)* is a message exchange protocol that uses a broker-less PubSub pattern for concurrent or distributed applications. It provides low-latency, high-throughput performance that integrates components easily [112]. However, this protocol is unreliable because of the load it places on local control modules. ZeroMQ also fails to manage relationships between all network components. Microsoft Message Queue (MSSQ) is a message broker for reliable message exchange between applications in an enterprise [113,114]. In MSSQ, resource failure may happen when the message limit exceeds its threshold. There are also synchronization problems between the operations (e.g., copy, move, send, retrieve, and receive). Amazon MQ is a java-based durable message broker developed for ActiveMQ that also supports for MQTT, AMQP, STOMP, and WebSocket [115]. Apache Qpid functionalities are similar to

Table 7
Summary of message brokers for application layers.

Protocol	Features	Advantages	Limitations
Zero MQ [112]	Broker-less PubSub pattern for concurrent or distributed applications	Low-latency and high-throughput	Unreliable
Microsoft MQ [113, 114]	Reliable message exchange	Asynchronous data delivery	Limited number of message exchanges (4 MB)
Amazon MQ [115]	Java-based durable message broker for ActiveMQ	Provisioning, failure detection and recovery	Latency
Apache QPID	Reliable message exchange	Able to tolerate failures, and Interoperable	Latency
HornetQ	Asynchronous, clustered, middleware, and multi-protocol for ActiveMQ	Clustering mechanism over the message queues, grater stability	Data loss
Open MQ	Java-based open source message protocol	High availability, clustering for scalability	Latency

Table 8
Summary of Application Layer Protocols used in various IoT Applications.

	Article	Protocols	Advantages	Remarks
Industrial IoT	[121]	CoAP	Security, interoperability on-demand	Protocols cannot act as a middleware, only as a service oriented architecture
	[122]	CoAP	Improves throughput and reduces the mapping delay	Inter-compatibility of CoAP
	[123]	CoAP + CBOR	Minimizes message size and delay compared with HTTP and other web service protocols	Supports interoperability through uniform identification and interaction, and system integration
	[124]	MQTT	Inexpensive and low-latency	Estimate round trip latency between the cloud and data source
	[125]	MQTT	Smart agriculture through remote monitoring	Collect and process instantaneous data of agricultural field atmosphere
	[126]	WebSocket	Automatic keyword mining	Network traffic management and security inspection
	[127]	WebSocket	Full duplex communication for fast data transmissions	Monitoring temperature, soil humidity and controlling the irrigation sluice
	[128]	DDS	Reliable communication and middleware support	Design of fully partitioned deployment
	[129]	XMPP	delay-aware, interoperable, scalable and secure	XMPP communication model improves the management of reactive power requirement in microgrids
Smart City and Smart Home	[130]	CoAP	Minimize resource usage, network overhead, and time synchronization	CoAP performed data transmissions between sensor nodes and the gateway.
	[131]	MQTT, DDS, & CoAP	Tested latency and bandwidth	Comparison of application layer protocols on Smart city applications
	[132]	CoAP, MQTT, XMPP & WebSocket	Mean response time	Comparison of application layer protocols on Smart parking applications
	[133]	MQTT	Latency-aware	Event-based message communication
	[134]	MQTT	Remote control and latency-aware	Measure the voltage of AC and measure the usage of the current at power sockets
	[135]	MQTT	Reliable and cost effective	Cost-effective home automated demand response for energy efficiency
	[136]	MQTT	Efficient resource allocation, low-latency and communication failure detection	Allowed the common storage facility in the neighborhood and also shared energy between them
	[137]	MQTT	Low-latency, energy-efficiency, and cross platform support	Fog computing based home automation using ZiFi
Healthcare	[138]	MQTT	Security and privacy, scalability, data management, regulations, and interoperability	This system provided living assistance, early warning systems, e-medicine, and implants
	[139]	CoAP	Minimize handshake and data transmissions delay	Efficient authentication mechanism incorporated in CoAP
	[140]	MQTT	Low power and flow consumption, and low delay	High concurrency control between the server and mobile platform
	[141]	HTTP & MQTT	Reliability and data analysis, interoperability with minimal transmission delay	The HTTP and MQTT helped data transmissions between the server and web platforms, and significantly alleviated the cross-platform issue
	[142]	AMQP	Reliable communication, guarantee message delivery	AMQP improved the speed of the data transmissions between cloud and client. This approach also performed data analytics.
Mobility Management	[143, 144]	MQTT	Interoperable, redundant and fault tolerant	Reconfigurable connection management between sets of nodes
	[145]	MQTT & MQTT-SN	Reduce latency and improve the average PDR, and energy efficiency	Works on Internet of Drone Things
	[146]	MQTT	Latency-aware data transmissions	Works on connected car, maintains in vehicular communication
	[147]	CoAP	Minimize the latency and packet loss	The CoMP designed for mobility management for IoT
	[148]	CoAP, CoMP	Message communication in unreliable transmissions	Mobility management based on the CON message communication

(continued on next page)

Table 8 (*continued*)

	Article	Protocols	Advantages	Remarks
	[149]	CoAP, CoMP-G	Latency-aware, better in terms of total signalling	Group mobility communication management
	[150]	CoAP	Congestion management, transmission delay	Proxy mobile IPv6-based mobility management for the sensor nodes.
Video Surveillance	[151]	DDS	Congestion-aware, limited and time-varying bandwidth	Avoids visual errors or interruptions during the streaming and maintain high quality
	[152]	WebSocket & HTTP	Avoids latency, frame loss errors, and visual errors	WebSocket acted as a gateway and maintained the bidirectional communications
	[153]	WebSocket	Minimum data transmission delay	It avoided unnecessary data transmissions between the client and proxy server under the same channel
Web of Things	[154]	HTTP & CoAP	Efficient communication among devices and Interoperability	Proxy design for intercepts communications followed by mapping between CoAP and HTTP
	[155]	CoAP & MQTT	Optimized transmission delay, energy consumption, RTT and throughput	Controlling and monitoring of various sensors and actuators
	[156]	AMQP	Data transmissions over network failures	Clients disconnected from the server get help from the other subscribed clients
	[157]	CoAP	Minimal energy consumption, bandwidth utilization, and efficient data aggregation	A multiple observation requests for data collection and notifications at proxies
	[158]	CoAP	Flexible and scalable services	Extended CoAP for web-based applications without using any proxies, gateway or application servers
	[159]	CoAP	Security	Supports only unicast messages
	[160, 161]	CoAP	Detect high-level events from the data	Support entensive semantic matchmaking through non-standard inference services
	[162]	WebSocket	Robust communication between the web and physical devices	WebSocket used here to react on disconnected nodes and also help in session managements
	[163]	MQTT	Data communications in the IoT	Mobile and web applications are used to access the data from the MySQL database
	[111]	MQTT	Self-interaction with the IoT devices	Archive the auto-configuration mechanism with self discovery
	[164]	MQTT	Reliable message transmissions	Maintained the order between the work environment and messages
	[165]	MQTT	Security, Network optimization, reliable and flexible data communications	Optimized the distribution of the Wi-Fi network

RabbitMQ, and are used to implement the AMQP for reliable message exchanges among elements [116]. It easily detects client failover and assigns messages to different brokers. HornetQ is an asynchronous clustered open source middleware multi-protocol message project under the umbrella of ActiveMQ. Its most recent version (v 2.4) also supports AMQP and STOMP [117]. The revised version of the HornetQ is Apache Artemis 1.0.0 which provides better performance and stability by combining ActiveMQ. Open Message Queue (OpenMQ) is an open source java-based messaging protocol developed by Oracle with scalable, clustered, and loosely-coupled architecture [118,119]. Latency-aware Popular Resource Re-caching (LEARN) is a middleware message broker developed to recache and reallocate heavily-loaded resources in the IoT [120]. LEARN is useful for reducing the average delay between two brokers with optimized resource utilization.

Apache Qpid, VerneMQ, HornetQ use simple and static clustering methods over message queues in different servers. ML-based clustering algorithms (such as *k*-means, hierarchical, or fuzzy-c-means) may be used in these protocols to make the clustering dynamic, efficient, and comfortable.

3. Significance of IoT application layer protocols in use cases

Real-time applications benefit from using the application layer protocols of IoT in terms of latency, energy-efficiency, and throughput. This section reviews protocols used in specific applications, such as industrial IoT, smart cities and homes, healthcare, WoT, mobility management, and video surveillance. A summary of various applications that use application layer protocols are listed in Table 8.

3.1. Industrial IoT

Industrial IoT (IIoT) refers to IoT devices interconnected with manufacturing, automation, or control systems to perform data collection and analysis for improved machine efficiency and productivity [166]. The IIoT is a delay-sensitive application that allows data transmissions and analytics to be performed rapidly when a correct application layer protocol is selected. Here, we discuss a few real-time works that adopted various IoT application layer protocols in the IIoT, which are summarized in Table 8.

In [121], *Derhamy* et al. used CoAP protocols for on-demand, transparent, and secure IIoT translation to provide interoperability between communication protocols. In this work, CoAP acted as a SOA instead of middleware. This application also proved that the CoAP supports low latencies. A CoAP was used for a smart grid in Ref. [122] to improve throughput by reducing the mapping delay. The CoAP gateway was intercompatible in the smart grid network. In Ref. [123], CoAP was also used for a smart grid application through a combination with Concise Binary Object Representation (CBOR) to improve interoperability, system integration, and interaction. This combined protocol reduced message sizes and times compared with HTTP and other web service protocols. Latency estimation between data sources and the cloud (i.e., RTT and MQTT) was performed in Ref. [124]. This protocol provided inexpensive low latency. Smart agriculture through a remote monitoring station was introduced in Ref. [125] using the IoT. Initially, sensor nodes collected data and transmitted it to a remote monitoring system using the MQTT protocol. The station processed the data to provide farmers with decisions.

Automatic keyword mining was proposed in Ref. [126] for network load and security management in WebSocket. Initially, it identified frequently appearing keywords before using a hidden semi-Markov approach to establish relations among them. The experimental validations outperformed previous WebSocket. In Ref. [127], *Sunardi* et al. used WebSocket protocol for agriculture applications to collect and transfer data from the field. In this context, a sensor node collects the temperature, soil moisturizer, and control of the field's irrigation conduit. A fully partitioned DDS for real-time middleware systems was introduced in Ref. [128] for reliable data communication among the IoT devices. This approach performed well in terms of communication functions on a distributed network. In Ref. [129], XMPP was used to manage the microgrid's reactive power requirements. XMPP provided an abstract communication service with delay-aware, interoperable, scalable, and secure data transmissions.

3.2. Smart city and smart home

Smart cities and homes are rapidly growing IoT applications that enable smart technology with remote access and monitoring [167–169]. Growing these applications generates a vast amount of data, and this data management requires efficient application layer protocols. A summary of various IoT application layer protocols used in smart cities and smart homes is summarized in Table 8.

A novel time synchronization mechanism was proposed in Ref. [130] for smart home applications using CoAP. This approach achieved high accuracy due to CoAP usage between the sensor nodes and the gateway for reliable data transmissions. This method also minimized network overload and resource utilization. In Ref. [170], an energy-efficient, privacy-preserving, and secure data transmission protocol was introduced for smart home application. *Bansal* et al. [131] tested application layer protocols over a smart city application, comparing them with latency and bandwidth parameters. MQTT provided the best performance in a real-time environment. CoAP, MQTT, XMPP, and WebSocket performances were analyzed for a smart parking application in Ref. [132]. WebSocket and XMPP provided better scalability, while the others had single point of failure. XMPP also had the lowest server interaction. Using MQTT for event-based message communication in smart cities was proposed in Ref. [133]. This approach collected data using a network of IoT devices (e.g., Arduino, Raspberry Pi, and ESP8266) along with sensors and actuators. The communication system's primary goal was to prevent latency by minimizing the number of data transmissions.

Remote control based smart home automation was presented in Ref. [134] using MQTT. This mechanism measured the current used in a house at each socket and also monitored the AC power while permitting remote access to home appliances. Similarly, an user energy management system with automated demand response was presented in Ref. [135] for smart home applications. In this work, *Cornel* et al. used MQTT protocols for data transmission over devices. In Ref. [136], an autonomous resource allocation mechanism was proposed for smart cities. In this approach, the *Jamborsalamati* et al. used MQTT for efficient data transmission between sensor nodes, an energy exchange, and a typical storage facility between neighborhoods. The method monitored network communication failures while investigating latency. In Ref. [137], fog computing-based home automation techniques were provided using ZiWi [171]. This approach used MQTT protocols for low-latency data communication. In this approach, home power consumption was reduced by ZiWi.

3.3. Healthcare

In healthcare and biomedical systems, the IoT has significantly addressed the most challenging issues, saving lives at minimal cost [172]. Using the IoT in the healthcare system has several benefits, such as remote health monitoring, monitoring hardware availability and accessibility, minimizing emergency room wait times, tracking patients and staff, addressing chronic disease, and managing drug plans [173]. In this section, we present application layer protocols that benefited healthcare IoT applications.

A patient-centric eHealth system was proposed in Ref. [138] using a layered architecture including IoT devices, fog, and the cloud for handling complex data. This system covered assisted living, e-medicine, mobile health, early warning systems, and implants. In Ref. [139], an authentication mechanism was proposed to protect physiological data from malicious users. A smart gateway based on DTLS was used for efficient authentication authorization along with CoAP. Because of the enhanced DTLS, this algorithm efficiently performed data transmissions and minimized handshake delay. A mobility-based remote health monitoring system was introduced in Ref. [140]. Here, IoT devices were used to collect various patient details (e.g., pulse and ECG signals, body temperature, and body gesture). These details were transferred to the remote server using MQTT. This approach proved that MQTT performed better than HTTP and XMPP for this application.

In [141], an IoT-based ECG monitoring system was presented. IoT devices collected patient information and transferred it directly to the cloud using HTTP and MQTT protocols. In this approach, the data transmissions and analyses were performed efficiently without delays between the server and web platforms. A smart healthcare system was proposed in Ref. [142], where IoT devices collected patient information and used AMQP to transfer the data to the cloud for further analysis. After analysis, a physician received sufficient decisions.

3.4. Mobility management

The number of mobility based applications is rapidly increasing in the IoT, making tracking and managing these devices a challenging issue [174]. The mobility management highly depends on the wireless communication channel, and the application layer protocols also play a significant role in it [175]. The advantages and support of IoT application layer protocols for mobility management are discussed in this section.

An MQTT-based connection management system for vehicles was proposed in Refs. [143,144]. Here, the MQTT protocol plays a major role in performing efficient data transmissions between vehicles, where MQTT was able to function even if the vehicle network was reorganized. Similarly, *Mukherjee* et al. [145] used MQTT protocols in the Internet of Drone Things to improve message delivery speed by reducing latency. Further, they used MQTT-SN to improve the average packet delivery rate by approximately 30%, confirming an energy efficient approach. In Ref. [146], *Dhall* et al. proposed a connected car maintenance approach based on the IoT. The authors used MQTT for efficient data exchanges between cars to help owners to schedule service, analyze traffic, and receive vehicle crash data. Here, MQTT enabled faster data transmissions with efficient usage of low-latency bandwidth.

A CoAP-based Mobility Management Protocol (CoMP) was introduced in Ref. [147] for the IoT. The CoMP used a separate signaling flow and message format during data transmission. It kept track of sensor node IP addresses in the network, and efficiently handled latency and packet losses in the IoT. An extension of this work was completed in Ref. [148] with reliable mobility management. In this, *Chun* et al. modified reliable CON message communication from unreliable transmissions. Another extension of CoMP was achieved in Ref. [149], which performed group mobility communications called “CoMP-G for IoT”. This approach also performed better in terms of latency and total signaling during data transmission. In Ref. [150], proxy mobile IPv6-based mobility management approaches were proposed for sensors. These sensors used CoAP for data transmission. The classification of moving targets for WSNs was achieved using range limited marginalization and parallel range limited marginalization algorithms [176]. This approach achieved optimal decision fusion in various WSN scenarios.

3.5. Video surveillance

Monitoring and tracking activities using video surveillance generates a massive amount of data through camera sensors [177]. This data has to be communicated to servers using the application layer protocols. IoT application layer protocols were tested for the Internet of Video Things in Ref. [178] by using various QoS parameters (e.g., memory usage, bandwidth, energy consumption, throughput, latency, packet, and payload size). The protocols are needed to handle low bandwidth and high latency communication channels when transmitting data between applications and servers [179,180]. In Ref. [181], *Hilal* et al. proposed IoT acoustic surveillance using Linear Discriminate Analysis (LDA) and a Support Vector Machine (SVM). The primary goal of this approach was to recognize human screams or vocal stress, and it was tested at Waterloo International Airport.

DDS middleware-based real-time video streaming was analyzed in Ref. [151]. The performance of network video transmission was tested with various QoS features. DDS prevented visual errors or interruptions during streaming and maintained acceptable quality. A WebSocket-based video surveillance service architecture was developed in Ref. [152]. WebSocket acted as a gateway for maintaining bidirectional communication. Compared with HTTP, WebSocket outperformed in terms of latency, frame loss rate, and visual errors. In Ref. [153], WebSocket subprotocols were used for media data transmissions between clients and servers without sending proxy server requests from the current channel. Here, WebSocket minimized unnecessary data transmission delay between the proxy server and same channel clients.

3.6. Web of Things

The WoT is defined as physical devices (e.g., IoT devices) that can interact with the WWW, simply by applying web technologies to the IoT [182]. The main goals of the WoT are to hide the low-level designed, self-configured network through decreased human intervention, provide a platform for design and testing, and manage multiple platforms and protocols simultaneously and transparently [183]. IoT application layer protocols play a major role in the WoT, and most native WoT applications are summarized in Table 8.

A proxy design for an efficient intercept communication module was designed for a swarm IoT in Ref. [154]. This proxy performed mapping between HTTP messages and CoAP and vice versa. This mechanism provided efficient communication with low latency and IoT interoperability. MQTT and CoAP were tested for the WoT in Ref. [155] for tiny IoT applications. In this work, *Prabhu* et al. used an architecture to control and monitor sensor nodes and their storage. This method produced minimal transmission delay and power consumption while improving PDR and throughput. However, network failure is possible and can cause data transmissions between servers and clients to be lost. Transmitting data through other clients during network failure was proposed with AMQP in Ref. [156]. In this case, when a publisher disconnects from a client, newly subscribed clients get status updates without any delay. Multiple observation requests for data collection and notifications at proxies were proposed in Ref. [157] for CoAP. This approach minimized energy consumption and also utilized bandwidth efficiently.

CoAP is fully designed for IoT devices and not for web clients. Without intermediate application servers, a gateway, and proxies, CoAP is unable to serve the WoT appropriately. In this context, *Castro* et al. [158] extended CoAP for web-based applications without using any proxies, gateway, or application servers. A DTLS security-based algorithm was proposed in Ref. [159] for CoAP using a less biased algorithm for WoT. This approach supports multicast message distributions and performs encryption and decryption. A mobile agent was designed in Refs. [160,161] for a semantic WoT using CoAP. In this work, sensor nodes collected data from the field and extracted meaningful high-level events from it. This approach also supports interoperability between hybrid sensors. In Ref. [162], *Williams* et al. used WebSocket protocol between physical devices and web servers to push data transmissions. The WebSocket API maintaind the connection lifecycle, managed sessions, and reacted to disconnections. WebSocket was also useful as middleware in twisted environments.

The MQTT protocol was studied in Ref. [163] for efficient data communication and collection from the IoT environment. In this article, *Atmoko* et al. used the MySQL database to store collected data for further analysis. Further, mobile and web-based applications were provided to access data from remote locations. In Ref. [111], the authors designed an architecture that autoconfigured the MQTT protocol for auto interaction with IoT devices using a semantic Web. This approach successfully achieved the autoconfiguration of MQTT-based devices with self-discovery mechanisms. Reliable message transmission using MQTT was studied in Ref. [164] for maintaining ordering between a work environment and the messages. An order flag was used along with the messages to maintain proper synchronization of the situation and use the message flag to process requests. In Ref. [165], *Lie* et al. proposed a distributed Wi-Fi network optimization method for the IoT. In this optimized network, the MQTT protocol was used for reliable and flexible data transmissions between devices. This method also extended security mechanisms over the system through the MQTT protocol.

3.7. Other applications

A novel IoT honeypot (ThingPot) protocol was developed to provide denial-of-service attack security on the IoT [184]. It used the basic proof of the XMPP and REST API. Initially, it was developed for IoT applications, but was then extended to the IoT platform. Similarly [185], implemented a reverse engineering method on the message format of application layer protocols to identify security vulnerabilities in the IoT. This method identified change points and divided them into segments according to their static properties. These segments were then processed further to determine the vulnerabilities. In Ref. [186], *Da* et al. proposed MiddleBridge approach to act as middleware for translating CoAP, MQTT, DDS, XMPP, and WebSocket messages into HTTP. The MiddleBridge performed message configuration on the fly while addressing message size and transmission delays.

A classification model for intrusion detection systems was introduced in Ref. [187] to detect IoT system attacks that used the MQTT protocol. In this model, attacks were classified using deep learning and recurrent networks. In Ref. [188], *La* et al. enhanced the security features of the MQTT protocol. Subscribers could dynamically control access to the data and data streams over time. The authors of [189] implemented a CoAP accelerator as a hardware module in a field programmable gate array. This accelerator reduced latency between IoT devices and improved other QoS parameters, such as throughput and bandwidth.

4. Scope of machine learning for further research

While there have been several advances in IoT application layer protocols, as discussed in Section 2, several challenges remain open and require further research. Several survey articles have covered possible open problems for these protocols. Even though there is a broad scope of ML in this layer, authors have not focused on adopting ML features. This section fills this gap by providing possible open challenges for further research to make the protocols intelligent and dynamic using ML.

4.1. Congestion control

In CoAP, there is a need for dynamic RTO calculation to optimize the number of retransmissions and efficient RTTs. ML provides dynamic RTO computational strategies using Reinforcement Learning (RL), Bayesian, Regression, or SVM with minimal computational requirements [12]. From these, RL does not require a predetermined dataset and can learn during run-time. While Bayesian requires datasets, it also provides accurate RTO for congestion management in CoAP. Existing CoAP protocols use established RTTs to compute RTOs, yet those RTTs can be noisy. Other network features (e.g., retransmission counts, delays, and throughput) can also be considered when determining an optimal RTO. However, existing methods work once congestion is identified on a network, even though they cannot avoid the congestion completely. ML has the prediction capability to determine congestion before it occurs, based on the previous transmissions and other network conditions. An ML-based protocol can control unnecessary communications or retransmissions to avoid congestion from these predictions. It can also choose an alternate routing path to reduce traffic or drop the packets at a source to reduce energy consumption in extreme cases. In the IoT, the logistic regression, Random Forest (RF), *k*-Nearest Neighbors (*k*-NN) and Q-learning provide efficient predictions for avoiding congestion. Logistic regression was used to control data flow and mitigate IoT congestion [190]. However, using logistic regression at the protocol level can provide greater benefits such as rapid classification, good accuracy, easy to implement and efficient to train. The success of RF, *k*-NN, and Q-learning for congestion control in WSNs indicates that these algorithms can control the congestion in the IoT [12]. Among these methods, RF is useful only for applications that do not have memory constraints, as RF requires more memory space [191,192].

4.2. Energy-efficiency and delay-sensitive

Constrained IoT devices are equipped with a limited energy battery. The energy consumption of a device mainly depends on data transmissions. Increasing the number of transmissions also increases energy consumption and vice versa. In this context, we can perform dimensionality reduction on the data before it is transmitted to reduce transmission overheads and memory overheads, as well as congestion and other computational necessities. Transmission overheads minimize latency for delay-sensitive applications (such as IIoTs, intelligent transportation systems, and healthcare applications). Computation overheads also affect energy consumption. The Singular Value Decomposition (SVD), Principal Component Analysis (PCA), and Independent Component Analysis (ICA) are most suitable for dimensionality reduction in most IoT protocols [193,194]. Apart from dimensionality reduction, eliminating outlier data, anomalies, and digital garbage (garbage data generated by sensor nodes) can also help reduce data transmission overheads, saving energy and conserving network bandwidth [195–197]. The *k*-NN, *k*-means, SVM, and density–based spatial clustering of applications with noise algorithms fulfill this necessity in IoT application protocols.

Dimensionality reduction, outlier or anomaly detection, and edge data prediction can be incorporated into CoAP, WebSocket, XMPP, AMQP, MQTT, and all their extended versions. While XMPP is particularly unsuited to large-scale IoT application due to its high message redundancy, its performance can be improved significantly through dimensionality reduction. ML is also useful for checking data redundancy and preventing multiple retransmissions. The feature selection process also reduces heavy data transmissions in the network, providing beneficial support for delay-sensitive applications [198,199]. The feature selection mechanism can also use for device classification [200]. PCA, ICA, and SVD are more accessible ML techniques for performing the feature extraction, and these are also useful in IoT application layer protocols. Among these techniques, ICA requires more computation, making PCA and SVD preferable for constrained devices. PCA was used in the IoT for anomaly detection and dimensionality reduction in Refs. [201,202]. Adopting these features in protocols provides benefit during the data collection process. Similarly, ICA also helps when addressing various issues discussed in subsection 4.2 [203–205]. RL also fulfills these task without requiring any training data set.

4.3. Message expiry

Message expiry indicates that messages sent by a publisher or broker to the message queue will be discarded after a while if there is no subscriber response. In PubSub protocols, this is a fundamental parameter that describes message queue management quality. This option is included in recent versions of PubSub protocols with a static message expiry. Calculating optimal and dynamic message expiry is essential because setting it too high will result in queues keeping unnecessary messages while an excessively low setting can cause messages to be discarded before a subscriber can retrieve them. A dynamic message expiry time can be decided at run-time by using ML approaches [206]. RL approaches do not require prerecorded datasets for system training, allowing adaptability to message transmission situations and queue availability when determining optimal message expiry times. Bayesian and decision tree approaches can provide an accurate message expiry that depends on the queue availability and channel occupancy.

4.4. Resource management

AMQP uses multiple message queues, and each queue handles certain messages as per predefined constraints, making it static in nature. Additionally, AMQP does not support priority queues, which may be alleviated by large message expiry times (though this can cause critical information to become stuck in a queue). Message queue classification can be made dynamic in AMQP by using ML algorithms. In addition to having a priority queue, dynamic message queue management is an essential issue in AMQP. While adding additional features into AMQP with ML may increase the load a bit, rapid delivery can be achieved with higher quality and reliability. RabbitMQ suffers from redundant message broker communication, which can be avoided with dimensionality reduction mechanisms. This protocol can also enjoy enhanced packet loss estimation with ML techniques. Packet loss estimation can be predicted using the RL approaches on-the-fly, with no training [207].

Similarly, for PubSub models, deciding message size, the required number of message queues, and the number of clients able to access a queue simultaneously for a specific application can be very complex without ML. Depending on traffic conditions, ML can determine a dynamic message size or the simultaneous access count for the message queues or brokers. Increasing or decreasing the number of message queues dynamically on a network is not logically possible. However, in heterogeneous networks, ML can resize queues according to various application traffic conditions. ML algorithms will efficiently perform these operations. Bayesian in particular can accurately adjust message queue sizes on-the-fly through training.

5. Conclusions

In this survey, we have presented recent advances in application layer protocols for the IoT, followed by its significance in real-time use cases and ML-related research directions. Recently, several application layer protocols have been published as modified versions of conventional protocols that have not been covered by existing surveys. In this article, we have studied the enhancements and improvements of conventional application layer protocols. We have also identified and summarized the benefits and limitations of these protocols. In addition, we have discussed various message queues and message brokers. The significance of request-response and PubSub protocols in use cases (such as IIoT, smart cities and homes, healthcare, mobility management, video surveillance, and the WoT) have been discussed. We have also highlighted benefits achieved by the use cases through the application layer protocols. However, traditional and improved application layer protocols have not yet satisfied IoT needs due to variations in the dynamic condition of the applications. ML can make these protocols intelligent and work dynamically according to application conditions and without human intervention. This article also extended the usage of ML for future research to solve issues such as congestion, energy awareness, delay sensitivity, message expiration, and resource management.

Declaration of competing interest

There are no potential conflicts of interest.

Acknowledgements

The authors would like to thank DST (SERB), Government of India for grant No. EEQ/2018/000888. The work was also supported by the Archimedes Foundation under the Dora plus Grant 11-15/OO/11476. We also acknowledge financial support to UoH-IoE by MHRD (F11/9/2019-U3(A)).

References

[1] Domenico Ciuonzo, Giacinto Gelli, Antonio Pescapé, Francesco Verde, Decision fusion rules in ambient backscatter wireless sensor networks, in: IEEE 30th Annual International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC), vol. 2019, IEEE, 2019, pp. 1–6.

[2] Ruixin Niu, Pramod K. Varshney, Performance analysis of distributed detection in a random sensor field, IEEE Trans. Signal Process. 56 (1) (2007) 339–349.

[3] Domenico Ciuonzo, Salvo Rossi, P. Dechade, Detecting slight changes with hard decisions in wireless sensor networks, Int. J. Gen. Syst. 47 (5) (2018) 535–548.

[4] J. Lin, W. Yu, N. Zhang, X. Yang, H. Zhang, W. Zhao, A survey on internet of things: architecture, enabling technologies, security and privacy, and applications, IEEE Internet Things J. 4 (5) (2017) 1125–1142.

[5] S. Li, L. Da Xu, S. Zhao, The internet of things: a survey, Inf. Syst. Front 17 (2) (2015) 243–259.

[6] A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, M. Ayyash, Internet of things: a survey on enabling technologies, protocols, and applications, IEEE Commun. Survey. Tutor. 17 (4) (2015) 2347–2376.

[7] P. Sethi, S.R. Sarangi, Internet of things: architectures, protocols, and applications, J. Elect. Comput. Eng. 2017 (2017).

[8] T. Salman, R. Jain, A Survey of Protocols and Standards for Internet of Things, 2019 arXiv preprint arXiv:190311549.

[9] S. Marsland, Machine Learning: an Algorithmic Perspective, Chapman and Hall/CRC, 2014.

[10] D. Michie, D.J. Spiegelhalter, C. Taylor, et al., Machine learning, Neural Stat. Classif. 13 (1994) (1994) 1–298.

[11] E. Alpaydin, Introduction to Machine Learning, MIT press, 2014.

[12] D. Praveen Kumar, A. Tarachand, A.C.S. Rao, Machine learning algorithms for wireless sensor networks: a survey, Inf. Fusion 49 (2019) 1–25.

[13] H. Alinejad-Rokny, E. Sadroddiny, V. Scaria, Machine learning and data mining techniques for medical complex data analysis, Neurocomputing 276 (2018) 1.

[14] M. Chen, Y. Hao, K. Hwang, L. Wang, L. Wang, Disease prediction by machine learning over big data from healthcare communities, Ieee Access 5 (2017) 8869–8879.

[15] F. Shan, J. Liu, X. Wang, W. Liu, B. Zhou, A smart access control method for online social networks based on support vector machine, IEEE Access 8 (2020) 11096–11103.

[16] M. Keyvanpour, Z.K. Zandian, M. Heidarypanah, OMLML: a helpful opinion mining method based on lexicon and machine learning in social networks, Soc. Netw. Anal. Mining 10 (1) (2020) 1–17.

[17] K.A. da Costa, J.P. Papa, C.O. Lisboa, R. Munoz, V.H.C. de Albuquerque, Internet of Things: a survey on machine learning-based intrusion detection approaches, Comput. Network. 151 (2019) 147–157.

[18] J. Jagannath, N. Polosky, A. Jagannath, F. Restuccia, T. Melodia, Machine Learning for Wireless Communications in the Internet of Things: a Comprehensive Survey, Ad Hoc Networks, 2019, p. 101913.

[19] M.S. Mahdavinejad, M. Rezvan, M. Barekatain, P. Adibi, P. Barnaghi, A.P. Sheth, Machine learning for Internet of Things data analysis: a survey, Digit. Commun. Netw. 4 (3) (2018) 161–175.

[20] Erwin Adi, Adnan Anwar, Zubair Baig, Sherali Zeadally, Machine learning and data analytics for the IoT, Neural Comput. Appl. 32 (2020) 16205–16233.

[21] V. Özdemir, N. Hekim, Birth of industry 5.0: making sense of big data with artificial intelligence,“the internet of things” and next-generation technology policy, OMICS A J. Integr. Biol. 22 (1) (2018) 65–76.

[22] Y.B. Zikria, H. Yu, M.K. Afzal, M.H. Rehmani, O. Hahm, Internet of Things (IoT): Operating System, Applications and Protocols Design, and Validation Techniques, Future Generation Computer Systems, 2018, pp. 699–706.

[23] M.R. Palattella, N. Accettura, X. Vilajosana, T. Watteyne, L.A. Grieco, G. Boggia, et al., Standardized protocol stack for the internet of (important) things, IEEE Commun. Survey. Tutor. 15 (3) (2012) 1389–1406.

[24] M. Collina, M. Bartolucci, A. Vanelli-Coralli, G.E. Corazza, Internet of Things application layer protocol analysis over error and delay prone links, in: 7th Advanced Satellite Multimedia Systems Conference and the 13th Signal Processing for Space Communications Workshop, vol. 2014, ASMS/SPSC). IEEE, 2014, pp. 398–404.

[25] A. Aijaz, A.H. Aghvami, Cognitive machine-to-machine communications for Internet-of-Things: a protocol stack perspective, IEEE Internet Things J. 2 (2) (2015) 103–112.

[26] J. Granjal, E. Monteiro, J.S. Silva, Security for the internet of things: a survey of existing protocols and open research issues, IEEE Commun. Survey. Tutor. 17 (3) (2015) 1294–1312.

[27] M.B. Yassein, M.Q. Shatnawi, et al., Application layer protocols for the Internet of Things: a survey, in: International Conference on Engineering & MIS (ICEMIS), vol. 2016, IEEE, 2016, pp. 1–4.

[28] S. Mijovic, E. Shehu, C. Buratti, Comparing application layer protocols for the Internet of Things via experimentation, in: IEEE 2nd International Forum on Research and Technologies for Society and Industry Leveraging a Better Tomorrow (RTSI), vol. 2016, IEEE, 2016, pp. 1–5.

[29] S. Saritha, V. Sarasvathi, A study on application layer protocols used in IoT, in: International Conference on Circuits, Controls, and Communications (CCUBE), vol. 2017, IEEE, 2017, pp. 155–159.

[30] V.M. Tayur, R. Suchithra, Review of interoperability approaches in application layer of Internet of Things, in: International Conference on Innovative Mechanisms for Industry Applications (ICIMIA), vol. 2017, IEEE, 2017, pp. 322–326.

[31] B. Safaei, A.M.H. Monazzah, M.B. Bafroei, A. Ejlali, Reliability side-effects in Internet of Things application layer protocols, in: 2017 2nd International Conference on System Reliability and Safety (ICSRS), IEEE, 2017, pp. 207–212.

[32] U. Tandale, B. Momin, D.P. Seetharam, An empirical study of application layer protocols for IoT, in: International Conference on Energy, Communication, Data Analytics and Soft Computing (ICECDS), vol. 2017, IEEE, 2017, pp. 2447–2451.

[33] I. Hedi, I. Speh, A. Sarabok, IoT network protocols comparison for the purpose of IoT constrained networks, in: 40th International Convention on Information and Communication Technology, vol. 2017, Electronics and Microelectronics (MIPRO), 2017, pp. 501–505.

[34] L. Năstase, I.E. Sandu, N. Popescu, An experimental evaluation of application layer protocols for the internet of things, Stud. Inf. Control 26 (4) (2017) 403–412.

[35] M. Pohl, J. Kubela, S. Bosse, K. Turowski, Performance evaluation of application layer protocols for the internet-of-things, in: Sixth International Conference on Enterprise Systems (ES), vol. 2018, IEEE, 2018, pp. 180–187.

[36] M. Sandell, U. Raza, Application layer coding for IoT: benefits, limitations, and implementation aspects, IEEE Syst. J. 13 (1) (2018) 554–561.

[37] H. Chaudhary, N. Vaishnav, B. Tank, Comparative analysis of application layer internet of things (IoT) protocols, in: Information and Communication Technology for Sustainable Development, Springer, 2018, pp. 173–180.

[38] Dimitrios Glaroudis, Athanasios Iossifides, Periklis Chatzimisios, Survey, comparison and research challenges of IoT application protocols for smart farming, Comput. Network. 168 (2020) 107037.

[39] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, et al., Hypertext Transfer protocol–HTTP/1.1. RFC 2616, june, 1999.

[40] N. Naik, Choice of effective messaging protocols for IoT systems: MQTT, CoAP, AMQP and HTTP, in: IEEE International Systems Engineering Symposium (ISSE), vol. 2017, 2017, pp. 1–7.

[41] Yun Li, Hui Ma, Lei Wang, Shiwen Mao, Guoyin Wang, Optimized content caching and user association for edge computing in densely deployed heterogeneous networks, IEEE Trans. Mobile Comput. (2020).

[42] S. Kille, Lightweight Directory Access Protocol (LDAP) Schema for Supporting the Extensible Messaging and Presence Protocol (XMPP), 2017. RFC 8284, https://toolsietforg/html/rfc8284 (Accessed 10 January 2020).

[43] H. Wang, D. Xiong, P. Wang, Y. Liu, A lightweight XMPP publish/subscribe scheme for resource-constrained IoT devices, IEEE Access 5 (2017) 16393–16405.

[44] P. Millard, P. Saint-Andre, R. Meijer, XEP-0060: Publish-Subscribe, vol. 1, XMPP Standards Foundation, 2010, p. 13.

[45] E. Khramtsov, XMPP over RELOAD (XOR), XMPP Standards Foundation, 2019.

[46] A. Hornsby, E. Bail, μXMPP: lightweight implementation for low power operating system Contiki, in: International Conference on Ultra Modern Telecommunications & Workshops, vol. 2009, IEEE, 2009, pp. 1–5.

[47] C. Bormann, A.P. Castellani, Z. Shelby, CoAP: an application protocol for billions of tiny internet nodes, IEEE Internet Comput. 16 (2) (2012) 62–67.

[48] C. Bormann, S. Lemay, H. Tschofenig, K. Hartke, B. Silverajan, B. Raymor, CoAP (Constrained Application Protocol) over TCP, TLS, and WebSockets. Internet Requests for Comments, RFC Editor, RFC, 2018, p. 8323.

[49] P.K. Donta, T. Amgoth, C.S.R. Annavarapu, Congestion-aware data acquisition with Q-learning for wireless sensor networks, in: IEEE International IOT, Electronics and Mechatronics Conference (IEMTRONICS), vol. 2020, IEEE, 2020, pp. 1–6.

[50] A. Betzler, C. Gomez, I. Demirkol, J. Paradells, CoAP congestion control for the internet of things, IEEE Commun. Mag. 54 (7) (2016) 154–160.

[51] A. Betzler, J. Isern, C. Gomez, I. Demirkol, J. Paradells, Experimental evaluation of congestion control for CoAP communications without end-to-end reliability, Ad Hoc Netw. 52 (2016) 183–194.

[52] A. Betzler, C. Gomez, I. Demirkol, J. Paradells, CoCoA+: an advanced congestion control mechanism for CoAP, Ad Hoc Netw. 33 (2015) 126–139.

[53] C. Suwannapong, C. Khunboa, Congestion control in CoAP observe group communication, Sensors 19 (15) (2019) 3433.
[54] G.A. Akpakwu, G.P. Hancke, A.M. Abu-Mahfouz, CACC: context-aware congestion control approach for lightweight CoAP/UDP-based Internet of Things traffic, in: Transactions on Emerging Telecommunications Technologies, 2019, p. e3822.
[55] S. Bolettieri, G. Tanganelli, C. Vallati, E. Mingozzi, pCoCoA: a precise congestion control algorithm for CoAP, Ad Hoc Netw. 80 (2018) 116–129.
[56] V. Rathod, N. Jeppu, S. Sastry, S. Singala, M.P. Tahiliani, CoCoA++: Delay Gradient Based Congestion Control for Internet of Things, Future Generation Computer Systems, 2019.
[57] J. Mišić, V.B. Mišić, Proxy cache maintenance using multicasting in CoAP IoT domains, IEEE Internet Things J. 5 (3) (2018) 1967–1976.
[58] M. Manini, J. Esquiagola, L. Costa, M. Zuffo, CoEP: a secure & lightweight application protocol for the Internet of Things, in: IEEE XXV International Conference on Electronics, Electrical Engineering and Computing (INTERCON), vol. 2018, IEEE, 2018, pp. 1–4.
[59] R.H. Randhawa, A. Hameed, A.N. Mian, Energy efficient cross-layer approach for object security of CoAP for IoT devices, Ad Hoc Netw. 92 (2019) 101761.
[60] C.S. Park, Security architecture for secure multicast CoAP applications, IEEE Internet Things J. 7 (4) (2020) 3441–3452.
[61] A.G. Roselin, P. Nanda, S. Nepal, X. He, J. Wright, Exploiting the remote server access support of CoAP protocol, IEEE Internet Things J. 6 (6) (2019) 9338–9349.
[62] S. Pérez, D. Garcia-Carrillo, R. Marín-López, J.L. Hernández-Ramos, R. Marín-Pérez, A.F. Skarmeta, Architecture of security association establishment based on bootstrapping technologies for enabling secure IoT infrastructures, Future Generat. Comput. Syst. 95 (2019) 570–585.
[63] B. Djamaa, A. Yachir, M. Richardson, Hybrid CoAP-based resource discovery for the internet of things, J. Ambient Intell. Human. Comput. 8 (3) (2017) 357–372.
[64] R. Herrero, Dynamic CoAP mode control in real time wireless IoT networks, IEEE Internet Things J. 6 (1) (2018) 801–807.
[65] R. Herrero, Supervised classification for dynamic CoAP mode selection in real time wireless IoT networks, Telecommun. Syst. (2020) 1–12.
[66] P. Krawiec, M. Sosnowski, J.M. Batalla, C.X. Mavromoustakis, G. Mastorakis, DASCo: dynamic adaptive streaming over CoAP, Multimed. Tool. Appl. 77 (4) (2018) 4641–4660.
[67] Yanyan Han, Dale Seed, Chonggang Wang, Xu Li, Quang Ly, Zhuo Chen, Delay-aware application protocol for internet of things, IEEE Network 33 (1) (2018) 120–127.
[68] D. Garcia-Carrillo, R. Marin-Lopez, Lightweight CoAP-based bootstrapping service for the internet of things, Sensors 16 (3) (2016) 358.
[69] Alper Kamil Demir, Fatih Abut, mlCoCoA: a machine learning-based congestion control for CoAP, Turk. J. Electr. Eng. Comput. Sci. 28 (5) (2020).
[70] I. Ishaq, J. Hoebeke, I. Moerman, P. Demeester, Observing CoAP groups efficiently, Ad Hoc Netw. 37 (2016) 368–388.
[71] A. Larmo, A. Ratilainen, J. Saarinen, Impact of CoAP and MQTT on NB-IoT system performance, Sensors 19 (1) (2019) 7.
[72] J. Mišić, M.Z. Ali, V.B. Mišić, Architecture for IoT domain with CoAP observe feature, IEEE Internet Things J. 5 (2) (2018) 1196–1205.
[73] S. Raza, H. Shafagh, K. Hewage, R. Hummen, T. Voigt, Lithe: lightweight secure CoAP for the internet of things, IEEE Sensor. J. 13 (10) (2013) 3711–3720.
[74] A. Bhattacharyya, T. Bose, S. Bandyopadhyay, A. Ukil, A. Pal, LESS: lightweight establishment of secure session: a cross-layer approach using CoAP and DTLS-PSK channel encryption, in: IEEE 29th International Conference on Advanced Information Networking and Applications Workshops, vol. 2015, IEEE, 2015, pp. 682–687.
[75] A. Khushu, D. Zgonjanin, N. Kim, Generic bootstrapping architecture (GBA) based security over constrained application protocol (CoAP) for IoT devices, in: Google Patents 15/661, 2019 Jan 31, p. 857. US Patent App.
[76] C. Wang, R. Di Girolamo, S.A. Rahman, X. Li, Z. Chen, Q. Ly, et al., Enhanced CoAP group communications with selective responses, in: Google Patents 15/752, 2019 Jan 10, p. 459. US Patent App.
[77] I. Fette, A. Melnikov, The websocket protocol, URL, https://toolsietf.org/html/rfc6455, 2016 (Accessed 10 January 2020).
[78] Y.T. Wong, Masterless websocket server system, in: Google Patents 14/815, 2017 Feb 2, p. 882. US Patent App.
[79] J.R. Fallows, S.R. Atkinson, Extending WebSocket protocol. Google patents, US Patent 9 (2016) 331–890.
[80] M.B. Yassein, M.Q. Shatnawi, S. Aljwarneh, R. Al-Hatmi, Internet of Things: survey and open issues of MQTT protocol, in: International Conference on Engineering & MIS (ICEMIS), vol. 2017, IEEE, 2017, pp. 1–6.
[81] A. Stanford-Clark, A. Nipper, MQTT, 2017.
[82] Andy Stanford-Clark, Hong Linh Truong, MQTT for sensor networks (MQTT-SN) protocol specification, in: International business machines (IBM) Corporation version 1, 2013, 2.
[83] D.G. Roy, B. Mahato, D. De, R. Buyya, Application-aware end-to-end delay and message loss estimation in Internet of Things (IoT)—MQTT-SN protocols, Future Generat. Comput. Syst. 89 (2018) 300–316.
[84] Ivan Vaccari, Giovanni Chiola, Maurizio Aiello, Maurizio Mongelli, Enrico Cambiaso, MQTTset, a new dataset for machine learning techniques on MQTT, Sensors 20 (22) (2020) 6578.
[85] Hanan Hindy, Ethan Bayne, Miroslav Bures, Robert Atkinson, Christos Tachtatzis, Xavier Bellekens, Machine Learning Based IoT Intrusion Detection System: an MQTT Case Study (MQTT-IoT-Ids2020 Dataset), 2020 arXiv preprint arXiv: 200615340.
[86] R. Bryce, T. Shaw, G. Srivastava, MQTT-G: a publish/subscribe protocol with geolocation, in: 41st International Conference on Telecommunications and Signal Processing, vol. 2018, IEEE, 2018, pp. 1–4.
[87] J.H. Park, H.S. Kim, W.T. Kim, DM-MQTT: An efficient MQTT based on SDN multicast for massive IoT communications, Sensors 18 (9) (2018) 3071.
[88] E. Longo, A.E.C. Redondi, M. Cesana, A. Arcia-Moret, P. Manzoni, MQTT-ST: a Spanning Tree Protocol for Distributed MQTT Brokers, 2019 arXiv preprint arXiv: 191107622.
[89] J. Velez, R. Trafford, M. Pierce, B. Thomson, E. Jastrzebski, B. Lau, IEEE 1451-1-6: providing common network services over MQTT, in: IEEE Sensors Applications Symposium (SAS), vol. 2018, IEEE, 2018, pp. 1–6.
[90] E. Paho-MQTT, MQTT-SN Software. Accessed: Mar, 2018.
[91] R. Light, Mosquitto: server and client implementation of the MQTT protocol, J. Open Source Softw. 2 (13) (2017) 265.
[92] K. Hwang, J.M. Lee, D.H. Lee, Modification of Mosquitto broker for delivery of urgent MQTT message, in: IEEE Eurasia Conference on IOT, Communication and Engineering (ECICE), vol. 2019, IEEE, 2019, pp. 166–167.
[93] M. Yue, Y. Ruiyang, S. Jianwei, Y. Kaifeng, A MQTT protocol message push server based on RocketMQ, in: 10th International Conference on Intelligent Computation Technology and Automation (ICICTA), vol. 2017, IEEE, 2017, pp. 295–298.
[94] M. HiveMQ Enterprise, Broker. MQTT Essentials Part 2, Publish & Subscribe, 2016.
[95] A. VerneMQ, VerneMQ-Messaging broker for MQTT, URL: https://vernemqcom/intro/indexhtml. 2020 (Accessed 10 January 2020).
[96] P. Kumar, B. Dezfouli, Implementation and analysis of QUIC for MQTT, Comput. Network. 150 (2019) 28–45.
[97] S. Vinoski, Advanced message queuing protocol, IEEE Internet Comput. 10 (6) (2006) 87–89.
[98] F. Gutierrez, AMQP with spring boot, in: Spring Boot Messaging, Springer, 2017, pp. 59–80.
[99] Yun Li, Shichao Xia, Qianying Yang, Guoyin Wang, Weiyi Zhang, Lifetime-priority-driven resource allocation for WNV-based internet of things, IEEE Internet Things J. 8 (6) (2021) 4514–4525.
[100] Yun Li, Yunjin Liang, Qilie Liu, Wang, Honggang. Resources allocation in multicell D2D communications for internet of things, IEEE Internet Things J. 5 (5) (2018) 4100–4108.
[101] A. RabbitMQ, RabbitMQ-Messaging that just works, URL: https://wwwrabbitmqcom, 2020 (Accessed 10 January 2020).
[102] ActiveMQ, Accessed: https://activemq.apache.org/components/classic/, 2020 (Accessed 10 January 2020).
[103] B. Christudas, ActiveMQ, in: Practical Microservices Architectural Patterns, Springer, 2019, pp. 861–867.
[104] A. ActiveMQ, ActiveMQ: the Apache Software Foundation, 2016. Retrieved.
[105] F.T. Johnsen, Using publish/subscribe for short-lived iot data, in: Federated Conference on Computer Science and Information Systems (FedCSIS), vol. 2018, IEEE, 2018, pp. 645–649.
[106] F. Karatas, I. Korpeoglu, Fog-based data distribution service (F-DAD) for internet of things (IoT) applications, Future Generat. Comput. Syst. 93 (2019) 156–169.
[107] H. Habib, N.F.K. Habib, M.M. Esfahani, O.A. Mohammed, S. Brahma, An Enhancement of Protection Strategy for Distribution Network Using the Communication Protocols, IEEE Transactions on Industry Applications, 2020.
[108] Z. Meng, Z. Wu, C. Muvianto, J. Gray, A data-oriented M2M messaging mechanism for industrial IoT applications, IEEE Internet Things J. 4 (1) (2016) 236–246.
[109] R. White, G. Caiazza, C. Jiang, X. Ou, Z. Yang, A. Cortesi, et al., Network reconnaissance and vulnerability excavation of secure DDS systems, in: IEEE European Symposium on Security and Privacy Workshops (EuroS&PW), vol. 2019, IEEE, 2019, pp. 57–66.
[110] T. White, M.N. Johnstone, M. Peacock, An Investigation into Some Security Issues in the DDS Messaging Protocol, 2018.
[111] G. Kim, S. Kang, J. Park, K. Chung, An MQTT-based context-aware autonomous system in oneM2M architecture, IEEE Internet Things J. 6 (5) (2019) 8519–8528.
[112] P. Fengping, C. Jianzheng, Distributed system based on ZeroMQ, Electron. Test 7 (7) (2012) 24–29.
[113] S. Horrell, Microsoft Message Queue (MSMQ), Enterprise Middleware, 1999, pp. 25–35.
[114] A. Redkar, K. Rabold, R. Costall, S. Boyd, C. Walzer, Pro MSMQ: Microsoft Message Queue Programming, Apress, 2004.
[115] AmazonMQ, Accessed, https://tutorialsdojo.com/aws-cheat-sheet-amazon-mq/, 2021 (Accessed 10 January 2020).
[116] A. Qpid, QPID: an Open Source AMQP Messaging, vol. 2013, AMQP, 2018.
[117] G.H.L. Sihai, Asynchronous message transfer with HornetQ, Software Guide 1 (12) (2010) 16.
[118] A.F. Klein, M. Ştefănescu, A. Saied, K. Swakhoven, An experimental comparison of ActiveMQ and OpenMQ brokers in asynchronous cloud environment, in: Fifth International Conference on Digital Information Processing and Communications (ICDIPC), vol. 2015, IEEE, 2015, pp. 24–30.
[119] C. Vințe, O. Solutions, Upon a trading system Architecture based on OpenMQ middleware, Open Source Sci. J. 1 (1) (2009).
[120] X. Sun, N. Ansari, Traffic load balancing among brokers at the IoT application layer, IEEE Trans. Netw. Serv. Manag. 15 (1) (2017) 489–502.
[121] H. Derhamy, J. Eliasson, J. Delsing, IoT interoperability—on-demand and low latency transparent multiprotocol translator, IEEE Internet Things J. 4 (5) (2017) 1754–1763.
[122] I.J. Shin, B.K. Song, D.S. Eom, International Electronical Committee (IEC) 61850 mapping with constrained application protocol (CoAP) in smart grids based

European telecommunications standard institute Machine-to-Machine (M2M) environment, Energies 10 (3) (2017) 393.

[123] M. Iglesias-Urkia, D. Casado-Mansilla, S. Mayer, J. Bilbao, A. Urbieta, Integrating electrical substations within the IoT using IEC 61850, CoAP, and CBOR, IEEE Internet Things J. 6 (5) (2019) 7437–7449.

[124] P. Ferrari, E. Sisinni, D. Brandão, M. Rocha, Evaluation of communication latency in industrial IoT applications, in: IEEE International Workshop on Measurement and Networking (M&N), vol. 2017, IEEE, 2017, pp. 1–6.

[125] S.V. Mukherji, R. Sinha, S. Basak, S.P. Kar, Smart agriculture using internet of things and MQTT protocol, in: International Conference on Machine Learning, Big Data, Cloud and Parallel Computing (COMITCon), vol. 2019, IEEE, 2019, pp. 14–16.

[126] B.C. Li, S.Z. Yu, Keyword mining for private protocols tunneled over websocket, IEEE Commun. Lett. 20 (7) (2016) 1337–1340.

[127] S. Sunardi, A. Afif, F. Noviyanto, Real time monitoring and irrigation control using the websocket protocol, in: Proceedings of the 1st International Conference on Science and Technology for an Internet of Things, European Alliance for Innovation (EAI), 2018, pp. 1–11.

[128] M. García-Valls, J. Domínguez-Poblete, I.E. Touahria, C. Lu, Integration of data distribution service and distributed partitioned systems, J. Syst. Architect. 83 (2018) 23–31.

[129] S.S. Hussain, M.A. Aftab, I. Ali, IEC 61850 modeling of DSTATCOM and XMPP communication for reactive power management in microgrids, IEEE Syst. J. 12 (4) (2018) 3215–3225.

[130] S.C. Son, N.W. Kim, B.T. Lee, C.H. Cho, J.W. Chong, A time synchronization technique for CoAP-based home automation systems, IEEE Trans. Consum. Electron. 62 (1) (2016) 10–16.

[131] S. Bansal, D. Kumar, IoT application layer protocols: performance analysis and significance in smart city, in: 10th International Conference on Computing, Communication and Networking Technologies (ICCCNT), vol. 2019, IEEE, 2019, pp. 1–6.

[132] P. Kayal, H. Perros, A comparison of IoT application layer protocols through a smart parking implementation, in: 20th Conference on Innovations in Clouds, Internet and Networks (ICIN), vol. 2017, IEEE, 2017, pp. 331–336.

[133] S. Jaloudi, MQTT for IoT-based applications in smart cities, Palestinian J. Technol. Appl. Sci. (PJTAS) 2 (2019).

[134] A. Cornel-Cristian, T. Gabriel, M. Arhip-Calin, A. Zamfirescu, Smart home automation with MQTT, in: 54th International Universities Power Engineering Conference (UPEC), vol. 2019, IEEE, 2019, pp. 1–5.

[135] K. Jia, J. Xiao, S. Fan, G. He, A MQTT/MQTT-SN-based user energy management system for automated residential demand response: formal verification and cyber-physical performance evaluation, Appl. Sci. 8 (7) (2018) 1035.

[136] P. Jamborsalamati, E. Fernandez, M. Moghimi, M.J. Hossain, A. Heidari, J. Lu, MQTT-based resource allocation of smart buildings for grid demand reduction considering unreliable communication links, IEEE Syst. J. 13 (3) (2018) 3304–3315.

[137] I. Froiz-Míguez, T.M. Fernández-Caramés, P. Fraga-Lamas, L. Castedo, Design, implementation and practical evaluation of an IoT home automation system for fog computing applications based on MQTT and ZigBee-WiFi sensor nodes, Sensors 18 (8) (2018) 2660.

[138] B. Farahani, F. Firouzi, V. Chang, M. Badaroglu, N. Constant, K. Mankodiya, Towards fog-driven IoT eHealth: promises and challenges of IoT in medicine and healthcare, Future Generat. Comput. Syst. 78 (2018) 659–676.

[139] P.M. Kumar, U.D. Gandhi, Enhanced DTLS with CoAP-based authentication scheme for the internet of things in healthcare application, J. Supercomput. (2017) 1–21.

[140] D. Yi, F. Binwen, K. Xiaoming, M. Qianqian, Design and implementation of mobile health monitoring system based on MQTT protocol, in: IEEE Advanced Information Management, Communicates, Electronic and Automation Control Conference (IMCEC), vol. 2016, IEEE, 2016, pp. 1679–1682.

[141] Z. Yang, Q. Zhou, L. Lei, K. Zheng, W. Xiang, An IoT-cloud based wearable ECG monitoring system for smart healthcare, J. Med. Syst. 40 (12) (2016) 286.

[142] C. Krishna, T. Sasikala, Healthcare monitoring system based on IoT using AMQP protocol, in: International Conference on Computer Networks and Communication Technologies, Springer, 2019, pp. 305–319.

[143] A. Schmitt, F. Carlier, V. Renault, Dynamic bridge generation for IoT data exchange via the MQTT protocol, Procedia Comput. Sci. 130 (2018) 90–97.

[144] A. Schmitt, F. Carlier, V. Renault, Data exchange with the MQTT protocol: dynamic bridge approach, in: IEEE 89th Vehicular Technology Conference (VTC2019-Spring), vol. 2019, IEEE, 2019, pp. 1–5.

[145] A. Mukherjee, N. Dey, D. De, EdgeDrone: QoS aware MQTT middleware for mobile edge computing in opportunistic internet of drone things, Comput. Commun. 152 (2020) 93–108.

[146] R. Dhall, V. Solanki, An IoT based predictive connected car maintenance, Int. J. Interact. Multimed. Artif. Intell. 4 (3) (2017).

[147] S. Chun, J. Park, Mobile CoAP for IoT mobility management, in: 12th Annual IEEE Consumer Communications and Networking Conference, vol. 2015, CCNC), 2015, pp. 283–289.

[148] S.M. Chun, J.T. Park, A mechanism for reliable mobility management for internet of things using CoAP, Sensors 17 (1) (2017) 136.

[149] M. Gohar, J.G. Choi, S.J. Koh, CoAP-based group mobility management protocol for the Internet-of-Things in WBAN environment, Future Generat. Comput. Syst. 88 (2018) 309–318.

[150] S. Choi, S. Koh, Use of proxy mobile IPv6 for mobility management in CoAP-based internet-of-things networks, IEEE Commun. Lett. 20 (11) (2016) 2284–2287.

[151] B. Almadani, M. Alsaeedi, A. Al-Roubaiey, QoS-aware scalable video streaming using data distribution service, Multimed. Tool. Appl. 75 (10) (2016) 5841–5870.

[152] G.D. Mandyam, Transferring Media Data Using a Websocket Subprotocol. Google Patents, 2016 Nov 17, p. 538. US Patent App. 15/146.

[153] G. D'Angelo, S. Rampone, A NAT traversal mechanism for cloud video surveillance applications using WebSocket, Multimed. Tool. Appl. 77 (19) (2018) 25861–25888.

[154] J. Esquiagola, L. Costa, P. Calcina, M. Zuffo, Enabling CoAP into the swarm: a transparent interception CoAP-HTTP proxy for the internet of things, in: Global Internet of Things Summit (GIoTS), vol. 2017, IEEE, 2017, pp. 1–6.

[155] P. Prabhu Kumar, G. Geetha, Web-cloud architecture levels and optimized MQTT and COAP protocol suites for web of things, Concurrency Comput. Pract. Ex. 31 (12) (2019), e4867.

[156] P. Bhimani, G. Panchal, Message delivery guarantee and status update of clients based on IOT-AMQP, in: Intelligent Communication and Computational Technologies, Springer, 2018, pp. 15–22.

[157] N. Correia, D. Sacramento, G. Schütz, Dynamic aggregation and scheduling in CoAP/observe-based wireless sensor networks, IEEE Internet Things J. 3 (6) (2016) 923–936.

[158] M. Castro, A.J. Jara, A.F. Skarmeta, Enabling end-to-end CoAP-based communications for the web of things, J. Netw. Comput. Appl. 59 (2016) 230–236.

[159] P. Singhal, P. Sharma, B. Hazela, End-to-end message authentication using CoAP over IoT, in: International Conference on Innovative Computing and Communications, Springer, 2019, pp. 279–288.

[160] M. Ruta, F. Scioscia, A. Pinto, F. Gramegna, S. Ieva, G. Loseto, et al., A CoAP-based framework for collaborative sensing in the semantic web of things, Procedia Comput. Sci. 109 (2017) 1047–1052.

[161] M. Ruta, F. Scioscia, A. Pinto, F. Gramegna, S. Ieva, G. Loseto, et al., CoAP-based collaborative sensor networks in the semantic web of things, J. Ambient Intell. Human. Comput. 10 (7) (2019) 2545–2562.

[162] M. Williams, C. Benfield, B. Warner, M. Zadka, D. Mitchell, K. Samuel, et al., Push data to browsers and micro-services with WebSocket, in: Expert Twisted, Springer, 2019, pp. 285–304.

[163] R. Atmoko, R. Riantini, M. Hasin, IoT real time data acquisition using MQTT protocol, in: Journal of Physics: Conference Series, vol. 853, IOP Publishing, 2017, 012003.

[164] H.C. Hwang, J. Park, J.G. Shon, Design and implementation of a reliable message transmission system based on MQTT protocol in IoT, Wireless Pers. Commun. 91 (4) (2016) 1765–1777.

[165] X. Liu, T. Zhang, N. Hu, P. Zhang, Y. Zhang, The method of Internet of Things access and network communication based on MQTT, Comput. Commun. 153 (2020) 169–176.

[166] L.D. Xu, W. He, S. Li, Internet of things in industries: a survey, IEEE Trans. Ind. Inform. 10 (4) (2014) 2233–2243.

[167] B. Ahlgren, M. Hidell, E.C.H. Ngai, Internet of things for smart cities: interoperability and open data, IEEE Internet Comput. 20 (6) (2016) 52–56.

[168] Th Kim, C. Ramos, S. Mohammed, Smart city and IoT, Future Generat. Comput. Syst. 76 (2017) 159–162.

[169] A. Crooks, K. Schechtner, A.K. Dey, A. Hudson-Smith, Creating smart buildings and cities, IEEE Pervas. Comput. 16 (2) (2017) 23–25.

[170] T. Song, R. Li, B. Mei, J. Yu, X. Xing, X. Cheng, A privacy preserving communication protocol for IoT applications in smart homes, IEEE Internet Things J. 4 (6) (2017) 1844–1852.

[171] R. Zhou, Y. Xiong, G. Xing, L. Sun, J. Ma, ZiFi: wireless LAN discovery via ZigBee interference signatures, in: Proceedings of the Sixteenth Annual International Conference on Mobile Computing and Networking, 2010, pp. 49–60.

[172] L. Catarinucci, D. de Donno, L. Mainetti, L. Palano, L. Patrono, M.L. Stefanizzi, et al., An IoT-aware architecture for smart healthcare systems, IEEE Internet Things J. 2 (6) (2015) 515–526.

[173] A. Redondi, M. Chirico, L. Borsani, M. Cesana, M. Tagliasacchi, An integrated system based on wireless sensor networks for patient monitoring, localization and tracking, Ad Hoc Netw. 11 (1) (2013) 39–53.

[174] S. Wang, M. Xia, Y.C. Wu, Backscatter data collection with unmanned ground vehicle: mobility management and Power allocation, IEEE Trans. Wireless Commun. 18 (4) (2019) 2314–2328.

[175] A.A.R. Alsaeedy, E.K.P. Chong, Mobility management for 5G IoT devices: improving power consumption with lightweight signaling overhead, IEEE Internet Things J. 6 (5) (2019) 8237–8247.

[176] Domenico Ciuonzo, Aniello Buonanno, Michele D'Urso, Francesco AN. Palmieri, Distributed classification of multiple moving targets with binary wireless sensor networks, in: 14th International Conference on Information Fusion, IEEE, 2011, pp. 1–8.

[177] N.H. Motlagh, M. Bagaa, T. Taleb, UAV-based IoT platform: a crowd surveillance use case, IEEE Commun. Mag. 55 (2) (2017) 128–134.

[178] T. Sultana, K.A. Wahid, Choice of application layer protocols for next generation video surveillance using Internet of video things, IEEE Access 7 (2019) 41607–41624.

[179] A. Rego, A. Canovas, J.M. Jiménez, J. Lloret, An intelligent system for video surveillance in IoT environments, IEEE Access 6 (2018) 31580–31598.

[180] M.A. Alsmirat, Y. Jararweh, I. Obaidat, B.B. Gupta, Internet of surveillance: a cloud supported large-scale wireless surveillance system, J. Supercomput. 73 (3) (2017) 973–992.

[181] Allaa R. Hilal, Aya Sayedelahl, Arash Tabibiazar, Mohamed S. Kamel, Otman A. Basir, A distributed sensor management for large-scale IoT indoor acoustic surveillance, Future Generat. Comput. Syst. 86 (2018) 1170–1184.

[182] N.K. Tran, Q.Z. Sheng, M.A. Babar, L. Yao, Searching the web of things: State of the art, challenges, and solutions, ACM Comput. Surv. 50 (4) (2017) 1–34.

[183] L. Belli, S. Cirani, L. Davoli, A. Gorrieri, M. Mancin, M. Picone, et al., Design and deployment of an IoT application-oriented testbed, Computer 48 (9) (2015) 32–40.

[184] M. Wang, J. Santillan, F. Kuipers, ThingPot: an Interactive Internet-Of-Things Honeypot. arXiv Preprint arXiv:180704114, 2018.

[185] J.Z. Luo, C. Shan, J. Cai, Y. Liu, IoT application-layer protocol vulnerability detection using reverse engineering, Symmetry 10 (11) (2018) 561.

[186] M.A. da Cruz, J.J. Rodrigues, P. Lorenz, P. Solic, J. Al-Muhtadi, V.H.C. Albuquerque, A proposal for bridging application layer protocols to HTTP on IoT solutions, Future Generat. Comput. Syst. 97 (2019) 145–152.

[187] H. Alaiz-Moreton, J. Aveleira-Mata, J. Ondicol-Garcia, A.L. Muñoz-Castañeda, I. García, C. Benavides, Multiclass classification procedure for detecting attacks on MQTT-IoT protocol, Complexity 2019 (2019).

[188] A. La Marra, F. Martinelli, P. Mori, A. Rizos, A. Saracino, Introducing usage control in MQTT, in: Computer Security, Springer, 2017, pp. 35–43.

[189] R. B, L. Brasilino, M. Swany, Low-latency CoAP processing in FPGA for the internet of things, in: International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData), vol. 2019, 2019, pp. 1057–1064.

[190] Dae-Young Kim, Seokhoon Kim, Houcine Hassan, Jong Hyuk Park, Adaptive data rate control in low power wide area networks for long range IoT services, J. Comput. Sci. 22 (2017) 171–178.

[191] Y. Alsouda, S. Pllana, A. Kurti, Iot-based urban noise identification using machine learning: performance of SVM, KNN, bagging, and random forest, in: Proceedings of the International Conference on Omni-Layer Intelligent Systems, 2019, pp. 62–67.

[192] S. Lakshmanaprabu, K. Shankar, M. Ilayaraja, A.W. Nasir, V. Vijayakumar, N. Chilamkurti, Random forest for big data classification in the internet of things using optimal features, Int. J. Mach. Learn. Cyber. 10 (10) (2019) 2609–2618.

[193] J. Vizárraga, R. Casas, Á. Marco, J.D. Buldain, Dimensionality reduction for smart IoT sensors, Electronics 9 (12) (2020) 2035.

[194] A. Alhowaide, I. Alsmadi, J. Tang, PCA, random-forest and pearson correlation for dimensionality reduction in IoT IDS, in: 2020 IEEE International IOT, Electronics and Mechatronics Conference (IEMTRONICS), IEEE, 2020, pp. 1–6.

[195] Davide Cinquegrana, Emiliano Iuliano, Investigation of adaptive design variables bounds in dimensionality reduction for aerodynamic shape optimization, Comput. Fluid 174 (2018) 89–109.

[196] Morteza Safaei Pour, Bou-Harb, Elias, Kavita Varma, Nataliia Neshenko, Dimitris A. Pados, Kim-Kwang Choo, Raymond, Comprehending the IoT cyber threat landscape: a data dimensionality reduction technique to infer and characterize Internet-scale IoT probing campaigns, Digit. Invest. 28 (2019) S40–S49.

[197] Yang Li, Yuanyuan Bao, Wai Chen, A stable dimensionality-reduction method for internet-of-things (IoT) streaming data, in: IEEE International Conference on Internet of Things and Intelligence System (IoTaIS), vol. 2019, IEEE, 2019, pp. 231–237.

[198] G. Sun, J. Li, J. Dai, Z. Song, F. Lang, Feature selection for IoT based on maximal information coefficient, Future Generat. Comput. Syst. 89 (2018) 606–616.

[199] S. Egea, A.R. Mañez, B. Carro, A. Sánchez-Esguevillas, J. Lloret, Intelligent IoT traffic classification using novel search strategy for fast-based-correlation feature selection in industrial environments, IEEE Internet Things J. 5 (3) (2017) 1616–1624.

[200] B. Chakraborty, D.M. Divakaran, I. Nevat, G.W. Peters, M. Gurusamy, Cost-aware Feature Selection for IoT Device Classification, IEEE Internet of Things Journal, 2021.

[201] Dang Hai Hoang, Ha Duong Nguyen, A PCA-based method for IoT network traffic anomaly detection, in: 20th International Conference on Advanced Communication Technology (ICACT), vol. 2018, IEEE, 2018, pp. 381–386.

[202] M.P.R. Kiran, Sai, Pachamuthu Rajalakshmi, Performance analysis of CSMA/CA and PCA for time critical industrial IoT applications, IEEE Trans. Ind. Inform. 14 (5) (2018) 2281–2293.

[203] Hanjun Duan, Xu Zhu, Yufei Jiang, Zhongxiang Wei, Sumei Sun, An adaptive self-interference cancelation/utilization and ICA-assisted semi-blind full-duplex relay system for LLHR IoT, IEEE Internet Things J. 7 (3) (2019) 2263–2276.

[204] A.L. Mayilvahanan, N. Stalin, S. Sutha, Improving solar power generation and defects detection using a smart IoT system for sophisticated distribution control (SDC) and independent component analysis (ICA) techniques, Wireless Pers. Commun. 102 (4) (2018) 2575–2595.

[205] Xin Wan, Xu Zhu, Yufei Jiang, Yujie Liu, Jiahe Zhao, An Interference Alignment and ICA Based Semi-blind Dual-User Downlink NOMA System for High-Reliability Low-Latency IoT, IEEE Internet of Things Journal, 2020.

[206] V.R. Konda, J.N. Tsitsiklis, Actor-critic algorithms, in: Advances in Neural Information Processing Systems, 2000, pp. 1008–1014.

[207] F. Hussain, S.A. Hassan, R. Hussain, E. Hossain, Machine learning for resource management in cellular and IoT networks: potentials, current solutions, and open challenges, IEEE Commun. Survey. Tutor. 22 (2) (2020) 1251–1275.