Received June 15, 2021, accepted June 27, 2021, date of publication July 5, 2021, date of current version July 14, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3094763

A Systematic Review on the Data Interoperability of Application Layer Protocols in Industrial IoT

ANAM AMJAD, FAROOQUE AZAM, MUHAMMAD WASEEM ANWAR, AND WASI HAIDER BUTT
Department of Computer and Software Engineering, College of Electrical and Mechanical Engineering, National University of Sciences and Technology, Islamabad 44000, Pakistan

Corresponding author: Muhammad Waseem Anwar (waseemanwar@ceme.nust.edu.pk)

ABSTRACT Due to the rising popularity of the Internet of Things (IoT), interconnected infrastructures with better flexibility and efficiency are being developed for industry 4.0. The use of IoT in industry 4.0 introduces a new term ''Industrial IoT (IIoT)'', which allows remote monitoring of wide-ranging industrial processes. IIoT tends to function with a large number of heterogeneous devices, therefore, data interoperability is the biggest challenge in achieving seamless real-time communication. To achieve data interoperability, there is a need to integrate various IoT application layer protocols; separately, as well as, with other data acquisition protocols along with hardware/software platforms. Although such integrations are frequently presented in the state-of-the-art, a comprehensive investigation and analysis of such approaches in single research work are hard to find. Therefore, in this article, a Systematic Literature Review (SLR) is presented by investigating 34 influential research works, published during 2014-2020, where interoperability of application layer protocols is performed for IIoT. Consequently, the selected research studies are divided into three categories: 1) Integration of IoT application layer protocols— 13 papers; 2) Integration of IoT and data acquisition protocols—6 papers; 3) Integration of IoT protocols with hardware/software platforms—15 papers to resolve interoperability issues. Moreover, nineteen (19) tools used for integration in the selected research works are identified. Furthermore, leading approaches to perform such integrations are recognized, e.g., Gateways. It has been concluded that although researchers frequently propose centralized (e.g., gateway or protocol converter) and distributed intermediator (e.g., Middleware) to achieve required interoperability for IIoT, it is a need of a day to develop a device level solution without the involvement of intermediator. This will solve different interoperability problems in industries like scalability, load balancing, and a single point of failure.

INDEX TERMS Application layer protocols, Internet of Things (IoT), industrial IoT, interoperability, integration, systematic literature review (SLR).

I. INTRODUCTION

Industrial IoT (IIoT) has become an important component for the implementation in industry 4.0 as it integrates the physical, social and cyber objects/things with industry and behaves intelligently to serve it's people [1], [2]. Up-till now, industries from different sectors i.e. logistics, transportation, and manufacturing are the major players of the IIoT market. For example, one of the smart cities Barcelona is successfully using intelligent parking system and automated transport system. According to the IDC prediction, estimated IoT devices will be 41.6 billion by 2025 [3]. In IIoT devices, data acquisition, processing, and real-time communication are the major cornerstones. However, communication relies on the heterogeneous smart devices speaking different languages leading to the interoperability issue. As per McKinsey analysis, one of the threats to predicted economic value is missing interoperability [4].

IEEE defines interoperability as ''the ability of two or more systems to exchange information and use of the information that has been exchanged'' [4]. Lack of interoperability in IoT forms the basis of different issues such as 1) Vendor lock-in: From the IoT provider perspective, IoT market is fragmented due to a wide range of heterogeneous services, applications, data formats, communication protocols, and devices. Here, lack of interoperability means users are restricted to use sole IoT device or software offered by a single provider bringing higher potential risks of operational

The associate editor coordinating the review of this manuscript and approving it for publication was Chien-Ming Chen.

cost and product functionality. 2) Non-applicability to cross-platform: From the application developer perspective, existing IoT systems are designed using application-specific API which is difficult to operate in a cross-platform environment. Low support for applications executing on different platforms is resulting in non-interoperability [4].

Few categories of interoperability related to IIoT are as follows. 1) Device-level interoperability [5]: To provide information exchange between physical and software components of the smart devices including communication protocols [6] e.g., Zigbee, Message Queuing Telemetry Transport (MQTT) and Constrained Application Protocol (CoAP). 2) Network interoperability [1]: It is required to deal with seamless communication of devices over different networks (e.g. wired and wireless networks). 3) Semantic interoperability [7]: It is linked with the meaning of the content for humans rather than machine interpretation of the content and 4) Platform interoperability [8]: It offers collaboration of the diverse platforms used in IoT due to diverse operating system, programming languages, and access mechanisms for data and things. As IIoT devices are increasing with each passing day, all these types of interoperability are required to support heterogeneity issues. It is important to note that the focus of this article is device-level interoperability where the heterogeneity of application layer protocols is a primary concern.

Generally, IIoT is challenged by the data interoperability of heterogeneous devices due to different vendors. So, interoperability acts as a potential barrier to achieve seamless data acquisition. To resolve this issue, interoperability can be achieved by either pushing standards or integration approaches. For example, in industry, one of the well-known Machine to Machine (M2M) standard offering interoperability at different layers is the European Telecommunication Standard Institute (ETSI) [9]. Also, interoperability in Industrial IoT is performed using diverse integration approaches such as gateway [10], [11] and middleware [12]. This paper aims to analyze the integration of different IoT application layer protocols due to the reason that application layer is the primary interface for data interoperability by interacting with different services and systems offered [13], [14].

Mostly, IoT devices use a single application layer protocol based on the primary requirements of the intended application. Such as requirement to turn on the light switch is different from monitoring the transmission lines. Moreover, if one protocol is good at securing the IoT devices then another protocol can be better at supporting low power and bandwidth usage. In literature, several research articles exist in the context of IoT communication. For example, a survey/review is performed in [8] to analyze the energy efficiency of heterogeneous communication protocols while performing interoperability. However, the main focus of the paper is the energy efficiency. Many research studies presents the general idea of the IoT communication in review paper such as [15] identifies the IoT as an ecosystem and provides detail on the various IoT components including architecture, devices, gateways, operating systems, middleware and communication protocols. Another survey [16] also discusses the architectures, protocols, applications, recent advancement in the IoT. However, the exhaustive survey which takes into account the integration or interoperability of application layer protocols is not present. Moreover, [17] presents the interoperabilty for context information sharing that is different from data interoperability. Data interoperability is important to enable communication between different applications and users as presented in this research study. The integration of application layer protocols to achieve data interoperability in IIoT is an attractive area for researchers in recent years. Several solutions have been proposed in this regard. However, a comprehensive Systematic Literature Review (SLR) targeting IIoT interoperability for application layer protocols is hard to find in the literature. We try to bridge this gap by systematically aggregating the research studies based on six research questions.

RQ1: What is the number of studies dealing with interoperability for industrial IoT, where the following integrations are performed: 1) IoT application layer protocols; 2) IoT application layer protocols with other data acquisition protocols; 3) IoT application layer protocols with hardware/software platforms?

RQ2: What are the significant application layer protocols used for the integration to meet data interoperability objectives in IIoT?

RQ3: What are the mostly used integration approaches used in IIoT?

RQ4: What are the leading tools to carryout protocol integration for IIoT?

RQ5: What are the important parameters to evaluate the performance as a result of integrated IoT application layer protocols?

RQ6: What are the challenges and corresponding future directions for the integration of protocols in IIoT?

It is the first SLR on the integration of IoT application layer protocols for industrial IoT and the major list of contributions made in this SLR is as follows.

1) This study comprehensively investigates device-level interoperability in IIoT with a focus on the heterogeneity of application layer protocols. To achieve this, three categories are developed, and analyzed i.e 1) Integration of IoT application layer protocols individually (13 research studies), 2) Integration of IoT application layer and data acquisition protocols (6 research studies) and 3) Integration of IoT application layer protocols with hardware/ software platform to meet data interoperability (15 research studies)
2) Existing integration approaches (i.e. centralized and distributed intermediator) in selected studies are highlighted and analyzed, which are used to integrate application layer protocols. Gateways are commonly used to integrate two or more than two application layer protocols. These gateways exist at either network layer or in the cloud to perform translation of the source to target application layer protocols. Few other integration

approaches include Application Programming Interface (API), middleware and Software Defined Network (SDN) used in the selected research studies.

3) Several tools used to achieve interoperability through the integration of application layer protocols are also recognized. Such that tools used for the integration of MQTT, CoAP and XMPP are identified. This analysis will help the practitioners to pick the right tool to enhance the interoperability.
4) The detailed analysis of selected studies leads to identify and present the realistic challenges and significant future directions for the integration of application layer protocols in Industrial IoT. Few challenges are security, scalability and diversity of existing gateways.

The organization and structure of this SLR providing the complete overview is shown in Figure 1. We have selected four scientific repositories in the search process (Section III-C). As a result, thirty-four (34) research studies are identified to fully comply with inclusion and exclusion criteria (Section III-B). The selected research studies are classified, and data is extracted with the help of data extraction and synthesis steps mentioned in Section III-E. For data synthesis, three integration categories: 1) Integration of IoT application layer protocols (Reference [10], [28]–[39]), 2) Integration of IoT and data acquisition protocols (Reference [11] and [40]–[44]) and 3) Integration of IoT protocols with hardware/software platforms (Reference [12], [45]–[58]) are created as defined in Section II-A. The detailed discussion on these categories is carried out in Section IV. Tool analysis is performed in Section IV-D and evaluation parameters for IoT application layer protocols are identified in Section IV-E. The comprehensive study of selected research studies leads towards the answer of research questions (Section V) and finally, SLR is concluded in Section VI.

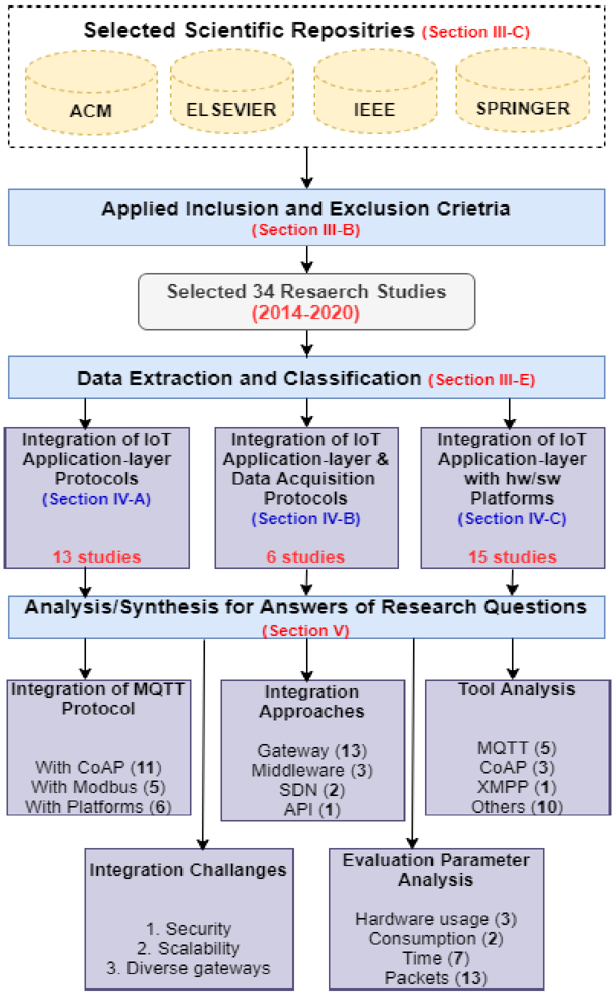
Selected Scientific Repositries (Section III-C)
ACM
ELSEVIER
IEEE
SPRINGER
Applied Inclusion and Exclusion Crietria (Section III-B)
Selected 34 Resaerch Studies (2014-2020)
Data Extraction and Classification (Section III-E)
Integration of IoT Application-layer Protocols (Section IV-A) 13 studies
Integration of IoT Application-layer & Data Acquisition Protocols (Section IV-B) 6 studies
Integration of IoT Application-layer with hw/sw Platforms (Section IV-C) 15 studies
Analysis/Synthesis for Answers of Research Questions (Section V)
Integration of MQTT Protocol
With CoAP (11)
With Modbus (5)
With Platforms (6)
Integration Approaches
Gateway (13)
Middleware (3)
SDN (2)
API (1)
Tool Analysis
MQTT (5)
CoAP (3)
XMPP (1)
Others (10)
Integration Challanges
1. Security
2. Scalability
3. Diverse gateways
Evaluation Parameter Analysis
Hardware usage (3)
Consumption (2)
Time (7)
Packets (13)

FIGURE 1. Overview of the SLR.

II. BACKGROUND

The understanding of a few basic concepts is essential before going into the details of SLR. Therefore, in this section, a brief introduction of application-layer protocols for interoperability and integration approaches is presented. Particularly, IIoT application-layer protocols for interoperability are discussed in Section II (A) and an overview of integration approaches is provided in Section II (B).

A. APPLICATION-LAYER PROTOCOLS FOR INTEROPRABILITY

In Industrial IoT, ''things'' have the capabilities to collaborate and communicate with each other. IoT protocols are used extensively to enable communication among heterogeneous IoT devices. It is useless to have several IoT devices if these cannot communicate with each other [16]. Therefore, dissimilar protocols at any layer of IoT architecture used by IIoT lead to interoperability issues [17]. Due to this reason, heterogeneity is harmonized using an integration of dissimilar application-layer protocols. It is essential to discuss the introduction and working of a few application layer protocols before moving to the next sections of this SLR where these protocols are used frequently. IoT application layer protocols are divided into two categories; device to device protocols and device to server protocol as explained below.

1) DEVICE TO DEVICE (D2D) PROTOCOLS

These protocols are used for the independent communication of smart devices. One of its examples is the Data Distribution Service (DDS).

1) *Data Distribution Service (DDS):* It is a device-to-device application layer standard introduced by Object Management Group (OMG) in 2004 for high performance, durability, and interoperability of data using real-time publish/subscribe pattern [18]. Publisher and subscriber exchange messages directly without a broker. One of the exciting features of DDS is publisher can publish data if no subscriber is interested while subscriber can have the information of publisher and required information of QoS.

It performs decentralized, peer to peer and multicast communication. DDS is data-centric protocol which means it defines the type and content of data. DDS architecture constitutes of two layers; Data-Centric Publish-Subscribe (DCPS) layer responsible for sharing of data from publisher to subscriber and Data-Local Reconstruction Layer (DLRL) responsible for sharing of data between distributed objects. DLRL is an optional layer. At the transport layer, UDP is used by default, but DDS can optionally facilitate TCP. Security is provided using TLS or DTLS depending upon the use of TCP or UDP, respectively. DDS can support both powerful and low-capacity devices for IoT applications [18].

2) DEVICE TO SERVER (D2S) PROTOCOLS

D2S is known for the transport of collected data from devices to server infrastructure. Few examples include AMQP, CoAP, MQTT and XMPP and REST HTTP.

1) *Application Layer Queuing Protocol (AMQP):* This protocol was originated in 2003 by John O'Hara. It has now become an OASIS standard. AMQP is based on the publish/subscribe pattern where publisher generates messages and subscribers received these messages with the help of named exchange attached to the message queue. There is corresponding unique key in each queue, such as Temperature has "Temp" unique key. A broker is placed between the publisher and subscriber to ensure that the right message is published and delivered to a subscriber. AMQP is made secure using TLS on transport layer beneath. TLS encrypts the sensitive information with public key which can only be decrypted by a corresponding private key stored on a server. This protocol is designed to exchange business messages for economic industries. AMQP provides security, reliability, interoperability, and open features [19].
2) *Constrained Application Protocol (CoAP):* This lightweight web transfer application layer protocol is standardized by the Internet Engineering Task Force (IETF) Constrained RESTful Environment (CoRE). CoAP is mainly designed for constrained nodes/networks in IoT. It is suitable to operate on low energy devices. It is based on a client-server architecture where the working of CoAP server is like standard servers. A client sends a request to the server and servers respond. The client can access the server using PUT, POST, GET and DELETE requests.
   CoAP operates using UDP on the transport layer to reduce the overhead. For UDP, the security of a server connection is provided using Datagram Transport Layer Security (DTLS) [20]. CoAP is a flexible, scalable protocol as home and building automation web applications deploy CoAP at the application layer.
3) *Message Queuing Telemetry Transport (MQTT):* It is a lightweight application layer protocol based on publish/subscribe pattern recently standardized by OASIS. It was created way back in 1999 by two engineers Andy Stanford-Clark and Arlen Nipper. It gained popularity due to its small size, consumption of less energy and well-organized distribution of information to its receivers. The working principle of MQTT is like client and server schema where the server is referred to as broker as an intermediary service here. It has two essential characteristics, i.e. message buffer to store messages temporarily and three Quality of Service (QoS) levels. Levels 0 is fire and forget, level 1 is for at least once delivery, while level 2 is associated with at most once delivery. ACK is responsible for message delivery in level 1 and level 2 [21].
   Two types of clients are interacted known as publisher and subscribers. Clients do not communicate directly; the communication takes place with the help of a broker. Every message which is required by the user is first subscribed to the broker, and through a broker, it can be published to the subscriber. Every message is subscribed and published using a hierarchical topic name, e.g. Hotel/Room101/Lights/Status. MQTT runs on top of TCP protocol at the transport layer, so the security mechanism adopted for TCP is Transport Layer Security (TLS). MQTT is extensively used in IoT and provides broad tool support. MQTT is best to use for resource- constrained environments such as smart homes [22].
4) *Extensible Messaging and Presence Protocol (XMPP):* It was introduced by Jeremie Miller in 1999. Real-time communication is based on the XML technology which can implement both request/response and publish/subscribe architectures with appropriate XMPP extension. Currently, there are various extensions of XMPP.[1] Client and server communication takes place using stanza. Four commonly used attributes of the stanza are from, to, type, and id. From represents the sender ID, if there is no content in "from" attribute, then the sender is server. To indicate the recipient's ID, if there is no content in "to" attribute then recipients is the server. Type of stanza is represented by "type" attribute, and "id" is used to specify the questions/answers regarding the stanza. In XMPP, there is generally three types of stanza; 1) Presence for registration of an entity, 2) Message to send data from one entity to another and 3) IQ (Info/Query) for receiving information from a server and sending settings. XMPP is used to send instant messages, audio and video calls along with authentication, access control, and encryption services [23].
5) *Representational State Transfer Hyper Text Transfer Protocol (REST HTTP):* It is based on client/server

[1] https://xmpp.org/extensions/

architecture where a client sends a request to the server and server respond. Recently, HTTP facilitates the REST architecture for the efficient development of web applications as it allows different entities to interact using web services. Integration of HTTP and REST allowed the devices to make their status available for CRUD (Create, Read, Update, Delete) operations. HTTP/2.0 is an improved version of HTTP/1.0 in the context of resource consumption, overhead and power usage; thus, it appears to be suitable for IoT devices for communication. At the transport layer, TCP is used, and ultimately, security is provided using TLS to encrypt and authenticate communication. REST HTTP does not provide the QoS levels for communication, but still, it is suitable for IoT web applications [24].

B. INTEGRATION APPROACHES

The integration of application layer protocols in Industrial IoT is carried out using different approaches in literature [4], [5]. For example, 1) Gateway is used to address the interoperability between IoT devices and perform conversion of the protocol of sending device to a protocol of receiving device. When IoT devices use dissimilar application layer protocols, the gateway acts as a hardware/software embedded in Programmable Logic Control (PLC) or Human Machine Interface (HMI) to perform an integration [10], [11]. In other words, gateway becomes an additional intermediator for communication between IoT devices where devices do not communicate directly.

Moreover, 2) Software Defined Network (SDN) is another network-level integration approach found in [1]. The objective of SDN is to enable the communication of heterogeneous devices such that sending devices in the router makes the information understandable to receiving devices. 3) Middleware [12] based approach provides a single interface to the heterogeneous protocols in dynamic IoT devices. 4) Application Program Interface (API) provides an interface to present data or functions in a high-level language for any specific application. The main objective to develop API is to facilitate cross-platform and cross-domain interoperability. Some research studies does not use gateway, SDN, middleware or API to perform integration. Due to this, a different category named as 5) others category is introduced. Other category may include broker extension or use of programming language to support integration of different application layer protocols.

III. REVIEW PROTOCOL

Among six sections of review protocol, two sections (Background and Research questions) are already discussed in the introduction (Section I). Other five sections (Category definition, Inclusion and exclusion criteria, Search Process, Quality assessment, Data extraction and synthesis) are discussed in subsequent sections as per the guidelines of SLR standard [25].

A. DEFINING CATEGORIES

Three categories are defined for the integration of application layer protocols in IIoT to answer developed research questions. A description of these categories is given below.

1) *Integration of IoT Application Layer Protocols:* In this category, only those research studies are placed where minimum two IoT application layer protocols are integrated to enhance interoperability. This category is important as IoT devices are becoming vertically integrated and face interoperability issues.
2) *Integration of IoT and Data Acquisition Protocols:* The proprietary data acquisition protocols used in industries usually leads to limited IoT solution due to additional costs. When a user is stuck with single vendor, interoperability issues arise. To solve this issue, a variety of data acquisition protocols such as Modbus, Profibus and OLE for Process Control (Object Linking and Embedding for process control) or OPC, etc. are integrated with application layer protocols to facilitate IIoT. By considering it, only those research studies are selected where integration of a minimum one IoT and data acquisition protocol is performed.
3) *Integration of IoT Application Layer Protocols With Hardware or Software Platforms:* In this category, existing IoT application layer protocols are made interoperable for hardware/ software platforms to be efficiently utilized in Industrial IoT applications. For example, CoAP is suitable for constrained resources. To enhance the interoperable capabilities CoAP protocol can be integrated with some hardware/software platform (e.g., FPGA/Amazon), which are discussed in this category. This category is beneficial to develop interoprable systems or system of system with reduced cost and high productivity.

B. INCLUSION AND EXCLUSION CRITERIA

The inclusion and exclusion criteria are summarized in Table 1. We have included only those research studies performing integration of application layer protocols, written in English, fall in four selected scientific databases (i.e., ACM, Elsevier, IEEE and Springer), published during 2014-2020 and peer reviewed. Other than these criteria, research studies are excluded.

C. SEARCH PROCESS

The search process is initiated by exploring four scientific databases i.e. ACM, Elsevier, IEEE Springer. Research papers are selected by applying inclusion criteria in Table 1 using nine (9) keywords. Only AND operator is used to carry out search process. OR operator is not used because it generated large number of results which are redundant and incompetent. Obtained results are further refined by year span that is from 2014 to 2020 and results are shown in Table 2. Among these research studies, thirty-four (34)

TABLE 1. Inclusion and exclusion criteria.

Sr.No	Inclusion Criteria	Sr.No	Exclusion Criteria
I1	Research studies performed the integration of application layer protocols for industrial IoT. This includes integration of IoT application layer protocols, integration of IoT and data acquisition protocols and integration of IoT with hardware/software platforms.	E1	Integration of other protocols is excluded such as [26] has performed integration of Named Data Networking (NDN) with IEEE802.15.4. This research study has performed integration for network layer and also not in industrial IoT context.
I2	Academic papers published in four globally recognized databases i.e. ACM, Elsevier, IEEE and Springer.	E2	Other than selected databases such as ISI web of science, Scopus, CiteSeerX and EI Compendex etc.
I3	Research studies published from 2014 to 2020.	E3	Studies published before 2014.
I4	Papers written in English language.	E4	Non-English written papers.
I5	Peer-reviewed studies.	E5	Non-peer reviewed studies.
		E6	Duplicated papers found in digital libraries (Most recent, complete, and improved version is selected).
		E7	Gray literature i.e. Abstracts, keynotes, editorials, prefaces, etc.

TABLE 2. Summary of keywords with results.

Keywords	Operator	ACM	Elsevier	IEEE Xplore	Springer
IoT, Communication protocols	AND	4	58	21	88
IoT, Application layer protocols	AND	2	8	10	13
IoT,Messaging protocols	AND	3	7	4	12
IoT, Integration	AND	34	114	50	166
Industrial IoT, Integration	AND	0	2	1	1
IoT, Interoperability	AND	20	35	65	70
Interoperability, Communication	AND	6	108	12	18
IoT, Connectivity	AND	42	90	96	165
IoT, Gateway	AND	130	286	204	531
Total		241	708	463	1064

papers are selected by performing certain steps as illustrated in Fig. 2.

1) We considered total 2,476 studies and 1,901 numbers of studies are discarded based on their title. The title-based exclusion is performed by authors.

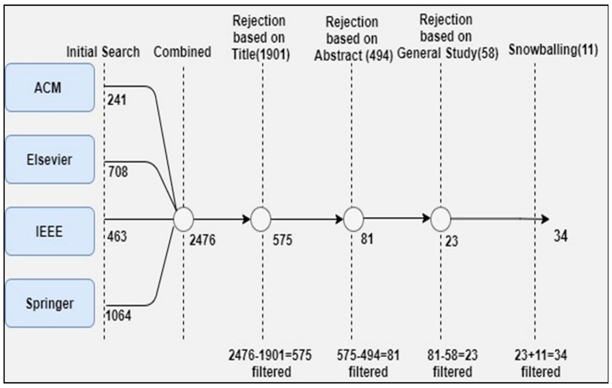

FIGURE 2. Summary of search process.

2) Remaining studies are 575. Among them, research studies eliminated based on abstract is 494.
3) The research studies eradicated based on general studies are 58. Such as [27] meets the inclusion criteria but title and abstract based rejection is not performed. However, different section such as introduction, communication and design principal are studied to know that this research study does not answer any of the research questions. Due to this reason, these type of research studies are discarded and as a result, total twenty-three (23) studies are identified.
4) Snowballing technique added eleven more research studies in selected papers making it thirty-four (34). In snowballing, references of the selected studies are considered to find relevant publications. This step intends to include relevant studies that are not identified in already assessed papers.

D. QUALITY ASSESSMENT

Quality assessment is a symbol of formative assessment cycle performed to ensure the precision of this SLR. Checklist to meet quality standard is developed(QA1-QA4) for selected research studies before comprehending the results.

QA1:Are selected research studies for integration of IoT application layer protocols ranges from 20142020?

QA2:Are selected research studies belong to the globally known scientific databases?

QA3:Do the selected research papers have used tools, frameworks, platforms and evaluation parameters to help us understand the integration for practical implications?

QA4:Does the data assessment of the research studies, based on the ground realities without any ambiguity?

Firstly, we tried to consider latest research studies of last six years only, i.e., from 2014 to 2020. The distribution of selected research studies based on each year is graphically represented in Fig. 3 below.

Secondly, it is ensured that selected research studies are published in well-known databases. Moreover, it is analyzed that selected research studies have used tools, framework, platforms or evaluation parameters to highlight the

TABLE 3. Data extraction and synthesis template.

Sr. No	Data	Description	Purpose
1	Study identifier	Unique ID of every research study	Documentation
2	Title, Year	–	Documentation
3	Article source	ACM, IEEE, Elsevier, Springer	Documentation
4	Type of article	Journal or conference	Documentation
5	Application context	Academic or industrial	Documentation
6	Research type	Validation research, solution proposal, experience papers	Documentation
7	Evaluation method	Controlled experiments, case study, action research	Documentation
8	Grouping of integration approaches	Categories of selected research studies (Table 5)	RQ1
9	Integration approaches	Integration of application layer protocols (Table 6), Integration of IoT and data acquisition protocols (Table 7), Integration of IoT application layer protocols with other hardware/software platforms (Table 8)	RQ1 and RQ2
10	Techniques of integration	Gateway, SDN, API, Middleware and others (Table 6, 7 and 8)	RQ3
11	Tools for integration	Tool analysis (Table 12)	RQ4
12	Evaluation parameters	Evaluation parameters for integration of IoT application layer protocols (Table 13)	RQ5
13	Challenges and future directions	Analysis of selected research studies (Table 6,7,8,9,10,11,12 and 13)	RQ6

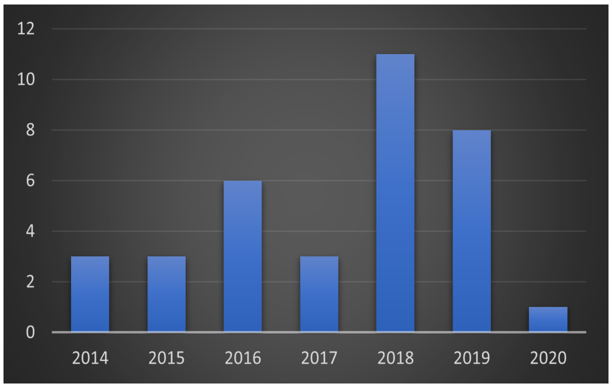

FIGURE 3. Distribution of selected research studies per publication year.

experimental aspects of these research studies and the data assessment of the research studies is based on the ground realities without any ambiguity.

E. DATA EXTRACTION AND SYNTHESIS

The aim to develop extraction and synthesis template was to find the answer of our six research questions. First, bibliographic information (i.e. title, year, source, type) is extracted. Then, research studies are observed based on their application context, research type and evaluation method.

At the end, data synthesis for each research question is done as shown in Table 3.

IV. RESULTS AND ANALYSIS

As shown in Table 4, to simplify the process of data synthesis, the selected research studies for integration of IoT application layer are divided into three (3) categories. Moreover, the references for each category are also provided for further investigation by the reader. The analysis of each category is provided in subsequent sections.

A. INTEGRATION OF IOT APPLICATION LAYER PROTOCOLS

In this section, a summary of these protocols is provided as shown in Table 6. The description of four (4) parameters

TABLE 4. Grouping of selected research studies.

Sr. No	Categories	References	Total
1	Integration of IoT application layer protocols	[10] [28] [29] [30] [31] [32] [33] [34] [35] [36] [37] [38] [39]	13
2	Integration of IoT and data acquisition protocols	[11] [40][41] [42] [43] [44]	6
3	Integration of IoT application layer protocols with hardware or software platforms	[45] [46] [47] [48] [49] [50] [51] [52] [53] [54] [55] [56] [57] [58] [12]	15

used in Table 6 is elaborated here. 1) Ref. No. refers to a research paper under consideration. 2) Application layer protocol lists the name of protocols which are integrated in a referred research paper. For this purpose, five extensively used application layer protocols (CoAP, DDS, HTTP, MQTT and XMPP) are selected. AMQP is excluded because no research study of this section integrated AMQP with other IoT application-layer protocols. 3) Integration approach (Section II-B) used is extracted from each research study and placed under this parameter. 4) Data storage is required when incoming data from IoT devices requires a medium for processing of data. Commonly used medium to store data is cloud or edge in the IoT.

Seven (7) out of thirteen (13) research studies have used gateway-based approach for integration. Such as, [10] has proposed an "atlas thing communication framework" by integrating CoAP, HTTP REST and MQTT using gateway for higher interoperability. Both research studies [28] and [29], integrate the same two IoT application layer protocols, i.e., MQTT and CoAP by extending Kura gateway. Kura gateway had built-in MQTT features, whereas CoAP is additionally integrated using REST API for efficiency, scalability and resource optimization in heterogeneous devices. By doing so, management operation such as resource limitation can be simplified. Another paper [30], proposes the same gateway-based approach for the co-existence of three application layer protocols CoAP, MQTT XMPP.

TABLE 5. **Summary for integration of IoT specific application layer protocols.**

Ref.No	CoAP	DDS	HTTP	MQTT	XMPP	Integration approach	Storage
[10]	✓	×	×	✓	✓	Gateway	Cloud
[28]	✓	×	×	✓	×	Gateway	–
[29]	✓	×	×	✓	×	Gateway	–
[30]	✓	×	✓	✓	×	Gateway	Cloud
[31]	✓	×	×	✓	×	MCI	–
[32]	✓	×	×	✓	×	Gateway	–
[33]	✓	×		✓	×	Broker	Cloud
[34]	×	×	✓	✓	×	Gateway	Server
[35]	✓	×	✓	×	×	Gateway	Cloud
[36]	✓	×	✓	✓	×	APAL	–
[37]	✓	×	×	✓	×	Java	Cloud
[38]	✓	×	✓	✓	×	SDN	Cloud
[39]	✓	×	×	✓	×	Middleware	Server

By making these application-layer interoperable, high-level knowledge can be acquired from low-level sensors data. Furthermore, [32] proposes Secure Multi-protocol Integration Bridge for IoT (SeMIBIoT) using a gateway. Six type of translation scenarios are used to show the applicability of SeMBIoT; 1) HTTP to MQTT, 2) HTTP to CoAP, 3) HTTP to XMPP, 4) MQTT to CoAP, 5) XMPP to MQTT and 6) XMPP to CoAP. As shown in Table 5, two more research studies [34] and [35] fall under gateway-based approach for integration (Section II-B). In [34] and [35], the proposed integration enables the application layer protocols to talk with REST endpoints and embedded systems respectively. In both studies, the gateway takes input from source protocol and convert it into the target protocol, commonly exists at the network layer. In the above research studies, main objective of using gateway is to provide translation/conversion between different application-layer protocols. Therefore, gateway acts as an intermediator between heterogeneous IoT devices and perform virtualization however, gateways do not ensure the scalability as if number of IoT devices increases it face load balancing problem and can degrade the system performance.

Other than gateways, few approaches for interoperability among different application layer protocols are also discussed here. Yet another selected study [31] provides an intermediator-based approach known as MQTT-CoAP Interconnector (MCI) to translate the messages between CoAP server and MQTT broker while [33] presents an IoT Multi-protocol Application layer Broker (IoTM2B), where, input message of CoAP and MQTT plugins are transformed into HTTP with the help of a broker in Web of Things (WoT). WoT is an emerging area utilizing IoT resources as the web resources. The integration achieved in [31] and [33] helps IoT devices to communicate without considering the application layer protocol to be used. Moreover, a research study [36] performs integration of IoT devices using Application Protocol Abstraction Layer (APAL) offering event consumer and dispatching facilities for incoming and outgoing messages. Similarly, [37] implements Jolie for IoT (JIoT) that is a java-based programming language to enable IoT cross-layer and cross-platform communication possible. In [38], integration is performed with the help of the Software Defined Network (SDN) as discussed in section II-B. Generally, interoperability has its own issues which involves cost, overhead, scalability and security. Few issues in these diverse integration approaches can be analyzed. Such as the number of application layer protocols is not scalable. Transparency is not achieved as many approaches needs configuration to make it operational.

Lastly, another research study [39] uses a middleware approach for integration which provides a single unified interface. Middleware address the interoperability by bridging the application-layer protocols however, the issue is middleware requires an additional interface to make it interoperable among other middlewares.

From these research studies, it is also analyzed that incoming data towards IoT application layer protocols are either stored in cloud or server for further processing. In six (6) research papers [10], [30], [33], [36], [37] and [38], the cloud is used to store data because here, the data is locally processed in a centralized manner to enable decision making. Cloud is frequently used to store and process the data for IoT devices because it is secure and cost effective than other approached used. In [28], real-time data is used without any persistent data storage medium. References [35] and [39] have used server to store data for further processing.

From Table 5, it is analyzed that eleven (11) out of thirteen (13) research studies have integrated MQTT and CoAP together. The reason of MQTT and CoAP integration is that both protocols are lightweight, energy efficient and work well in constrained environment(e.g. smart homes). Six (6) out of thirteen (13) research studies have focused on integration of MQTT and REST HTTP protocol while MQTT is integrated with XMPP in only two (2) research studies. The reason to integrate MQTT with either CoAP or XMPP enables the Web usage in IIoT because CoAP is a lightweight version of HTTP. However, after analysis, it is revealed that no selected research study has integrated DDS with other IoT application layer protocols despite of the fact that DDS is the most versatile IoT application layer protocol and manage communication between tiny devices and ensure high performance.

Besides, DDS eliminates the presence of broker or server and thus, easy to deploy and maintain.

B. INTEGRATION OF IOT DATA ACQUISITION PROTOCOLS

In this category, total six (6) research studies fulfilling the desired criteria are placed. In Table 6, IoT application layer protocols i.e., DDS, MQTT and HTTP are used by selected research studies. This is the reason CoAP and XMPP are not listed. Moreover, in Table 6, three industry data acquisition protocols: Modbus, OPC UA and MMS are considered, which are used by any of the six research studies. Integration approach used and Data storage parameters are same as Table 5.

MQTT and Modbus protocols are frequently integrated in industrial IoT such as a research study [11] performs integration of MQTT and Modbus using gateway while in [40], same protocols MQTT and Modbus are integrated using Node-Red. The aim of MQTT and Modbus integration is to increase the market growth by industrial manufacturing using IoT services. In both research studies, conversion of incoming messages from Modbus into MQTT is performed where the sole purpose of Modbus is to read/write request, whereas, MQTT is recognized as transmitting the messages using a broker. Another paper [42] presents the integration of MQTT being an IoT application layer protocol with Modbus as industry data acquisition protocol using API to help in growing Personal Digital Assistant (PDA). Three APIs for iOS, Windows and Android are written by C programming language to support interoperability. However, developing a common and open-source API is the need of the hour to enable interoperability. Also, [43] proposes a communication platform to optimize multi micro-grid applications using a gateway in which Modbus uses TCP/IP protocol for simplicity and inter-microgrid communication is performed using MQTT protocol. Other than MQTT and Modbus, few protocols are also integrated. For example, a research study [41] proposes the integration of DDS and OPC Unified Architecture (OPC UA) protocols using mapping rules. By doing so, both protocols can be used in industrial IoT with the help of similar communication patterns. Moreover, [44] proposes an integration of Manufacturing Message Specification (MMS) with MQTT using Broker Extension (BE) for industrial applications. This integration aims to overcome the limitation of MMS as it does not support single message control for multiple devices. MQTT broker is extended with the help of Arduino in such a way that it connects with MMS client, MMS server and real machines to send and receive data. It is summarized that four out of six research studies have used the cloud to store the data.

It is further analyzed from Table 6 that, IoT application-layer protocol MQTT is used by five (5) out of six research studies to integrate with data acquisition protocols. The reason of using MQTT is that it provides efficient and secure integration of industrial automation with cloud functionality in IIoT. Moreover, the combination of MQTT and Modbus is analyzed in four (4) out of six research studies to increase the robustness and performance required in industrial setting. The reason for integrating MQTT with other industrial protocols is to increase the interoperability of IoT devices. It is done to optimize production with efficiency and low cost in industry 4.0.

C. INTEGRATION OF IOT APPLICATION LAYER PROTOCOLS WITH HARDWARE/SOFTWARE PLATFORMS

In this category, the important parameters used in Table 7 are as follows; Along with 1) Ref. No., 2) IoT application layer protocol represents four protocols i.e. CoAP, DDS, MQTT and HTTP, which are used in the selected fifteen (15) research studies. 3) Hardware(HW)/ software(SW) platforms are also integrated with IoT application layer protocols. Hardware is any physical device used in / with machine, while software

TABLE 6. Summary for integration of IoT application layer protocols with industry data acquisition protocols.

Ref. No	DDS	MQTT	HTTP	Mod-bus	OPC UA	MMS	Integration approach	Storage
[11]	×	✓	×	✓	×	×	Gateway	Cloud
[40]	×	✓	×	✓	×	×	DIL	Cloud
[41]	✓	×	×	×	✓	×	Mapping Rules	–
[42]	×	✓	×	✓	×	×	Forms API	Cloud
[43]	×	✓	✓	✓	×	×	Gateway	Cloud
[44]	×	✓	×	×	×	✓	BE	–

system refers to the program instructions to perform a specific operation/task. Software platforms are further categorized into standards, architectures and software system. A standard comprises set of rules, which is used by one or more developers to work on computer programs. Architecture defines the structure of the solution that meets operational and technical requirements. 3) Integration approach parameters is same as mentioned in Table 5 and Table 6.

First research study of this category [45], presents the integration of DDS with Software Defined Network (SDN) that is an architecture for providing network agility and flexibility through its controllers. The DDS and SDN integration enhance the interoperable capabilities in IoT. In [46] and [47], integration of CoAP with two different standards such as IEC 61850 using Concise Binary Object Representation (CBOR) and ISO IEEE 71103 Domain Information Model (DIM) using the gateway is presented to achieve data interoperability. IEC is a standard for substations in power grids used for their modeling, controlling and monitoring while ISO IEEE 71103 standard is used to develop domain information model. Gateway act as an intermediator to enable the communication.

In [48], the IoT platform of Amazon Web Service (AWS) known as IoT Core is integrated with MQTT, a protocol for sending and receiving messages using API of MQTT. Therefore, API based integration approach can be useful for cross platform communication between heterogeneous IoT devices. The IoT core platform used helps to efficiently integrate diverse IoT devices with AWS services. Another research study [49] integrates HTTP protocol with Web of Virtual Things (WoVT) software using gateway. Gateway is used to make data mobility and monitoring easier among IoT devices. Besides, [50] performs integration of MQTT protocol with REST architecture used in software development with the help of Programming Language (PL). In another paper [51], CoAP protocol is integrated with ISO/IEEE 11073 standard serving for Personal Health Devices (PHD) in the medical field. In above research studies, the use of HTTP and CoAP represents the transfer of data using Web technology. Furthermore, another contribution in [52] presents an integration of MQTT protocol with Distributed Tolerant Network (DTN) i.e. software. Proposed integration is achieved by implementing MQTT negative edge gateway which takes

TABLE 7. Summary for integration of IoT application layer protocols with hardware/software platforms.

Ref. No	CoAP	DDS	MQTT	HTTP	Hardware	Architecture	Standard	SW	Integration approach
[45]	×	✓	×	×	×	✓	×	×	Middleware
[46]	✓	×	×	×	×	×	✓	✓	Mapping
[47]	✓	×	×	×	×	×	✓	✓	Gateway
[48]	×	×	✓	×	×	×	×	✓	API
[49]	×	×	×	✓	×	×	×	✓	Gateway
[50]	×	×	✓	×	×	×	×	✓	PL
[51]	✓	×	×	×	×	×	✓	×	SE
[52]	×	×	✓	×	×	×	×	✓	Gateway
[53]	×	×	✓	×	✓	×	×	×	PL
[54]	✓	×	×	×	✓	×	×	×	Gateway
[55]	✓	×	✓	×	×	×	×	✓	BF
[56]	×	×	✓	×	×	×	×	✓	PM
[57]	×	×	✓	×	×	×	✓	×	DC
[58]	×	×	✓	×	×	×	✓	×	Proxy
[12]	×	×	✓	×	×	×	×	✓	Middleware

MQTT messages from broker and passed to the DTN bundle for further processing.

Not only IoT application layer protocols integrated with architecture, standard or software but hardware is also focused in two research studies [53] and [54]. Such that [54] integrates Field Programmable Gateway Array (FPGA) with CoAP server to enhance the performance of sensor nodes using a gateway. Input data is sent to the FPGA which is then passed to application layer level where CoAP is implemented for sending and receiving data.

Rest of the research studies deals with the heterogeneous IoT devices using the MQTT protocol. The contributions made by [55] is to integrate two application layer protocols MQTT and CoAP with Binding Function(BF). For this purpose, translation method makes incoming data capable of translating into either MQTT or CoAP depending upon the choice of IoT device. Moreover, another research study [56] integrates geolocation feature to MQTT client packets using Packet Modification (PM). In [57], MQTT is integrated with European Telecommunication Standard Institutes (ETSI) standard of diverse vertical application into one platform using Digital Certificates(DC) and in [58] with Robot Operating System (ROS) using the proxy. Furthermore, in [12], MQTT is made capable for monitoring medical devices using middleware. Hence, MQTT protocol can be a suitable choice to develop various IoT devices and ensures interoperability mostly through broker functionality.

It is analyzed from Table 7, that among fifteen (15) research studies, MQTT is integrated with hardware/software platforms by nine (9) studies, CoAP is integrated with others by six (6) out of fifteen (15) studies while only two research studies used DDS and HTTP protocol for integration. The reason to perform integration with MQTT and CoAP by large number of studies is due to lightweight and easy to use nature offered by these two protocols.

From the selected studies (34) it is analyzed that gateway approach is used by thirteen (13), SDN by one (1), middleware by another three (3), API by two (2) and other different integration approaches by fifteen (15) research studies. Gateway is used by maximum number of selected research studies to enhance interoperability. The reason is gateway as an intermediator at network layer comes with many advantages such as it ensures the data mobility, provides reliable data exchange and remote monitoring. However, the efforts are increased to deploy an additional device at network layer that is centralized in nature. Due to centralized approach of gateway, it is resource constrained and prone to attacks. Gateways are not scalable as when number of devices increases it becomes challenging for a gateway to send responses immediately.

As shown in Table 8 below, hardware platforms used in experiments/implementation are identified along with the reference of the research studies. Hardware platforms such as Raspberry Pi, Arduino and Beagle- bone are used to provide gateway services for sending and receiving the data.

Thirteen (13) research studies have used Raspberry Pi as a hardware platform for development because it uses fewer resources and cost-effective as compared to Beagle-bone and Arduino. Other hardware platforms are also mentioned in Table 8. Four research studies [33], [37], [44] and [49] have used Arduino as a micro-controller for mostly client services. When Arduino is used as client it selects and implement the lightweight protocol. Beagle-bone is used by another five research studies [10], [12], [32] and [39] as it is a powerful hardware platform. Among thirty-four (34) research studies, thirteen have not used any hardware platform, e.g., [30] and [36] have focused on architecture level solution for IoT where hardware related information is not required.

In Table 9, Linux OS is used widely in six (6) different research studies, followed by Windows, which is a target for three research studies. Linux is open source, scalable and secure with many distributions making it a suitable choice for the development in IIoT. The details of the rest of the OS are covered in Table 9. For example, Contiki is an operating system suitable for constrained and low power IoT devices used by one research study i.e. [32].

TABLE 8. Summary of hardware platforms used.

Sr.No	Hardware platform	References	Total
1	Arduino	[33] [37] [44] [49]	4
2	Beagle-bone	[10][12] [32] [39]	5
3	ESP 8266	[33] [37]	2
4	FPGA	[54]	1
5	MCU	[42]	1
6	NodeMCU	[48]	1
7	Raspberry pi	[10] [11] [28] [29] [32] [33] [34] [35][40] [43] [49[51][53]	13
8	Postman	[46]	1
9	Vital node box	[58]	1
10	Zolertia	[32]	1

TABLE 9. Summary of operating systems used.

Sr.No	Operating System	Availability	References	Total
1	Android	Open source	[33] [42] [51]	3
2	Contiki OS	Open source	[32]	2
3	iOS	Proprietary	[42]	1
4	Linux	Open source	[10] [12] [32] [37] [49][58]	6
5	Ubuntu	Open source	[33]	1
6	Xubuntu	Open source	[28] [29]	2
7	Windows	Proprietary	[11] [40] [42]	3

TABLE 10. Summary of framework proposed/used.

Sr.No	Framework	Developed/ Used	Availability	References	Total
1	Atlas	Proposed	Open source	[10]	1
2	Eclipse Californium	Used	Open source	[28] [29] [34]	3
3	Eclipse Vert.x	Used	Open source	[33]	1
4	Eclipse Kura	Used	Open source	[28] [29] [31]	3
5	Hybrid IoT communication	Proposed	–	[38]	1
6	Kryo	Used	Open source	[29]	1
7	Netty	Used	Open source	[37]	1
8	REST	Used	Open source	[50]	1

In Table 10, eight frameworks and their corresponding information are presented. The reason to use frameworks in IoT ensures the safe and reliable delivery of the IoT products such as sensors, devices or applications. Here, three (3) research studies [28], [29], and [34] have used Californium framework for CoAP. In [28], [29] and [31], Eclipse Kura framework for gateway-based integration is used. Due to open source nature of Eclipse Californium and Eclipse Kura, these two frameworks are extensively used in IoT. Other frameworks targeted by different research studies are shown in Table 10 with corresponding details.

D. TOOL ANALYSIS

It is also essential to analyze the tools used for IoT application layer protocols in thirty-four (34) research studies. The parameters of Table 11 are: 1) Tool name depicts the name of the tool while 2). Context represents the functionality of the tool such as client, server or broker. 3) Availability indicates either the selected tool is open source or proprietary 4) Relevant reference of the research study is provided against each tool.

As shown in Table 11, we have identified five (5) tools for MQTT protocol. In MQTT, the client must connect with a broker to mediate the communication between two devices. Eclipse Paho is one of the popular open- source client tools to facilitate MQTT publish and subscribe functionality available in different languages such as Java, C++ and Python. Eclipse Paho is used by [29], [32], [33] and [43] research studies. A broker is the heart of the MQTT protocol as clients do not connect directly with each other. The broker is necessary in MQTT as it is responsible for receiving messages from publishers, filtering these messages for subscribed clients and sending to subscribers. In short, broker act as a hub from where every message passes. Broker offers authentication and is extensible too. Mosquitto is an open-source plugin of Eclipse used by thirteen research studies to achieve broker functionality as shown in Table 11. The reason to use Mosquitto in large number is that it is lightweight and suitable for low power devices used in IoT. In some research studies, more than one broker is also used such as HiveMQ is used by [10], [11] and [43] research studies, because it offers open-source Java SDK plugins to facilitate modifications. Orion context broker is used by only one research study [35] and provides MQTT services. It is analyzed that all MQTT tools are available as open-source tools.

Few Tools for CoAP and XMPP are also listed used by selected research studies. In CoAP, messages are sent using client and server components. Therefore, LibCoAP act as a server providing services to the clients upon request. LibCoAP is used to implement the lightweight application layer protocols for constrained IoT devices. Two client tools in CoAP are CoAPThon and copper. Such as Copper tool is designed to add Web based features for clients. Moreover, the only tool found for XMPP in selected research studies is Ejabbered that is a defacto XMPP server and provide many features such as instant chat and real time communications to its users.

Some research studies have used other tools as well, for various purposes, e.g., development, testing or simulation as represented by the context. In Table 11, three (3) tools used for IoT development are: MATLAB [43], Node RED [40] and LambdaNative [12]. Wireshark tool is used to analyze the performance of application layer protocols (e.g. packet size, latency, power consumption) in three research studies. In this section, we have identified nineteen (19) tools categorized as MQTT tools, CoAP tools, XMPP tool, and other tools. It is evident from Table 11 that for MQTT, Mosquitto is a widely used broker and Eclipse Paho is extensively used as client. Finally, in CoAP, Copper is used by the highest number of studies that ensures the reliable transmission of messages between clients and server.

TABLE 11. **Tool anlaysis.**

Sr.No	Tool name	Context	Availability	Reference	Total
			MQTT Tools		
1	Eclipse Paho [60]	Client	Open source	[29] [32] [33] [43]	4
2	ActiveMQ	Broker	Open source	[11]	1
3	HiveMQ [61]	Broker	Open source	[10] [11] [43]	3
4	Mosquitto[62]	Broker	Open source	[11] [12] [28] [29] [30] [32] [35] [39] [50] [53] [56] [58]	12
5	Orion context	Broker	Open source	[35]	1
			CoAP Tools		
6	LibCoAP [63]	Server	Proprietary	[39]	1
7	CoAPthon	Client	Open source	[32] [33]	2
8	Copper	Client	Open source	[28] [46] [51]	3
			XMPP Tools		
9	Ejabbered	XMPP Server	Open source	[32]	1
			Other Tools		
10	OAuth 2.0 [65]	Authorization	Open source	[30]	1
11	Apache Jmeter [66]	Packet generator	Open source	[34] [49]	2
12	Eclipse [67]	Modeling	Open source	[46]	1
13	Mininet	Emulator	Open source	[52]	1
14	MATLAB [68]	Development	Proprietary	[43]	1
15	Node RED	Development	Open source	[40]	1
16	LambdaNative	Development	Open source	[12]	1
17	SPIN	Simulation	Open Source	[53]	1
18	Vivado	Synthesis	Open source	[54]	1
19	Wireshark[69]	Packet analyzer	Open source	[39] [46] [54]	3

E. EVALUATION PARAMETERS

Different evaluation parameters are used to evaluate/compare the performance of one application layer protocol with others in the same settings. In Table 12, category groups the similar evaluation parameters and their corresponding references. So, thirteen (13) evaluation parameters are grouped into five categories. First category is based on the hardware usage, groups: 1) CPU and Random-Access Memory (RAM) usage and 2) memory usage, evaluated by three (3) research studies. Second category is about the consumption of resources such as energy and power consumption which is evaluated by two research papers.

TABLE 12. **Summary of evaluation parameters.**

Sr. No	Categories	Evaluation Parameters	References	Total
1	Hardware usage	CPU and RAM usage	[29][32]	3
		Memory	[49]	
2	Consumption	Energy consumption	[10]	2
		Power consumption	[24]	
3	Time	Completion time	[10][11]	7
		Response time	[33][58]	
		Round trip time	[32][44][51]	
4	Verification	Deadlock	[53]	1
5	Packet	Packet Size	[35][56][39] [49][52][57]	13
		Number of packets	[51]	
		Packet lost and delay	[31][39] [52][54]	
		Overhead	[39][56]	

Third category is related to Time, which is an important parameter taken into consideration by seven research studies. It is divided into completion time, response time, round trip time and delay analyzed in the provided references. Fourth category is about the Verification of integrated protocol, which is examined by one research study for the deadlock property. Last category is Packet, considered as an evaluation parameter by thirteen research studies. Their related aspects are considered for data sending and receiving shown by six research studies in Table 12.

It is also important to discuss the limitations observed in the selected research studies. The number of research studies mentioning the data acquisition devices (e.g., sensors) and the use of programming languages for implementation is not discussed in this research study due to unavailable data. Such as in [30] and [42], TMP30 and MLX90621 temperature sensors are used respectively for data acquisition but many research studies did not explicitly mention data acquisition methods. Similarly, two research studies such as [28] and [38] have used Java and [32] used Python language for development. Overall, limited data is available for the support of programming language in Table 5, 6 and 7. Another critical limitation found in the selected research studies is the lack of a simulation environment/tools to evaluate integrated IoT application layer protocols.

V. ANSWERS TO RESEARCH QUESTIONS

In this section, answers of six Research Questions (RQ's) are provided in detail.

RQ1: What Is the Number of Studies Dealing With Interoperability for Industrial IoT, Where Following Integrations Are Performed: 1) IoT Application Layer Protocols; 2) IoT

Application Layer Protocols With Other Data Acquisition Protocols; 3) IoT Application Layer Protocols With Hardware/Software Platforms?:

Answer: We have identified a total of thirty-four (34) research studies where integration amongst IoT protocols is performed as per the inclusion and exclusion criteria (Section II). In Table 5, thirteen research studies performed the integration of IoT application layer protocols.

In Table 6, integration of IoT and industrial data acquisition protocols is carried out. Moreover, the integration of IoT application layer protocols with hardware/software platforms is presented in Table 7.

RQ2: What Are the Significant Application Layer Protocols Used for the Integration to Meet Data Interoperability Objectives in IIoT?:

Answer: In selected thirty-four (34) research studies, MQTT is integrated with the highest number of research studies such as in Table 5. MQTT and CoAP are integrated into eleven research studies. Five studies in Table 6 have used MQTT to integrate with industrial data acquisition protocols. MQTT with other hardware/software platforms is integrated by nine research studies as shown in Fig. 4.

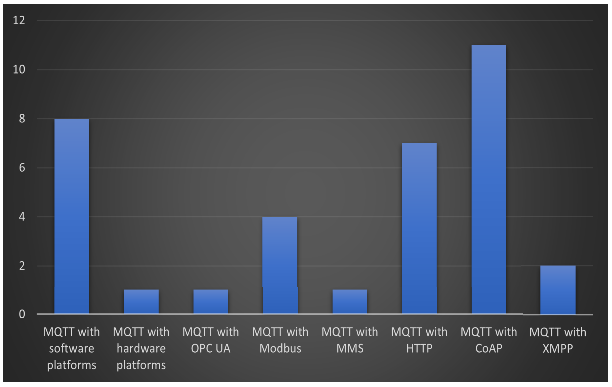

FIGURE 4. Summary of MQTT integration.

MQTT usage in higher number of research studies is due to its significance in Industrial IoT due to several advantages. Few advantages are listed below.

1) MQTT supports many to many communication as many clients can communicate with broker at one time. In industrial IoT, IoT devices are greater in number, MQTT helps an efficient communication between these devices.
2) MQTT is lightweight protocol suitable for the implementation of both heavily and constrained IoT devices such as smart watch, smart phone and smart bulb in industrial IoT.
3) MQTT is a scalable application layer protocol as it is asynchronous that decouples the client and broker in both time and space.
4) MQTT is a flexible protocol and can be applied to many real-world scenarios this is the reason its presence is found in three categories developed in Section II-A.

RQ3: What Are the Mostly Used Integration Approaches for IIoT and Why These Integration Approaches Are Used?:

Answer: The purpose of introducing these integration approaches is to enhance interoperability between IoT devices. These integration approaches play the role of intermediator either by using IoT architecture i.e., SDN or translator such as gateway. From Table 5, Table 6 and Table 7, it is analyzed that among thirty-four (34) research studies, thirteen (13) research papers have used gateway-based approach making it a leading integration approach. There are several advantages of gateways due to which it is widely used.

1) Gateway can connect IoT devices for a variety of application layer protocols that includes MQTT, CoAP and HTTP etc.
2) Gateway can perform automated translation of the incoming data from source IoT device into required format of target IoT device.
3) Gateway can also process and store data for local processing.

Other than the gateway-based approach, one research study has used SDN based approach, API based approach used by two research studies, three research papers performed middleware-based integration while other approaches such as broker extension or use of programming languages (i.e. Java) are also used by fifteen research studies as depicted in Fig.5.

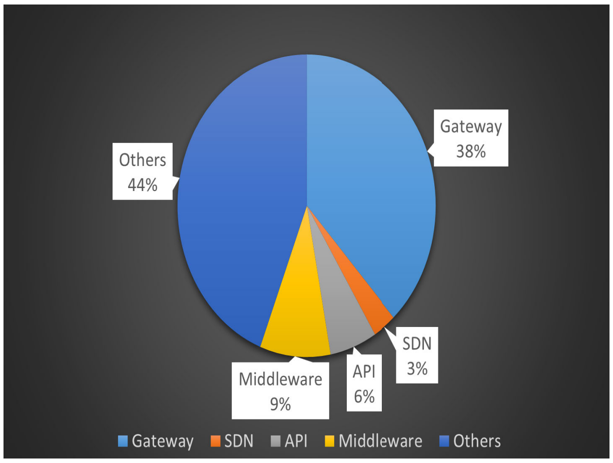

FIGURE 5. Summary of integration approaches used in selected research studies.

RQ4: What Are the Leading Tools to Carryout Protocol Integration for IIoT?:

Answer: Tool analysis is provided in Table 11 in which five MQTT tools, three CoAP tools, one XMPP tool, and other ten tools are identified. MQTT can be integrated with other IoT application layer protocols, data acquisition protocols or other hardware/software platforms in twenty-six (26) out of thirty-four (34). Among twenty-six (26), MQTT tools are mentioned by fifteen (15) research studies. CoAP tools are used by seven (7) number of research studies. XMPP tool is used by one research paper and HTTP tool is also used by one research study. Moreover, eleven research studies used other tools for different purposes i.e. implementation, evaluation or verification.

RQ5: What Are the Important Parameters to Evaluate the Performance as a Result of Integrated IoT Application Layer Protocols?:

Answer: There is a wide range of evaluation parameters such as: packet size, overhead, round trip time and energy consumption, etc., to evaluate the performance of IoT application layer protocols after integration. From Table 12, it is evident that thirteen (13) research studies have focused on packet evaluation parameters including packet size, number of packets, overhead, packet loss and delay, indicating its importance and popularity in IoT interoperability. For example, when heterogeneous IoT devices are made interoperable via integration approach, it is important to ensure that packet size do not become large and overhead is not increased. If small packet size and low overhead is achieved, seamless transmission of the data is made possible.

RQ6: What Are the Challenges and Possible Future Directions Regarding the Integration of Protocols in IIoT?:

Answer: Researchers are putting effort to minimize the risk of non-interoperability, but it has laid the foundation to several challenges in industrial IoT as well which are listed below.

A. SCALABILITY

From Table 13, it can be analyzed that among thirty-four (34) research studies, eleven (11) research studies have integrated only two protocols while other seven (7) research studies integrated three protocols. Only one (1) out of thirty-four (34) research studies performed the integration of four protocols as shown in Table 13. Integration of more than four protocols is not observed in any selected research study. Providing end-user with more options for integrating number of protocols is one of the challenging tasks in industrial IoT [8]. As long as, intermediator such as gateway, server, broker or middleware are used for protocol conversion therefore, higher scalability is difficult to achieve. To overcome this challenge, the dependency on intermediator needs to be removed so that, devices themselves can support more than one protocol and talk to each other.

TABLE 13. Summary of scalability in selected research studies.

No. of Protocols	References	Total
Integration of two protocols	[11] [28] [29] [31] [35] [37] [39] [40] [41] [42][44]	11
Integration of three protocols	[10] [30] [33] [34] [36] [38] [43]	7
Integration of four protocols	[32]	1
Integration of five protocols	–	0

B. LACK OF SIMULATION ENVIRONMENT

Industrial IoT is still in its infancy to achieve data interoperability. Heterogeneous industrial devices using diverse protocols is another integration challenge in industry 4.0. It is analyzed from Table 6 that a very small number of research papers focused on the integration of IoT and data acquisition protocols. To simplify the integration of IoT application layer protocols with other industry standards, a scalable solution for information exchange to provide rich IoT services is needed [33]. Also, for industrial IoT, more interoperability for small and medium-size manufacturing companies is required, to exchange data with more desired features; such as, limited hardware capabilities, energy efficiency, real-time performance, low cost, enhanced security/privacy and reliable communication. For this purpose, open-source tool support is required; for such development and simulation.

For example in IoT, simulation can be carried out using various tools, some of the well-known tools are mentioned in the Table 14. Parameters of Table 14 include 1) Tool name 2) License type indicating whether its open source or commercial tool. 3) Modeling support represents the Graphical User Interface (GUI) support to design IoT scenario. 4) Programming Language support is listed so that open source code can be customized. 5) Platform support for each simulation tool is also indicated in Table 14. 6) Support of three protocols i.e. Routing protocols, transport layer protocols and application layer protocols are also listed.

From Table 14, it is analyzed that total five tools are mentioned where three tools are open source and rest of the tools are commercial. NetSim and Simple IoT Simulator are commercial tools. NetSim, NS3 and OMNET++ provides the modeling view of the IoT scenario. Besides, support for programming language, platform, routing protocols and transport layer protocols for each tool is also mentioned in Table 14. Moreover, the significant analysis of application layer protocols concludes that existing IoT Simulation tools support only limited number of application layer protocols to perform experiments.

C. HIGH DEPENDENCY ON MQTT PROTOCOL

MQTT is one of the widely used application layer protocol for sending and receiving messages in IoT. It is analyzed that among thirty-four (34) research studies, twenty- six (26) research papers have used MQTT for integration. However, some other limitations of MQTT are also observed. One of the limitations is the unspecified data format, every client is free to use whatever data format is required. If the publisher and subscriber are not agreed on the data format, they are not able to communicate with each other. On the other hand, brokers of MQTT are not intelligently routing the messages, as it passes the message what it gets. It cannot distinguish the new or already transmitted messages thus, consuming extra bandwidth. Therefore, it can be argued that dependency on the most common protocol i.e. MQTT with its current limitations is challenging for large-scale applications in industrial IoT [28].

Industrial IoT is diverse and there is room for other specialized protocols, as well. None of the selected research studies have used Advance Messaging Queuing Protocol (AMQP) for integration. Despite of the fact that like MQTT, AMQP

TABLE 14. Summary of IoT simulation tools.

Sr.No	Tool name	License type	Modeling support	Programming Language support	Platform support	Routing protocols	Transport layer protocols	Application layer protocols
1	NetSim[70]	Commercial	Yes	C, Python	Windows, Linux	AODV, RPL	TCP, UDP	CoAP, HTTP
2	Cooja [71]	Open source	No	C,C+, Java	Contiki OS	RPL	TCP, UDP	–
3	NS3 [72]	Open source	Yes	C++, Python	Ubuntu	RPL	TCP, UDP	HTTP
4	OMNET++ [73]	Open source	Yes	C++	Windows, Linux, Free BSD	AODV, DSR	TCP	HTTP
5	Simple IoT Simulator[74]	Commercial	No	–	Ubuntu, CentOS, Linux	–	TCP	CoAP, HTTP, MQTT, SNMP

is also based on publish-subscribe pattern. AMQP is also reliable, secure [18], scalable and supporting multi hosted infrastructure [75]. Similarly, in data acquisition, Modbus is frequently used in selected research studies. However, another well-known data acquisition protocols such as Profibus has desirable features but it is not selected by any research study which can be targeted in the future for Industrial IoT.

D. SECURITY

Application layer protocols for Industrial IoT are either supporting client-server or publish- subscribe architectures for data sending and receiving. However, by relying on these architectural models, all communication is routed using a central server or broker, which if not protected can introduce security vulnerabilities for users and IoT devices [8]. However, to perform integration of application layer protocols, different security measures are important to ensure such as device authentication or access control for IoT devices.

E. DIVERSE INTERACTION MODES FOR REPRESENTATIONS AND INTERACTIONS OF THINGS

As heterogeneity comes with a different representation of things, it becomes difficult to represent the things using common semantics, i.e. ontology or shared schema. For example, MQTT supports binary format while HTTP supports textual format for sending and receiving messages. If a device with HTTP protocol wants to send data to MQTT, the message is translated using any intermediator device and ensures semantic interoperability. Also, for the heterogeneous devices in industrial IoT, modes of interactions (e.g. search, find and access) can be different. This leads to syntactic and semantic interoperability issues for IIoT devices [8].

F. DIVERSE GATEWAY SOLUTIONS

It is easy to deploy the network layer based solutions such as gateway to enhance interoperability as nodes join the network and start sending and receiving messages. However, it is difficult to perform data communication depending on application layer only in IIoT devices. It is analyzed from selected research studies that various IoT gateways are deployed as shown in Table 15. The purpose to introduce multiple gateways depends on the requirement of each research

TABLE 15. Summary of gateway-based approaches.

Ref.	Name	Deployment	Centralized	Open source
[10]	Translator gateway	Cloud/ thing	✓	–
[11]	–	Server	✓	–
[28]	Kura extended gateway	Cloud	✓	✓
[29]	Kura extended gateway	–	✓	✓
[30]	Semantic gateway as service	Cloud	✓	✓
[32]	Secure Multi-protocol Integration Bridge	Server	✓	–
[34]	Application layer gateway	Raspberry pi	✓	–
[35]	IoT gateway	Cloud	✓	–
[43]	–	Microgrid	✓	×
[47]	ISO/IEEE 11073 translator gateway	FHIR server	✓	✓
[49]	–	Cloud	✓	–
[52]	MQTT near-user edge gateway	Edge	×	–
[54]	–	Server	✓	–

study such as deployment requirement, nature of the supported architecture i.e. centralized or distributed to perform the integration. There is no standard gateway available that can convert/translate one protocol into another or integrate all IoT application layer protocols. It is also analyzed from Table 15 that most gateways are deployed on cloud, server, edge, or some other nodes.

Few drawbacks of gateway based solution are discussed here. 1) Gateway is comprises of centralized architecture which leads to another issue i.e. single point of failure. 2) Besides, if more number of protocols are added in the centralized intermediator, load balancing becomes a problem and performance can be degraded. Therefore, to overcome these limitations, a gateway-less solution named as "Devices as intermediator" is required which can systematically integrate application layer protocols at device level in industrial IoT as shown in Fig. 6.

Devices as intermediator approach is possible future direction to resolve diverse gateway problem that occurs between IoT devices using different application layer protocols. When different IoT devices plays the role of intermediator then single point of failure is no longer an issue. In addition, due to distributed nature of IoT devices, the issue of load balancing does not occur. Let's consider an example shown in Figure 6, if a smart car supporting MQTT protocol want to talk with smart bulb using CoAP protocol, then nearby

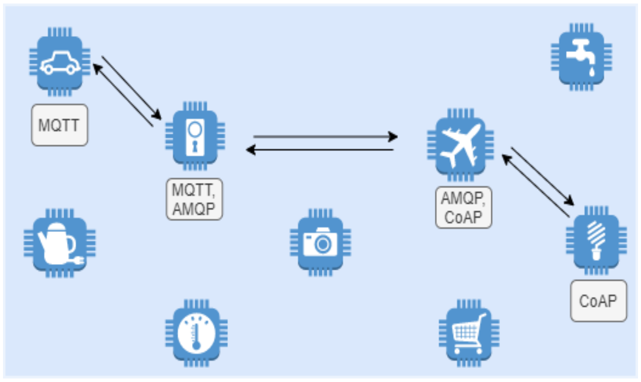

FIGURE 6. Devices as intermediator approach.

devices act as intermediator to enable the communication between heterogeneous devices. The proposed solution comprises of different steps as discussed below.

1) *Identification of Compatible Devices:* In this step, source device identify the nearby devices which are supporting more than one application layer protocols. It is assumed that few devices must support more than one application layer protocols to act as a bridge. This step can be achieved using PING message sent from source device to all nearby devices.
2) *Discovery of the Shortest Path:* When the nearby devices with two protocols are identified then possible routes from source to destination can be found. Among these routes, shortest path is selected using Dijkstra algorithm.
3) *Message Transmission:* After selecting the shortest path, the data from source to target device is sent successfully using two different application layer protocols.

We believe, devices as intermediator approach is the promising solution to eradicate the dependency on a translator/intermediator or gateway between heterogeneous devices. This solution does not face single point of failure and load balancing issues as IoT devices are of distributed nature.

VI. CONCLUSION AND FUTURE WORK

This article comprehensively investigates IoT application layer protocols used for integration to support industrial IoT requirements. This SLR finally, filters down thirty-four (34) research studies published during 2014-2020. These research studies are further divided into three categories i.e. 1) integration of IoT application layer protocols (13), 2) integration of IoT and data acquisition protocols (6) and 3) integration with hardware/software platforms (15). Each category is thoroughly analyzed to answer the developed six research questions. Moreover, ten (10) hardware platforms, seven (7) operating systems and eight (8) frameworks are also identified. Finally, nineteen (19) tools are extracted among 34 research studies, which made the integration possible on the practical basis.

It is concluded that integration of IoT application layer protocols is extensively supported using any intermediator such as gateway, API or middleware. Although these intermediators enable the communication between IoT devices but leads to several other issues such as a single point of failure, load balancing and scalability.

Therefore, it is highly important to integrate industrial data acquisition and IoT application layer protocols without depending on any intermediator to facilitate industry 4.0. In this context, practitioners/researchers can develop the Devices as intermediator approach to 1) identify the intermediator devices, 2) find shortest path and 3) enable message transmission as discussed in this SLR. By doing so, scalability can be achieved and dependency on diverse gateways for different application layer protocols can be eradicated.

REFERENCES

[1] J. Wan, S. Tang, Z. Shu, D. Li, S. Wang, M. Imran, and A. V. Vasilakos, "Software-defined industrial Internet of Things in the context of industry 4.0," *IEEE Sensors J.*, vol. 16, no. 20, pp. 7373–7380, May 2016.

[2] C. Zhu, J. J. P. C. Rodrigues, V. C. M. Leung, L. Shu, and L. T. Yang, "Trust-based communication for the industrial Internet of Things," *IEEE Commun. Mag.*, vol. 56, no. 2, pp. 16–22, Feb. 2018.

[3] A. Wasicek, "The future of 5G smart home network security is micro-segmentation," *Netw. Secur.*, vol. 11, pp. 11–13, Nov. 2020.

[4] M. Noura, M. Atiquzzaman, and M. Gaedke, *Interoperability in Internet of Things Infrastructure: Classification, Challenges, and Future Work* (Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering), vol. 246. Cham, Switzerland: Springer, Jan. 2018, pp. 11–18.

[5] V. R. Konduru and M. R. Bharamagoudra, "Challenges and solutions of interoperability on IoT," Tech. Rep., 2017, pp. 1–5.

[6] L. Babun, K. Denney, Z. B. Celik, P. McDaniel, and A. S. Uluagac, "A survey on IoT platforms: Communication, security, and privacy perspectives," *Comput. Netw.*, vol. 192, 2021, Art. no. 108040, doi: 10.1016/j.comnet.2021.108040.

[7] H. Rahman and M. I. Hussain, "A comprehensive survey on semantic interoperability for Internet of Things: State-of-the-art and research challenges," *Trans. Emerg. Telecommun. Technol.*, vol. 31, no. 12, p. e3902, 2020.

[8] B. Rana, Y. Singh, and P. K. Singh, "A systematic survey on Internet of Things: Energy efficiency and interoperability perspective," *Trans. Emerg. Telecommun. Technol.*, early access, pp. 1–41, Dec. 2020, doi: 10.1002/ett.4166.

[9] A. Zyane, M. N. Bahiri, and A. Ghammaz, "IoTScal-H: Hybrid monitoring solution based on cloud computing for autonomic middleware-level scalability management within IoT systems and different SLA traffic requirements," *Int. J. Commun. Syst.*, vol. 33, no. 14, p. e4495, Sep. 2020.

[10] A. E. Khaled and S. Helal, "Interoperable communication framework for bridging RESTful and topic-based communication in IoT," *Futur. Gener. Comput. Syst.*, vol. 92, pp. 628–643, Mar. 2019.

[11] G. Corotinschi and V. G. Gaitan, "Enabling IoT connectivity for modbus networks by using IoT edge gateways," in *Proc. Int. Conf. Develop. Appl. Syst. (DAS)*, May 2018, pp. 175–179.

[12] M. Görges, G. A. Dumont, C. L. Petersen, and J. M. Ansermino, "Using machine-to-machine/'Internet of Things' communication to simplify medical device information exchange," in *Proc. Int. Conf. Internet Things (IOT)*, 2014, pp. 49–54.

[13] S. A. Olamidipupo and K. Danas, "Review of interoperability techniques in data acquisition of wireless ECG devices," *IOSR J. Mob. Comput. Appl.*, vol. 2, no. 2, pp. 42–2394, 2015.

[14] A. Mavrogiorgou, A. Kiourtis, K. Perakis, S. Pitsios, and D. Kyriazis, "IoT in healthcare: Achieving interoperability of high-quality data acquired by IoT medical devices," *Sensors*, vol. 19, no. 9, p. 1978, 2019.

[15] S. Bansal and D. Kumar, "IoT ecosystem: A survey on devices, gateways, operating systems, middleware and communication," in *International Journal of Wireless Information Networks*, vol. 27, no. 3. Cham, Switzerland: Springer, Sep. 2020, pp. 340–364.

[16] W. Kassab and K. A. Darabkh, "A–Z survey of Internet of Things: Architectures, protocols, applications, recent advances, future directions and recommendations," *J. Netw. Comput. Appl.*, vol. 163, Aug. 2020, Art. no. 102663.

[17] E. de Matos, R. T. Tiburski, C. R. Moratelli, S. J. Filho, L. A. Amaral, G. Ramachandran, B. Krishnamachari, and F. Hessel, "Context information sharing for the Internet of Things: A survey," *Comput. Netw.*, vol. 166, Jan. 2020, Art. no. 106988.

[18] N. Q. Uy and V. H. Nam, "A comparison of AMQP and MQTT protocols for Internet of Things," in *Proc. 6th NAFOSTED Conf. Inf. Comput. Sci. (NICS)*, Dec. 2019, pp. 292–297.

[19] G. S. Gaba, G. Kumar, T.-H. Kim, H. Monga, and P. Kumar, "Secure device-to-device communications for 5G enabled Internet of Things applications," *Comput. Commun.*, vol. 169, pp. 114–128, Mar. 2021.

[20] M. Iglesias-Urkia, A. Orive, A. Urbieta, and D. Casado-Mansilla, "Analysis of CoAP implementations for industrial Internet of Things: A survey," *J. Ambient Intell. Humanized Comput.*, vol. 10, no. 7, pp. 2505–2518, Jul. 2019.

[21] X. Liu, T. Zhang, N. Hu, P. Zhang, and Y. Zhang, "The method of Internet of Things access and network communication based on MQTT," *Comput. Commun.*, vol. 153, pp. 169–176, Jan. 2020.

[22] U. Hunkeler, H. L. Truong, and A. Stanford-Clark, "MQTT-S—A publish/subscribe protocol for wireless sensor networks," in *Proc. 3rd IEEE Int. Conf. Commun. Syst. Softw. Middleware Workshops (COMSWARE)*, Jan. 2008, pp. 791–798.

[23] E. Ferrera, D. Conzon, P. Brizzi, R. Rossini, C. Pastrone, M. Jentsch, P. Kool, C. Kamienski, and D. Sadok, "XMPP-based infrastructure for IoT network management and rapid services and applications development," *Ann. des Telecommun. Telecommun.*, vol. 72, nos. 7–8, pp. 443–457, 2017.

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

[25] B. Kitchenham, "Procedures for performing systematic literature reviews," Keele Univ., Keele, U.K., Tech. Rep. TR/SE-0401 NICTA TR-0400011T.1, 2004, p. 33.

[26] A. Abane, P. Muhlethaler, M. Daoui, and H. Afifi, "A down-to-earth integration of named data networking in the real-world IoT," in *Proc. 6th Int. Conf. Future Internet Things Cloud Workshops (FiCloudW)*, Aug. 2018, pp. 243–249.

[27] Z. Sheng, D. Tian, and V. C. M. Leung, "Toward an energy and resource efficient Internet of Things: A design principle combining computation, communications, and protocols," *IEEE Commun. Mag.*, vol. 56, no. 7, pp. 89–95, Jul. 2018.

[28] P. Bellavista and A. Zanni, "Towards better scalability for IoT-cloud interactions via combined exploitation of MQTT and CoAP," in *Proc. IEEE 2nd Int. Forum Res. Technol. Soc. Ind. Leveraging Better Tomorrow (RTSI)*, Sep. 2016, pp. 1–6.

[29] P. Bellavista and A. Zanni, "Scalability of Kura-extended gateways via MQTT-CoAP integration and hierarchical optimizations," in *Proc. 11th Int. Conf. Body Area Netw.*, 2017, pp. 210–216.

[30] P. Desai, A. Sheth, and P. Anantharam, "Semantic gateway as a service architecture for IoT interoperability," in *Proc. IEEE Int. Conf. Mobile Services*, Jun. 2015, pp. 313–319.

[31] M. Dave, J. Doshi, and H. Arolkar, "MQTT-CoAP interconnector: IoT interoperability solution for application layer protocols," in *Proc. 4th Int. Conf. I-SMAC (IoT Social, Mobile, Anal. Cloud) (I-SMAC)*, Oct. 2020, pp. 122–127.

[32] E. Palavras, K. Fysarakis, I. Papaefstathiou, and I. Askoxylakis, "SeMIBIoT: Secure multi-protocol integration bridge for the IoT," in *Proc. IEEE Int. Conf. Commun. (ICC)*, May 2018, pp. 1–7.

[33] V. A. Barros, S. A. B. Junior, S. M. Bruschi, F. J. Monaco, and J. C. Estrella, "An IoT multi-protocol strategy for the interoperability of distinct communication protocols applied to Web of things," in *Proc. 25th Brazillian Symp. Multimedia Web*, 2019, pp. 81–88.

[34] M. A. A. Da Cruz, J. J. P. C. Rodrigues, E. S. Paradello, P. Lorenz, P. Solic, and V. H. C. Albuquerque, "A proposal for bridging the message queuing telemetry transport protocol to HTTP on IoT solutions," in *Proc. 3rd Int. Conf. Smart Sustain. Technol. (SpliTech)*, 2018, pp. 1–5.

[35] P. Peniak and M. Franekova, "Model of integration of embedded systems via CoAP protocol of Internet of Things," in *Proc. Int. Conf. Appl. Electron. (AE)*, Sep. 2016, pp. 201–204.

[36] D. Petrova-Antonova, G. Andreev, and S. Ilieva, "Unified connectivity of IoT devices through abstraction of application protocols," in *Proc. 7th Int. Conf. Inf. Commun. Manage. (ICICM)*, vol. F1312, 2017, pp. 56–61.

[37] M. Gabbrielli, S. Giallorenzo, I. Lanese, and S. P. Zingaro, *Linguistic Abstractions for Interoperability of IoT Platforms*, vol. 347. Cham, Switzerland: Springer, 2019.

[38] C.-H. Lee, Y.-W. Chang, C.-C. Chuang, and Y. H. Lai, "Interoperability enhancement for Internet of Things protocols based on software-defined network," in *Proc. IEEE 5th Global Conf. Consum. Electron.*, Oct. 2016, pp. 4–5.

[39] D. Thangavel, X. Ma, A. Valera, H.-X. Tan, and C. K.-Y. Tan, "Performance evaluation of MQTT and CoAP via a common middleware," in *Proc. IEEE 9th Int. Conf. Intell. Sensors, Sensor Netw. Inf. Process. (ISSNIP)*, Apr. 2014, pp. 1–6.

[40] M. Tabaa, B. Chouri, S. Saadaoui, and K. Alami, "Industrial communication based on modbus and node-RED," *Procedia Comput. Sci.*, vol. 130, pp. 583–588, Jan. 2018.

[41] J. Pfrommer, S. Gruner, and F. Palm, "Hybrid OPC UA and DDS: Combining architectural styles for the industrial Internet," *IEEE Int. Work. Fact. Commun. Syst. (WFCS)*, May 2016, pp. 1–7.

[42] L. O. Figueiredo, L. C. M. Maia, M. T. Rocha, J. A. N. Barbosa, Jr., A. P. V. A. Aguiar, R. B. C. Lima, G. A. Junior, and P. R. Barros, "Thermal vision for remote monitoring through cross-platform application," in *Proc. 13th IEEE Int. Conf. Ind. Appl. (INDUSCON)*, Nov. 2018, pp. 698–703.

[43] R. J. Howlett, *Sustainability in Energy and Buildings*. Berlin, Germany: Springer-Verlag, 2009.

[44] D.-H. Kim, H.-Y. Lee, and D.-S. Kim, "Enhanced industrial message protocol for real-time IoT platform," in *Proc. Int. Conf. Electron., Inf., Commun. (ICEIC)*, Jan. 2018, pp. 1–2.

[45] A. Hakiri, P. Berthou, A. Gokhale, and S. Abdellatif, "Publish/subscribe-enabled software defined networking for efficient and scalable IoT communications," *IEEE Commun. Mag.*, vol. 53, no. 9, pp. 48–54, Sep. 2015.

[46] M. Iglesias-Urkia, D. Casado-Mansilla, S. Mayer, J. Bilbao, and A. Urbieta, "Integrating electrical substations within the IoT using IEC 61850, CoAP, and CBOR," *IEEE Internet Things J.*, vol. 6, no. 5, pp. 7437–7449, Oct. 2019.

[47] W. Li and J. Park, "Design and implementation of integration architecture of ISO 11073 DIM with FHIR resources using CoAP," in *Proc. Int. Conf. Inf. Commun. (ICIC)*, Jun. 2017, pp. 268–273.

[48] N. Imtiaz Jaya and M. F. Hossain, "A prototype air flow control system for home automation using MQTT over websocket in AWS IoT core," in *Proc. Int. Conf. Cyber-Enabled Distrib. Comput. Knowl. Discovery (CyberC)*, Oct. 2018, pp. 111–117.

[49] B. Negash, T. Westerlund, and H. Tenhunen, "Towards an interoperable Internet of Things through a Web of virtual things at the fog layer," *Future Gener. Comput. Syst.*, vol. 91, pp. 96–107, Feb. 2019.

[50] I. Garrigós and M. Wimmer, *Current Trends in Web Engineering*, vol. 1, I. Garrigós and M. Wimmer, Eds. Cham, Switzerland: Springer, 2018, pp. 156–165.

[51] B. Oryema, H.-S. Kim, W. Li, and J. T. Park, "Design and implementation of an interoperable messaging system for IoT healthcare services," in *Proc. 14th IEEE Annu. Consum. Commun. Netw. Conf. (CCNC)*, Jan. 2017, pp. 45–51.

[52] P. Manzoni, E. Hernández-Orallo, C. T. Calafate, and J. C. Cano, "A proposal for a publish/subscribe, disruption tolerant content island for fog computing," in *Proc. 3rd Work. Exp. Des. Implement. Smart Objects, Co-Located With MobiCom (SMARTOBJECTS)*, 2017, pp. 47–52.

[53] S. Chouali, A. Boukerche, and A. Mostefaoui, "Towards a formal analysis of MQTT protocol in the context of communicating vehicles," in *Proc. 15th ACM Int. Symp. Mobility Manage. Wireless Access*, Nov. 2017, pp. 129–136.

[54] R. Tsugami and Y. Shibata, "FPGA implementation of lightweight communication protocol processing for IoT," *Adv. Intell. Syst. Comput.*, vol. 772, pp. 506–517, Jul. 2019.

[55] A. Epishkina, M. Finoshin, and K. Kogos, *Information Science and Applications (ICISA) 2016* (Lecture Notes in Electrical Engineering), vol. 376. 2016, pp. 641–650.

[56] R. Bryce, T. Shaw, and G. Srivastava, "MQTT-G: A publish/subscribe protocol with geolocation," in *Proc. 41st Int. Conf. Telecommun. Signal Process. (TSP)*, Jul. 2018, pp. 1–4.

[57] H. W. Chen and F. J. Lin, "Converging MQTT resources in ETSI standards based M2M platform," in *Proc. IEEE Int. Conf. Internet Things (iThings), IEEE Green Comput. Commun. (GreenCom), IEEE Cyber, Phys. Social Comput. (CPSCom)*, Sep. 2014, pp. 292–295.

[58] M. Mukhandi, D. Portugal, S. Pereira, and M. S. Couceiro, "A novel solution for securing robot communications based on the MQTT protocol and ROS," in *Proc. IEEE/SICE Int. Symp. Syst. Integr. (SII)*, Jan. 2019, pp. 608–613.
[59] M. R. Palattella, R. Soua, A. Stemper, and T. Engel, "Aggregation of MQTT topics over integrated satellite-terrestrial networks," *ACM SIGMETRICS Perform. Eval. Rev.*, vol. 46, no. 3, pp. 96–97, Jan. 2019.
[60] *Eclipse Paho*. Accessed: Jun. 7, 2020. [Online]. Available: https://www.eclipse.org/paho/
[61] *HiveMQ*. Accessed: Jun. 7, 2020. [Online]. Available: https://www.hivemq.com/
[62] *Mosquitto*. Accessed: Jun. 7, 2020. [Online]. Available: https://mosquitto.org/
[63] *LibCoAP*. Accessed: Jun. 7, 2020. [Online]. Available: https://libcoap.net/
[64] *Ejabbered*. Accessed: Jun. 7, 2020. [Online]. Available: https://www.ejabberd.im/
[65] *0Auth*. Accessed: Jun. 7, 2020. [Online]. Available: https://www.oauth.com/
[66] *Apache Jmeter*. Accessed: Jun. 7, 2020. [Online]. Available: https://jmeter.apache.org/
[67] *Eclipse*. Accessed: Jun. 7, 2020. [Online]. Available: https://www.eclipse.org/
[68] *MATLAB*. Accessed: Jun. 7, 2020. [Online]. Available: https://www.mathworks.com/products/ matlab.html
[69] *Wireshark*. Accessed: Jun. 7, 2020. [Online]. Available: https://www.wireshark.org/
[70] *NetSim*. Accessed: Jun. 7, 2020. [Online]. Available: https://www.tetcos.com/
[71] *Cooja*. Accessed: Jun. 7, 2020. [Online]. Available: https://github.com/contiki-os/contiki/wiki/An-Introduction-to-Cooja
[72] *NS3*. Accessed: Jun. 7, 2020. [Online]. Available: https://www.nsnam.org/
[73] *OMNET++*. Accessed: Jun. 7, 2020. [Online]. Available: https://omnetpp.org/
[74] *Simple IoT Simulator*. Accessed: Jun. 7, 2020. [Online]. Available: https://www.simplesoft.com/SimpleIoTSimulator.html
[75] J. E. Luzuriaga, M. Perez, P. Boronat, J. C. Cano, C. Calafate, and P. Manzoni, "A comparative evaluation of AMQP and MQTT protocols over unstable and mobile networks," in *Proc. 12th Annu. IEEE Consum. Commun. Netw. Conf. (CCNC)*, Jan. 2015, pp. 931–936.

ANAM AMJAD received the B.S. degree in computer sciences from Fatima Jinnah Women University, Pakistan, and the M.S. degree in software engineering from the National University of Sciences and Technology (NUST), Pakistan, where she is currently pursuing the Ph.D. degree with the Department of Computer and Software Engineering, CEME. Her research interests include business process automation through model driven software engineering and communication protocols in the Internet of Things (IoT).

FAROOQUE AZAM is currently a Key Faculty Member of the Department of Computer Software Engineering, EME College, NUST, Pakistan. He has been involved in post graduate teaching and research, since 2007. He has 138 international journal and conference publications on his credit, in April 2020. His areas of research interests include model driven software engineering, model driven testing, model driven embedded applications, model driven web engineering, software design, and architectures. He is a regular member of evaluation committees of Pakistan Engineering Council (PEC) and Higher Education Commission's Technological Development Funding (HEC-TDF).

MUHAMMAD WASEEM ANWAR is currently pursuing the Ph.D. degree with the Department of Computer and Software Engineering, CEME, National University of Sciences and Technology, Pakistan. He is currently a senior researcher and an industry practitioner in the field of model based system engineering (MBSE) for embedded and control systems. His major research interest includes model based system engineering (MBSE) for complex and large systems.

WASI HAIDER BUTT is currently an Assistant Professor with the Department of Computer and Software Engineering, College of Electrical and Mechanical Engineering, National University of Sciences and Technology, Pakistan. His areas of interests include model driven software engineering and Web development and requirement engineering.