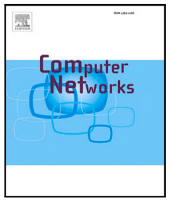
Computer Networks 225 (2023) 109640

Contents lists available at ScienceDirect

Computer Networks

journal homepage: www.elsevier.com/locate/comnet



Use of QUIC for AMQP in IoT networks

Faheem Iqbal [a], Moneeb Gohar [a,*], Hanen Karamti [b], Walid Karamti [c,d], Seok-Joo Koh [e], Jin-Ghoo Choi [f]

[a] Department of Computer Science, Bahria University, Islamabad, Pakistan
[b] Department of Computer Sciences, College of Computer and Information Sciences, Princess Nourah Bint Abdulrahman University, P.O. Box 84428, Riyadh 11671, Saudi Arabia
[c] Department of Computer Science, College of Computer, Qassim University, Buraydah 51452, Saudi Arabia
[d] Data Engineering and Semantics Research Unit, Faculty of Sciences of Sfax, University of Sfax, Tunisia
[e] School of Computer Science and Engineering, Kyungpook National University, South Korea
[f] Department of Information and Communication Engineering, Yeungnam University, South Korea

ARTICLE INFO

Keywords:
QUIC
AMQP
AMQP 1.0
Internet of Things (IoT)
Performance analysis
Docker
TCP
Energy analysis
Network layer

ABSTRACT

The use of IoT devices is expanding every day in today's environment. An interoperable protocol like AMQP is essential for supporting multiple IoT use cases and interconnecting IoT devices from different vendors. Many IoT applications are sensitive to delays, which researchers are working to avoid as much as possible. One of the main sources of the delay is the underlying transport layer protocol, such as TCP or UDP. TCP is more reliable than UDP, although it is slower due to the three-way handshake and the use of TLS for security. QUIC, a new transport layer protocol standardized by the Internet Engineering Task Force, combines the finest aspects of UDP and TCP to provide quick and reliable communication. We use the Go programming language to integrate AMQP 1.0 with QUIC to reduce latency and improve battery life. The Docker tool is used to containerize the AMQP 1.0 Broker, Sender, and Receiver implementations, and various scenarios are tested in the NS3 simulator. The findings demonstrated that even though Round Trip Time was 71% higher for QUIC, using QUIC at the transport level improved Startup Latency and Total Communication Time by 62% and 22%, respectively. The proposed scheme (AMQP 1.0 over QUIC) transported 3.5 times more data than the existing scheme (AMQP 1.0 over TCP), but QUIC's throughput was 7 times higher, which shorten the communication time and 31% less energy was consumed. Furthermore, metrics including Packet Loss, Delay, and Channel Bandwidth were used to compare the two schemes. The results showed that, with the exception of the low channel bandwidth scenario, the proposed scheme consistently outperformed the existing scheme.

1. Introduction

The Internet of Things (IoT) refers to objects that are connected to the Internet. IoTs do not have a single universal or standard definition. These gadgets have an embedded system that collects data, processes it, and sends it to other devices. The communication with the Internet is the main distinction between IoTs and wireless sensor networks. Wireless sensor networks are not directly connected to the Internet, whereas IoT devices are. Instead, the sensors are connected to the central node, also known as the Cluster Head, which is then connected to the Internet. The goal of the Internet of Things is to connect relatively inept items, like as temperature sensors, to the Internet. Mostly IoTs have limited resources in terms of storage, computation, battery, etc., [1] provides a more comprehensive definition of IoT, encompassing both physical and virtual things which can be identified and connected to the Internet. IoTs have many use cases like smart cities, smart grids, telemetry, smart manufacturing, smart agriculture, etc. These devices can be densely deployed and are extraordinarily numerous, numbering in the billions. According to a study carried by CISCO [2] number of devices connected to IP are expected to reach 29.3 billion up from 18.4 billion in 2018, which will be more than three folds of global human population.

ITU-T Y-2060 recommendations [1] identifies four fundamental characteristics of IoT devices which are: interconnectivity, enormous scale, heterogeneity and dynamic change. IoT devices need to be connected to the Internet so that they remain accessible at all time from all the locations, therefore, inter-connectivity is core characteristics of IoT devices. Secondly, number of IoT devices is very huge and increasing rapidly. In order to support such a large number of devices, networks must be developed with sufficient capacity. Additionally, more IPs must be made available in order to link such a large number of IoT devices,

* Corresponding author.
E-mail addresses: moneebgohar@gmail.com, mgohar.buic@bahria.edu.pk (M. Gohar).

https://doi.org/10.1016/j.comnet.2023.109640
Received 18 July 2022; Received in revised form 2 February 2023; Accepted 15 February 2023
Available online 17 February 2023

hence a switch from IPV4 to IPV6 is imminent. Thirdly, IoT devices are manufactured by many different vendors and unfortunately no specific standard is being followed due to which devices differ to a great extent. Moreover, there are different use cases of IoT devices which completely have different sets of requirements. For instance, remote surgery, industrial manufacturing, autonomous vehicles, etc., are very sensitive to delay and very high precision is required. On the other side, temperature sensing devices for weather forecasts, etc., do not have such stringent requirements. Capabilities of the devices also differ, depending on their use case. Few have limited battery resources, while other do not have any such constraints, hence, heterogeneity pertaining to capabilities of the devices, the way these devices communicate and transport data is at higher end. Finally, one of the purposes of IoT devices is to gather data from the surroundings, which is changing from time to time. Therefore, the IoT devices also need to adapt to these changes like changing states from connected mode to sleeping mode and vice versa. Moreover, the requirements may change as new devices get connected to the Internet, generating new insights. Hence, dynamic change is the prime characteristic of IoTs.

IoT use cases have different requirements, therefore, a single application layer protocol does not fit for all. Consequently, there are many application level protocols currently being used, like Message Queuing Telemetry Transport (MQTT), Constrained Application protocol (CoAP), Extensible Messaging and Presence Protocol (XMPP), Advanced Message Queuing Protocol (AMQP), etc. These protocols have their own advantages and disadvantages due to which selecting a specific protocol for IoT device is a challenge [3]. Authors in [4] presented comprehensive overview of different IoT protocols along with strengths and weaknesses of each, which can be helpful in choosing the desired protocol for a particular use case. However, there is a dire need for having an interoperable and extensible protocol to cater for heterogeneity and adopting to the constantly developing new IoT use cases. In light of these requirements, XMPP and AMQP stand out from the other protocols. XMPP does not provide any QoS and encoding is text based, whereas AMQP provides three level of QoS and supports binary encoding.

Latency is an important parameter for many IoT use cases and researchers are working to find out ways to reduce it as much as possible. Researchers in [5] studied the performance of MQTT broker in smart city scenario and showed that latency is impacted by bandwidth and hardware resources. One of the factors introducing delay in communication is the underlying transport protocol like TCP (Transmission Control Protocol) and UDP (User datagram Protocol). TCP uses three-way handshake for establishing connections and depends on TLS for security due to which additional delay occurs. Although TCP provides reliable communication yet it is slow. Moreover, it faces issues like head of line blocking, half open connections, etc. A detailed discussion on the issues being faced by TCP in IoT scenario are presented in [6]. On the other hand, UDP provides quick but unreliable communication which is not desirable in many IoT scenarios. Recently a new protocol, QUIC [7] has been standardized at IETF whose aim is to combine the good qualities of both UDP and TCP so that quick and reliable communication can be realized. Additionally, QUIC can resolve a number of TCP related issues, such as head of line blocking, half open connections, and sending keep alive messages for maintaining established connections, all of which will be extremely helpful in IoT scenarios.

There are a number of IoT deployment scenario including smart farming, basement-installed smart metering, and other indoor applications like elevators where network coverage is usually inadequate, therefore, [8] regards enhanced coverage as a critical design requirement for large-scale IoT deployment. Poor coverage leads to packet loss, which increases the need for retransmissions and, in addition to the communication delay, more battery or energy is consumed. Therefore, a reliable transport layer protocol that can withstand delays and packet loss is essential. Further, IoT devices often do not need to communicate data continuously, so they enter sleep mode to conserve battery life. These devices are reactivated when data is ready to be sent. Reliable connections, like TCP, are necessary for reliable communication, but when no real data is being sent across a TCP connection, dummy keep-alive packets must be sent in order to prevent the TCP connection from being dropped. This would imply that either the devices cannot be put into sleep mode or that a new connection must be established every time data needs to be transferred. Higher energy use will follow from either case.

In this study we integrated AMQP 1.0 with QUIC to have quick and secure communication. We selected AMQP 1.0 because it is an extensible and interoperable protocol [4,9,10] which can deal with interconnectivity challenges in a heterogeneous IoT environment. On the other hand, using QUIC rather than TCP reduced communication time because cryptographic and transport parameters could be exchanged during a single QUIC handshake. As a result, both communication time and energy usage were significantly improved. QUIC is also more resilient to delays and packet losses, making it a superior choice in lossy network conditions. According to the findings [11,12], QUIC performs significantly better in lossy networks than TCP and UDP.

The remainder of this article is organized as follows. Section-II briefly presents Literature Review. Experimental Setup and Implementation used to analyze performance of AMQP 1.0 over TCP and AMQP 1.0 over QUIC is presented in Section 3. Results and Analysis is given in section Section-IV, whereas, Section-V concludes the paper along with the tasks planned in the future.

2. Literature review

A lot of research has been carried out in the past pertaining to IoT application and transport layers protocols, which are briefly reviewed below:

2.1. Preliminaries/background

MQTT, CoAP, XMPP and AMQP are few of the well known application level protocols being used in IoT scenarios. A brief overview of each is given below:

- MQTT was invented in 1999 and is OASIS standard. Its version 3.1.1 [13] has been endorsed by ISO as well. It is a lightweight protocol having very small code footprint, much suited for constrained devices and uses publish/ subscribe structure through which messages can be transferred from one to many devices.
- CoAP is a machine to machine communication protocol which uses request/response architecture and can be considered as light weight alternative of HTTP (Hypertext Transport Protocol) for constrained devices. CoAP was standardized in 2014 by Internet Engineering Task Force (IETF) and elaborated in RFC 7252 [14].
- XMPP is an open standard that was originally known as Jabber and allows for real-time communication, voice calls, video calls, and instant messaging. It is based on Extensible Markup Language (XML) and widely deployed over the web. XMPP is a viable option which can be used in heterogeneous IoT environment and can provide interoperability. The core functionality of XMPP protocol was standardized in 2004 through RFC 3920 and RFC 3921 which were later updated in 2011 through RFC 6120 [15] and RFC6121 [16].
- AMQP was initially developed for finance industry by JP Morgan Chase and then expanded into OASIS Working Group. It has several experimental releases (0.8, 0.9.1, 0.10) and standardized AMQP version 1.0 [17] is ISO/IEC 19464 standard since 2014. AMQP uses both publish /subscribe and request/response architecture for the exchange of messages. It is based on TCP and also support SCTP for multiple connections. It relies on TLS/SASL for security. AMQP 1.0 [17] uses layered model using transport and connection security protocol, frame transfer protocol, and

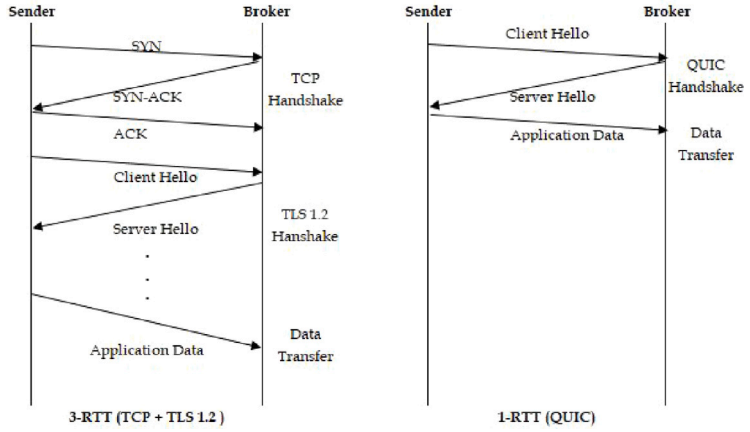


Fig. 1. TCP and QUIC Handshake.

Table 1
Performance comparison of IoT protocols.

Protocol	Transport layer	Security	Delay (s)
MQTT	TCP	Poor	0.1101
AMQP	TCP	Excellent	0.1255
CoAP	UDP	Average	0.8661
HTTP	TCP	Good	1.2360

Table 2
Architectural comparison.

Parameter	Existing scheme	Proposed scheme
Transport	TCP	QUIC
RTT	3-RTT	1-RTT
Security	TLS 1.2	TLS 1.3
Protocol overhead	Low	High
Connection migration	No	Yes
Multi streaming	No	Yes
Startup latency	High	Low

message transfer protocol. It supports message brokers, peer to peer messaging and hyper scale messaging infrastructures. In experimental releases of AMQP, broker consisted of three components namely exchange, binding and queues. However, AMQP 1.0 provides only a protocol specification and remains silent about broker architecture.

Authors in [18] compared performance of AMQP, MQTT, CoAP and HTTP using Raspberry Pi (client) and PC (server). It has been shown that delay of AMQP is slightly more than MQTT, but considerably less than CoAP and HTTP. The findings of [18] are summarized in the Table 1.

Transport layer is responsible for transfer of data between the two end points. It is responsible for service point addressing, segmentation and reassembly of data, connection control, flow control and error control. The reliability of data depends on transport layer. There are two well-known protocols at transport level namely Transmission Control Protocol (TCP) and User Datagram Protocol (UDP). TCP's specifications were set in RFC 675 [19] in 1974 and reclassified through RFC 7805 [20]. UDP is officially defined in RFC 768 [21] in 1980. TCP is a connection-oriented protocol in which connection needs to be established between client and server using three-way handshake process. TCP relies on Transport Layer Security (TLS) to offer security, and as additional parameters must be exchanged for TLS, there is an additional delay. TCP provide reliable delivery of packets in orderly fashion along with error checking. On the other hand, UDP is connectionless protocol which does need to establish connection with the client and server therefore it is quicker and more efficient than TCP, however, orderly delivery of packet is not ensured. UDP is more suitable in scenarios where error checking is not an essential requirement.

Although TCP is a reliable protocol for data transmission yet it faces various problems. Firstly, it uses three-way handshake for connection setup which normally takes around 1-Round Trip Time (RTT) and for security handshake addition 2 or 3 RTTs are required, this all adds up to delay. Secondly, keep-alive packets need to be sent in TCP to prevent the connection from drop out which consumes battery of constrained devices. The devices typically enter sleep mode when no data has to be transferred, therefore, a new connection must be established each time data needs to be sent. Thirdly, if multiple streams are being transmitted using Stream Control Transmission Protocol (SCTP) and the leading stream faces packet drops, all the other streams get blocked this is known as head of line blocking issue. Lastly, if TCP connection ends up abruptly from one side it might remain open on the other side, this is termed as half open connections and may be used by an attacker to cause Distributed Denial of Service (DDOS) through TCP SYN flooding [22].

QUIC is a new transport protocol, standardized by IETF [7] in 2021. It was initially proposed by Google and was known as gQUIC [23]. The objective of it is to combine the benefits of TCP and UDP in order to deliver TCP level reliability utilizing UDP, allowing for efficient and dependable data transfer. The communication between two end points take place using QUIC Packets which are carried in UDP datagram. Applications communicate with QUIC through streams which are structured and ordered sequences of bytes. QUIC streams can be of two types, i.e., bidirectional streams, in which both end points can send data and unidirectional streams in which only one endpoint can send data. QUIC uses feedback, congestion control and mechanism for detection and recovery from loss. QUIC uses built in security feature and during QUIC's initial handshake cryptographic parameters are also exchanged a part from transport parameters due to which delay is reduced considerably. In fact, it is even possible to set up connection with 0-RTT in which previously exchanged (stored) parameters are used until handshake completes. Additionally, since each QUIC connection is identified by a unique ID, the underlying IP address and port number can change without having an effect on the ongoing connection. This feature of QUIC, known as connection migration, enables the transfer of active connections between different networks. Further, QUIC also helps in addressing problems like head of line blocking and half open connections.

Fig. 1 shows the comparison of TCP and QUIC handshake, whereas architectural comparison between AMQP 1.0 over TCP (existing scheme) and AMQP 1.0 over QUIC (proposed scheme) is presented in the Table 2.

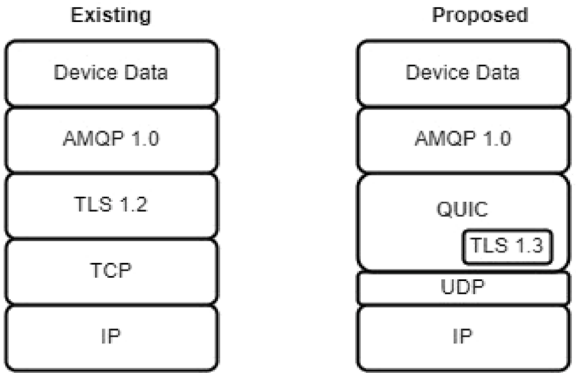


Fig. 2. Existing & Proposed Protocol Stack.

2.2. Related work

Researchers in the past have shown that QUIC can be a superior choice instead of TCP and UDP in IoT scenarios. According to authors in [24], QUIC can be very beneficial in IoT scenario because IoT devices are normally deployed in the areas where wireless conditions are normally not adequate, therefore TCP creates considerable delay and consumes more battery life of the constrained devices. Every time a device has to send data, a new connection needs to be established first. QUIC solves these shortcomings of TCP through 0-RTT. QUIC uses connection ID to identify the device so connection can be migrated, moreover, it also overcomes head of line blocking issue.

Researchers in past [25] studied the performance of MQTT over QUIC using wireless, wired and long distance test-beds built with Raspberry Pi 3B devices and demonstrated that MQTT with QUIC outperforms MQTT with TCP in terms of processor usage, memory usage and latency. Authors in [11] integrated MQTT with QUIC using Go programming language and carried out extensive testing in NS3 simulator. Performance of MQTT over QUIC was benchmarked with MQTT over TCP using three different network configurations, i.e., WiFi, 4G/LTE and Satellite and for each configuration 40 samples were collected. The study concluded that QUIC performs much better than TCP especially in case of lossy network conditions.

Authors in [26] integrated CoAP with QUIC and compared it with CoAP over UDP and CoAP over TCP. Mathematical model for message loss using Gilbert-Elliott's two state Markov channel model [27,28] is presented which is validated through emulation in VPS+. It has been argued that QUIC is less lossy than TCP and UDP.

MQTT and CoAP, which are two widely used protocols in IoT, have already been integrated with QUIC by the researchers in [25,26], respectively. However, AMQP 1.0 which is also an important extensible and inter-operable IoT protocol has not been yet integrated with QUIC. Therefore, in this study, we integrated AMQP 1.0 with QUIC and carried out extensive testing to evaluate its performance. Moreover, this study also considers power consumption analysis.

3. Experimental setup and implementation

Existing and proposed protocol stacks, used to transport AMQP 1.0's data are shown in Fig. 2.

AMQP 1.0's data is delivered over TCP in the existing scheme, and security is provided using TLS version 1.2. The proposed scheme, on the other hand, transports AMQP 1.0's data over QUIC. Because QUIC has built-in security and includes both transport and crypto parameters in its initial handshake, communication time is reduced. Both the aforementioned schemes were implemented in the Go programming language, and extensive testing was carried out in NS3. The Go programming language was selected because Go packages pertaining to native AMQP 1.0 [29] and QUIC [30] are publicly accessible. Similarly, NS3 is an open source discrete event network simulator, widely used by the researchers. The AMQP 1.0 sender, broker, and receiver were containerized using the Docker tool, and by utilizing the concept of Tap Bridges, they appeared as ghost nodes in NS3 and were connected with one another across a simulated WiFi network.

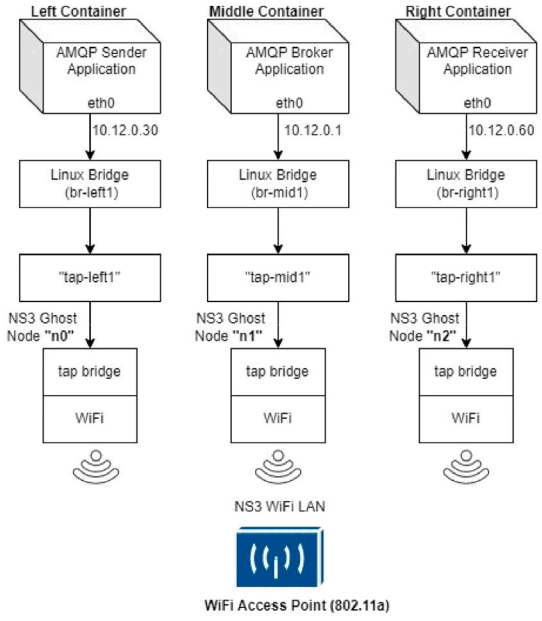


Fig. 3. Experimental Setup.

A high level block diagram of the experimental setup used to perform the testing is shown in Fig. 3. The left most column of Fig. 3 represents AMQP 1.0 Sender, the middle column is for AMQP 1.0 Broker, whereas the right most column shows the AMQP 1.0 Receiver. The WiFi LAN used for communication was IEEE 802.11a, operating at 5 GHz frequency with a 20 MHz channel bandwidth and Channel Number set to 36. The choice of specific WiFi version is not very important in the current study because we are primarily interested in comparing the proposed and the existing schemes. Additionally, the same WiFi version is used for both schemes, making the comparison valid.

Key performance indicators (KPIs) such as Total Communication Time, Startup Latency, Round Trip Time (RTT), and Throughput were used to compare the performance of our proposed scheme, AMQP 1.0 over QUIC, to that of the existing scheme, AMQP 1.0 over TCP. These KPIs are defined as under:

- **Total Communication Time**
  Total Communication Time is the total time from start to end of the communication, i.e., the time duration between first packet and last packet of the communication. It is measured in second and the formula is given below:

$$TotalCommunicationTime = Timeof\,LastPacket - Timeof\,FirstPacket$$

- **Startup Latency**
  The amount of time it takes for the first useful application packet to be sent from an AMQP 1.0 Sender to an AMQP 1.0 Broker is known as startup latency. A lower value for this KPI means that the useful application packet was transmitted earlier, resulting in a quicker start of actual communication. In case of using TCP at the transport layer, Startup Latency is the time it takes for the Sender to send the first application packet to the Broker after

the TCP and TLS handshake is complete. In the case of QUIC, Startup Latency is the time it takes for the Sender to send the first application packet with a destination connection ID (DCID) to the Broker.

- **Round Trip Time**
  Round trip time (RTT) is the time it takes for a packet to travel from source to destination and back. The AMQP 1.0 Sender acted as the source and the AMQP 1.0 Broker as the destination in the current scenario. RTT is measured in seconds.
- **Throughput**
  Throughput is defined as the rate of transfer of data or data volume per unit time. It is measured in bits/s (bps) or kilobits/s (Kbps).
- **Energy Consumption**
  Replacement of batteries or energy source is usually not feasible on a deployed node, therefore, energy consumption is used as performance metric to evaluate different protocols [31]. The NS3 Energy module, described in detail by the authors in [32], was used to assess battery performance. The authors of [33] have detailed the Energy Framework for NS3, which consists of two fundamental models: Energy Source Models and Device Energy Models. Energy Source Models are used to model the Power Supplies or batteries that are connected to the nodes and can be linear or non-linear. On the other hand, device Energy Models, such as the WiFi Radio Model, are used to model a node's energy usage.
  A node can be in different WiFi states, like Transmitting state (Tx), Receiving State (Rx), Clear Channel Assessment Busy state (CCA Busy), Idle state, Sleep state, Switching state and Off state. The current drawn by the node in different states varies. We have used the values mentioned in the official NS3 documentation [32]. The duration in each state is measured and then the following formula is used to calculate the energy consumption:

$$Energy(\mathrm{J}) = Voltage(\mathrm{V}) * Current(\mathrm{A}) * Time(\mathrm{s})$$

  According to authors in [34], the supply voltage of commercial IoT devices varies, although the common range is between 1.8 V and 3.3 V. Therefore, we used a 3 V voltage supply and the NS3 linear/ideal energy source model for our research. Authors of [31] argued that ideal/linear energy models do not depict the actual energy consumption in wireless sensor network whereas the result obtained from non-linear model (Rakhmatov Vrudhula Model) are more accurate. However, for the sake of this study, a linear/ideal energy source model was used because it is simple to implement and a fair comparison can be drawn because both the schemes are evaluated using the same model.

3.1. Implementation of AMQP 1.0 over TCP

The AMQP 1.0 Sender, Broker, and Receiver were implemented using the freely downloadable Go package [29]. AMQP 1.0 standard [17] specifies TLS 1.2, defined in RFC 5246 [35], to be used to provide security besides SASL, defined in RFC 4616 [36]. However, the aforementioned implementation of AMQP 1.0 did not use TLS 1.2, therefore, necessary modifications were made accordingly.

3.2. Integrating AMQP 1.0 with QUIC

A new Go package namely qConn was developed to integrate AMQP 1.0 with QUIC. In this new package Listen and Dial functions were defined along with other supporting functions which help creating QUIC broker, sender and receiver. The function qConn.Listen () was used to implement QUIC AMQP 1.0 Broker. This function, defined in the qConn package, makes use of quic.ListenAddrEarly() to create QUIC Broker, listening on a specified address. It works like the function quic.ListenAddr (), but returns connections before the handshake completes. The function, quic.ListenAddrEarly(), enables AMQP 1.0 Sender that has previously connected to the AMQP 1.0 Broker to access the data kept in the session's cache. As a result, re-establishing the connection does not necessitate re-negotiating every parameter, and data exchange occurs before the handshake is completed. Similarly, QUIC AMQP 1.0 Sender and QUIC AMQP 1.0 receiver uses the function qConn.Dial () which subsequently uses quic.DialAddrEarly () to establish a new QUIC connection with QUIC AMQP 1.0 Broker.

Table 3
Parametric values used for performance Benchmarking.

Parameter	Variation
Bandwidth (Kbps)	8, 16, 32, 512, 1024
Delay (ms)	0, 200, 400, 600
Packet loss (%)	0, 5, 10, 15

4. Results and analysis

A comprehensive testing campaign was conducted on NS3, which was installed on a virtual machine with 4 processors, 8 GB of RAM, and the Ubuntu 20.04 LTS operating system. Multiple scenarios pertaining to multiple AMQP connections, single AMQP connection and energy consumption were tested. Furthermore, both the existing and proposed schemes were evaluated under a variety of network situations in order to benchmark their performance. Network factors such as Delay, Packet Loss, and Channel Bandwidth were varied using tc tool and tests were conducted in this regard. Table 3 shows different configuration settings used in the testing.

All of the testing scenarios were run multiple times in order to reduce errors and produce better results.Moreover, *pcap* traces were enabled within NS3 script so that results could be obtained conveniently and with accuracy. The results and analysis of the testing performed is presented below:

4.1. Multiple AMQP connections

In this test case, depicted in Fig. 4, a single message was sent from AMQP 1.0 Sender to AMQP 1.0 Broker and then the connection was closed. In each experiment 100 messages were sent. Furthermore, each experiment was run 30 times. The purpose of sending 100 messages in a single experiment was to create 100 new connections in order to get accurate statistics. Additionally, the experiment was repeated 30 times to obtain even better statistical results. Consequently, we collected 30000 samples for each of the existing and the proposed schemes. Total Communication Time to send 100 messages from AMQP 1.0 Sender to AMQP 1.0 Broker was measured. Similarly, Total Communication Time to receive 100 messages by AMQP 1.0 Receiver from AMQP 1.0 Broker was also measured. The aim was to analyze the advantage brought by QUIC's 1-RTT or 0 -RTT connection establishment over traditional three-way handshake of TCP. Time comparison between QUIC AMQP 1.0 Sender and TCP AMQP 1.0 Sender is given in Fig. 6 while AMQP 1.0 Receiver time comparison in shown in Fig. 7. On average Total Communication Time to send 100 messages was reduced by 22% while receive time was reduced by 15% using QUIC at transport level.

4.2. Single AMQP connection

In this testing scenario, shown in Fig. 5, 1000 messages were sent in a single AMQP connection over WiFi channel and then the connection was closed. Since there would only be one connection made during each experiment in this scenario, the number of messages per experiment was increased from 100 to 1000 so that each experiment could be finished in a reasonable amount of time. Total Communication Time measured during this testing scenario is shown in Fig. 8, while Fig. 9 represents Startup Latency and RTT. The study

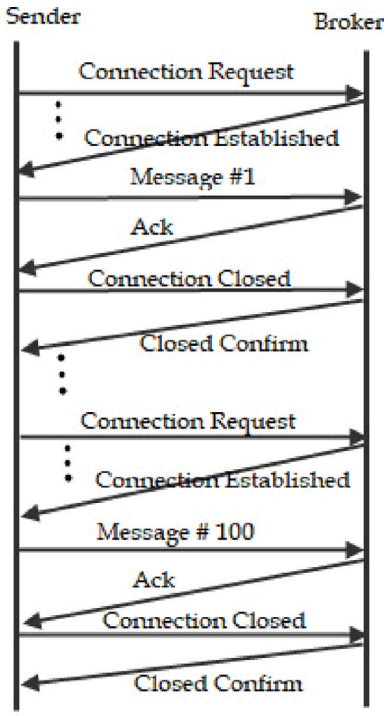


Fig. 4. Multiple AMQP Connections.

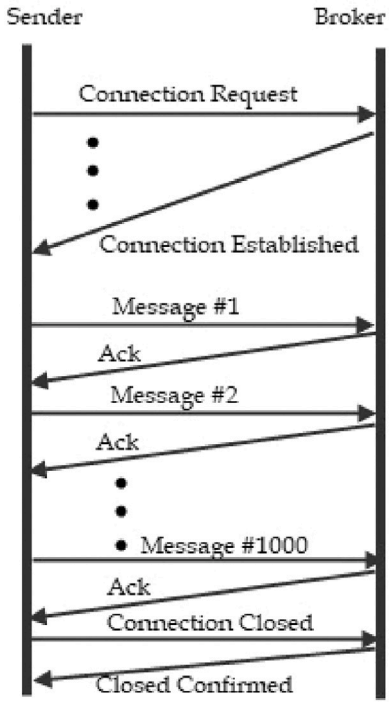


Fig. 5. Single AMQP Connection.

demonstrated that, despite QUIC's RTT being 71% higher, startup latency and total communication time had improved by 62% and 21%, respectively. Therefore, the improvement in Startup Latency and Total Communication Time cannot be attributed to RTT.

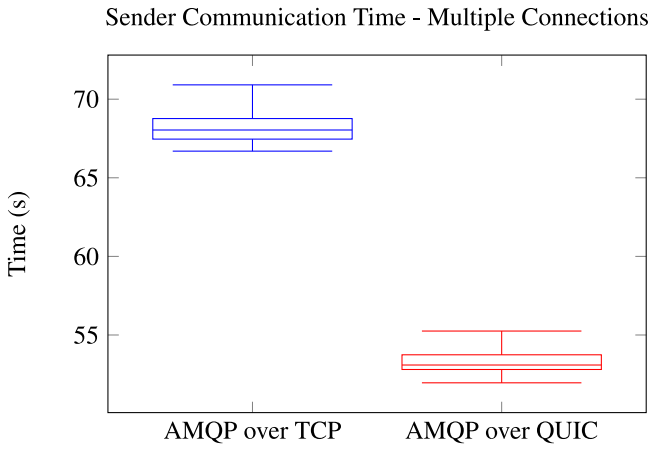


Fig. 6. Total Communication Time between Sender and Broker over Multiple AMQP Connections.

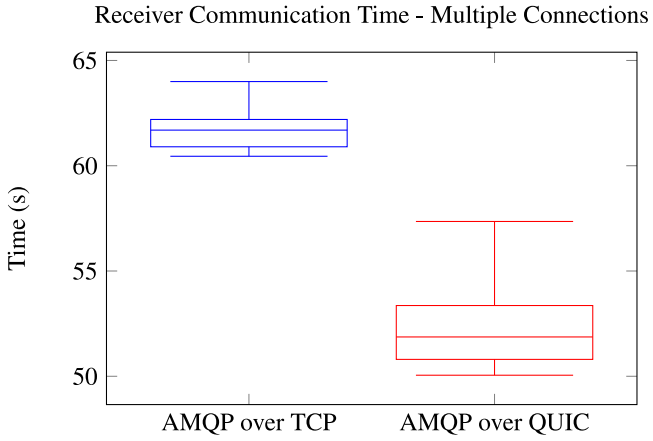


Fig. 7. Total Communication Time between Receiver and Broker over Multiple AMQP Connections.

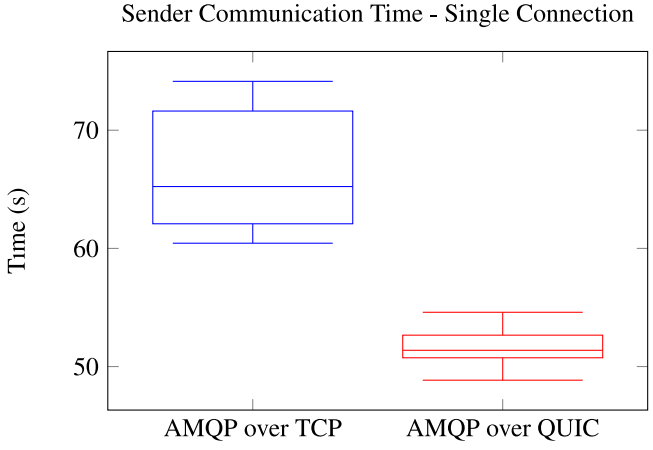


Fig. 8. Total Communication Time between Sender and Broker over Single AMQP Connection.

4.3. Energy analysis

A scenario similar to that described in Fig. 4 was utilized for energy evaluation, however, in each experiment 10 connections were established and on each connection a sample message of 138 bytes

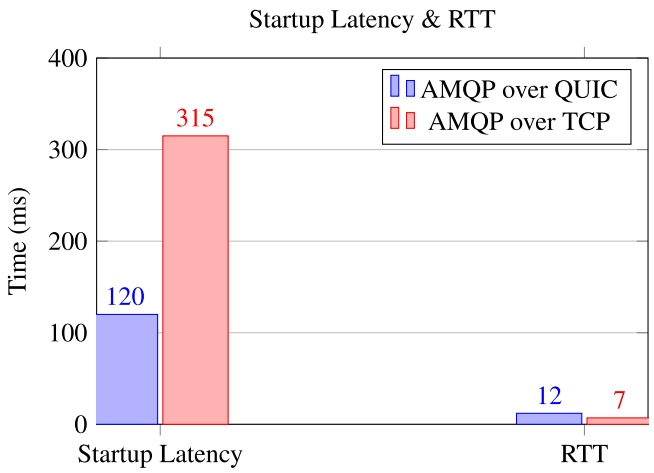


Fig. 9. Startup Latency & RTT.

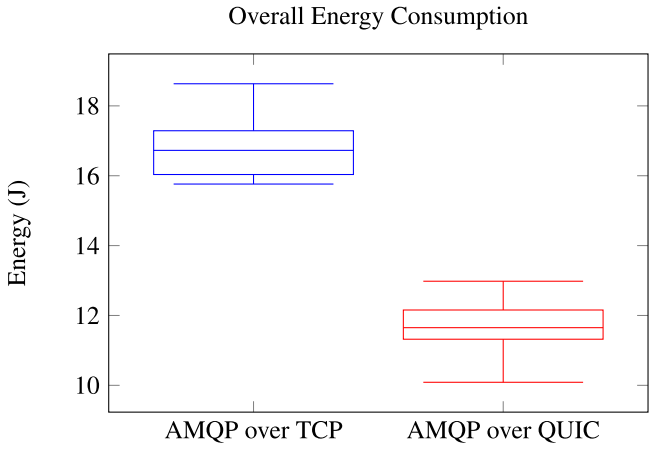


Fig. 10. Overall Energy Consumption.

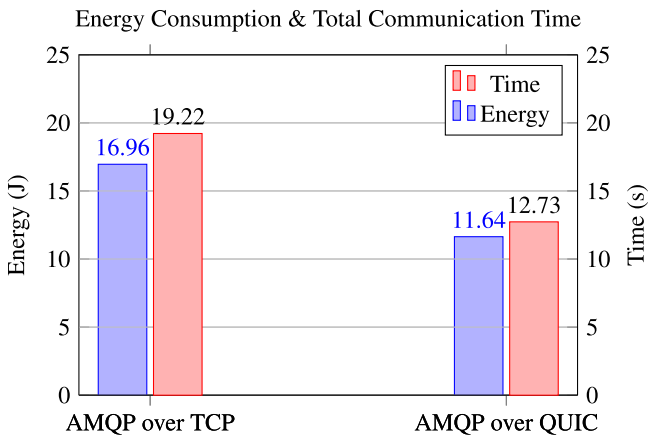


Fig. 11. Energy Consumption & Total Communication Time.

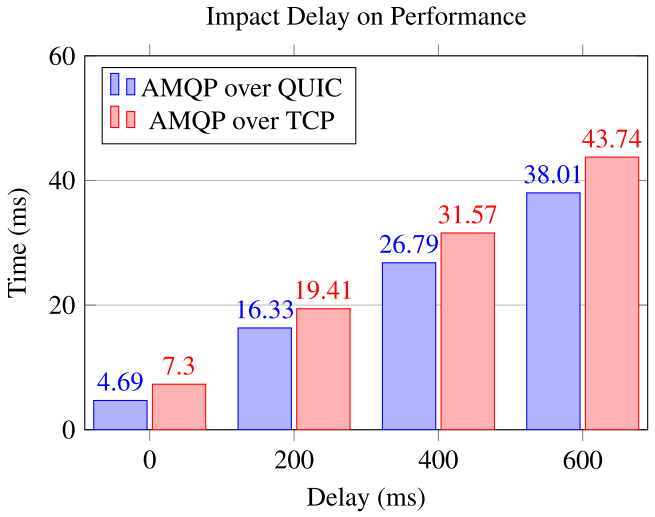


Fig. 12. Impact of Delay on Performance.

was sent from AMQP 1.0 Sender to AMQP 1.0 Broker. Moreover, the experiment was repeated 15 times. Fig. 10 shows a box plot of total energy consumption for the proposed and existing schemes, and Fig. 11 compares average energy consumption and average Total Communication Time for both the schemes. On average, it was found that adopting QUIC at the transport level reduced energy consumption by 31% and Total Communication Time by 34%.

4.4. Varying network parameters

Network parameters such as Delay, Packet Loss and Channel Bandwidth were tweaked using tc tool, and the performance of AMQP 1.0 over TCP and AMQP 1.0 over QUIC was evaluated. In a single experiment, 10 connections were established for each setting of a particular parameter, and the same experiment was repeated five times. Therefore, for each configuration we established 50 connections. Parametric settings shown in the Table 3 were utilized for testing performance and Total Communication Time was measured in each scenario. Testing results are presented as under:

4.4.1. Delay

Delay is an important factor in many IoT scenarios, such as remote surgery, autonomous vehicles, industrial automation, etc., thus we evaluated both AMQP 1.0 over TCP and AMQP 1.0 over QUIC under various delay situations to see how it affected their performance. In the first experiment, no additional delay was introduced in the network and we referred it as the baseline delay scenario or the zero-delay scenario. However, it should be emphasized that the network has some intrinsic delay, which is definitely not zero. The impact of delay on the performance of both the schemes is shown in Fig. 12. The Total Communication Time for both AMQP 1.0 over TCP and AMQP 1.0 over QUIC increased as the delay increased, however, in all of the delay-related scenarios, AMQP 1.0 over QUIC outperformed AMQP 1.0 over TCP.

4.4.2. Packet loss

Packet Loss has a significant impact on performance, causing retransmissions and communication delays. The Packet Loss bar graph in Fig. 13 shows the results of the experiments. It can be seen that the performance of AMQP 1.0 over TCP was noticeably worse than that of AMQP 1.0 over QUIC at higher Packet Loss Rates. Total Communication Time doubled when using TCP at a Packet Loss rate of 15%, but it only increased by 50% when using QUIC.

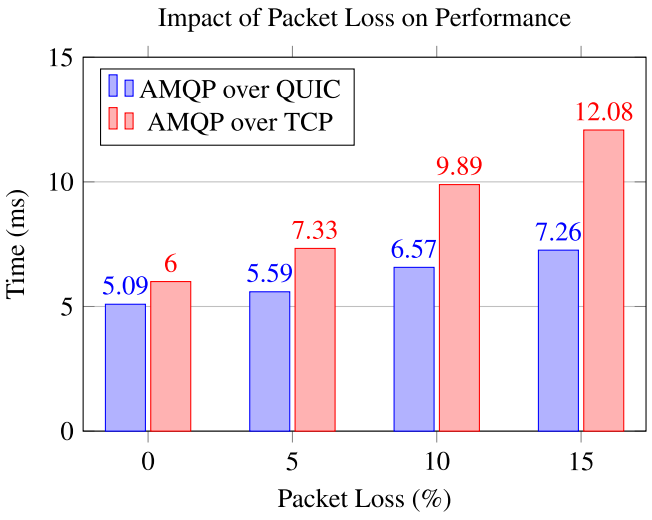


Fig. 13. Impact of Packet Loss on Performance.

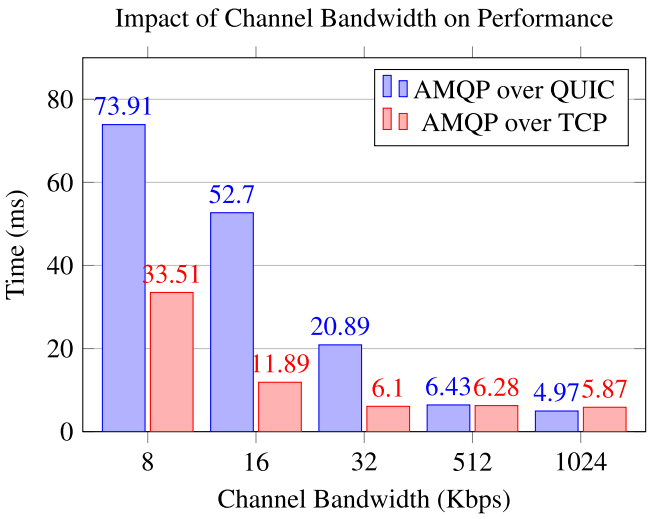


Fig. 14. Impact of Channel Bandwidth on Performance.

4.4.3. Channel bandwidth

Channel Bandwidth is the maximum allowed transmission rate, whereas data rate is actual transmission rate, hence Channel Bandwidth corresponds to the maximum capacity of the channel. Fig. 14 depicts the findings of this testing. It can be seen that QUIC outperformed TCP at higher Channel Bandwidths, however, TCP outperformed QUIC at lower Channel Bandwidths (below 32Kbps). The reason for this behavior is that QUIC is a more secure protocol than TCP, hence the overhead of QUIC is higher than that of TCP. As a result, in the case of QUIC, more data must be transferred, and when Channel Bandwidth is insufficient, performance suffers.

The volume of data transferred using QUIC and TCP over different Channel Bandwidth settings is shown in Fig. 15, while Fig. 16 compares Data Volume and Data Throughput using QUIC and TCP, when the Channel Bandwidth is not restricted. It should be noted that even in this scenario, QUIC sent 3.5 times more data than TCP. Hence, security overheads, rather than retransmissions, account for the additional data in the case of QUIC. Furthermore, because QUIC's throughput was 7 times better than that of TCP, QUIC was able to reduce Total Communication Time.

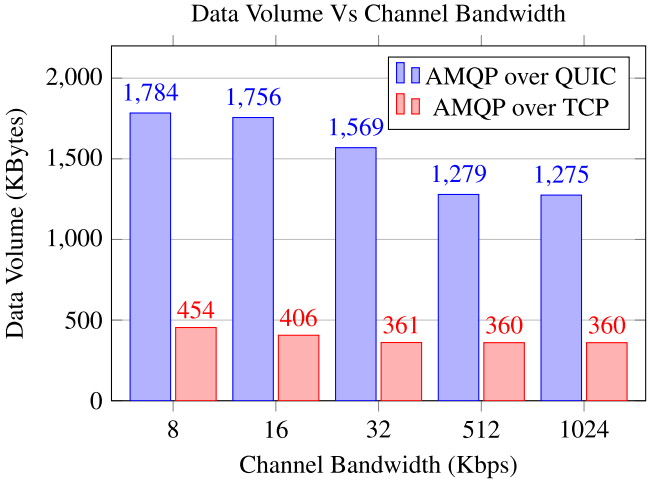


Fig. 15. Data Volume Vs Channel Bandwidth.

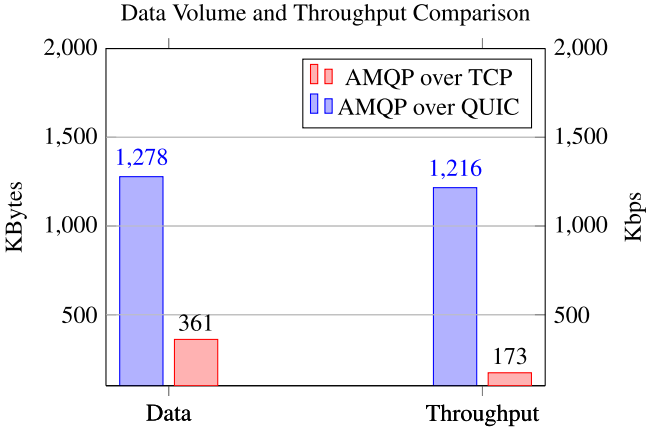


Fig. 16. Data Volume and Throughput Comparison.

5. Conclusion & future work

We implemented AMQP 1.0 over QUIC and used NS3 to do extensive testing. The Total Communication Time and Startup Latency have been improved significantly, allowing for speedy and dependable connection. The results showed that utilizing QUIC instead of TCP at the transport level lowered Total Communication Time by 22% and Startup Latency by 62 percent. Furthermore, energy testing was carried out, which revealed that AMQP 1.0 over QUIC used 31% less energy than AMQP 1.0 over TCP. The existing and the proposed schemes were put to the test under a variety of network situations. The results demonstrated that AMQP 1.0 over QUIC outperformed AMQP 1.0 over TCP in all of the delay tests. Total Communication Time increased by two times for AMQP 1.0 over TCP at a packet loss rate of 15%, whereas for AMQP 1.0 over QUIC, an increase of 50% was seen. The results also revealed that AMQP 1.0 over QUIC beat AMQP 1.0 over TCP at higher Channel Bandwidths, however, AMQP 1.0 over TCP outperformed AMQP 1.0 over QUIC at lower Channel Bandwidths (below 32Kbps). This behavior is exhibited because QUIC, as a more secure protocol, has greater overheads. Further, AMQP 1.0 over QUIC transported 3.5 times more

data than AMQP 1.0 over TCP, and AMQP 1.0 over QUIC had a 7-fold higher throughput than AMQP 1.0 over TCP, according to the findings of the tests.

We employed linear/ideal energy sources for energy analysis, which yielded reasonable results; but, in the future, we want to use more realistic non-linear models to obtain more accurate energy estimates. We also want to test QUIC's multi-streaming and connection migration features in the future.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

No data was used for the research described in the article.

Acknowledgments

This reaserch was supported by Princess Nourah bint Abdulrahman University Researchers Supporting Project number (PNURSP2023R192), Princess Nourah bint Abdulrahman University, Riyadh, Saudi Arabia.

References

[1] I.T.S. Sector, Recommendation ITU-T Y. 2060: Overview of the Internet of things, in: Series Y: Global Information Infrastructure, Internet Protocol Aspects and Next-Generation Networks-Frameworks and Functional Architecture Models, 2012, pp. 2060–201206.

[2] U. Cisco, Cisco annual internet report (2018–2023) white paper, March 2020, 2020.

[3] M. Bansal, et al., Performance comparison of MQTT and CoAP protocols in different simulation environments, in: Inventive Communication and Computational Technologies, Springer, 2021, pp. 549–560.

[4] E. Al-Masri, K.R. Kalyanam, J. Batts, J. Kim, S. Singh, T. Vo, C. Yan, Investigating messaging protocols for the internet of things (IoT), IEEE Access 8 (2020) 94880–94911.

[5] D.L. de Oliveira, A.F. da S. Veloso, J.V.V. Sobral, R.A.L. Rabêlo, J.J.P.C. Rodrigues, P. Solic, Performance evaluation of MQTT brokers in the internet of things for smart cities, in: 2019 4th International Conference on Smart and Sustainable Technologies, SpliTech, 2019, pp. 1–6.

[6] C. Gomez, A. Arcia-Moret, J. Crowcroft, TCP in the Internet of Things: from ostracism to prominence, IEEE Internet Comput. 22 (1) (2018) 29–41.

[7] J. Iyengar, M. Thomson, QUIC: A UDP-based multiplexed and secure transport, in: Request for Comments, (9000) RFC Editor, 2021, RFC 9000.

[8] G.A. Akpakwu, B.J. Silva, G.P. Hancke, A.M. Abu-Mahfouz, A survey on 5G networks for the internet of things: Communication technologies and challenges, IEEE Access 6 (2018) 3619–3647.

[9] D. Glaroudis, A. Iossifides, P. Chatzimisios, Survey, comparison and research challenges of IoT application protocols for smart farming, Comput. Netw. 168 (2020) 107037.

[10] S. Elhadi, A. Marzak, N. Sael, S. Merzouk, Comparative study of IoT protocols, in: Smart Application and Data Analysis for Smart Cities, SADASC'18, 2018.

[11] F. Fernández, M. Zverev, P. Garrido, J.R. Juárez, J. Bilbao, R. Agüero, And QUIC meets IoT: performance assessment of MQTT over QUIC, in: 2020 16th International Conference on Wireless and Mobile Computing, Networking and Communications, WiMob, 2020, pp. 1–6.

[12] R. Herrero, Analysis of the constrained application protocol over quick UDP internet connection transport, Internet Things 12 (2020) 100328.

[13] O. Standard, MQTT Version 3.1. 1, Vol. 1, 2014, URL http://docs.oasis-open.org/mqtt/mqtt/v3.

[14] Z. Shelby, K. Hartke, C. Bormann, The constrained application protocol (CoAP), in: Request for Comments, (7252) RFC Editor, 2014, RFC 7252.

[15] P. Saint-Andre, Extensible messaging and presence protocol (XMPP): Core, in: Request for Comments, (6120) RFC Editor, 2011, RFC 6120.

[16] P. Saint-Andre, Extensible messaging and presence protocol (XMPP): Instant messaging and presence, in: Request for Comments, (6121) RFC Editor, 2011, RFC 6121.

[17] O. Standard, OASIS advanced message queuing protocol (AMQP) version 1.0, Int. J. Aerosp. Eng. Hindawi 2018 (2012) www.hindawi.com.

[18] Z. Fan, J. Kim, Transmission performance comparison and analysis with different publish/subscribe protocol, in: Proceedings of the Korean Society of Computer Information Conference, Korean Society of Computer Information, 2020, pp. 77–80.

[19] Specification of internet transmission control program, in: Request for Comments, (675) RFC Editor, 1974, RFC 675.

[20] A. Zimmermann, W. Eddy, L. Eggert, Moving outdated TCP extensions and TCP-related documents to historic or informational status, in: Request for Comments, (7805) RFC Editor, 2016, RFC 7805.

[21] User datagram protocol, in: Request for Comments, (768) RFC Editor, 1980, RFC 768.

[22] W. Eddy, TCP SYN flooding attacks and common mitigations, in: Request for Comments, (4987) RFC Editor, 2007, RFC 4987.

[23] Google-QUIC, https://www.chromium.org/quic.

[24] U.S. Kumar, et al., Application layer protocols for embedded system applications: A comparison, 2020.

[25] P. Kumar, B. Dezfouli, Implementation and analysis of QUIC for MQTT, Comput. Netw. 150 (2019) 28–45.

[26] R. Herrero, Analysis of QUIC transported CoAP, SN Comput. Sci. 2 (2) (2021) 1–11.

[27] E.N. Gilbert, Capacity of a burst-noise channel, Bell Syst. Tech. J. 39 (5) (1960) 1253–1265.

[28] E.O. Elliott, Estimates of error rates for codes on burst-noise channels, Bell Syst. Tech. J. 42 (5) (1963) 1977–1997.

[29] AMQP-Electron-Package, https://pkg.go.dev/github.com/apache/qpid-proton/go/pkg/electron.

[30] QUIC-GO-Package, https://pkg.go.dev/github.com/lucas-clemente/quic-go.

[31] M.A. Spohn, P.S. Sausen, F. Salvadori, M. Campos, Simulation of blind flooding over wireless sensor networks based on a realistic battery model, in: Seventh International Conference on Networking, Icn 2008, IEEE, 2008, pp. 545–550.

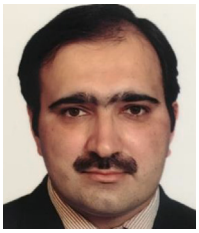
[32] NS3-Energy-Model, https://www.nsnam.org/doxygen/group-energy.html.

[33] H. Wu, S. Nabar, R. Poovendran, An energy framework for the network simulator 3 (ns-3), in: Proceedings of the 4th International ICST Conference on Simulation Tools and Techniques, 2011, pp. 222–230.

[34] M. Alioto, M. Shahghasemi, The Internet of Things on its edge: Trends toward its tipping point, IEEE Consum. Electron. Mag. 7 (1) (2017) 77–87.

[35] T. Dierks, E. Rescorla, The transport layer security (TLS) protocol version 1.2, 2008, RFC 5246, August.

[36] K. Zeilenga, The PLAIN Simple Authentication and Security Layer (SASL) Mechanism, Tech. rep., 2006, RFC 4616, August.



Faheem Iqbal He did his MS from Bahria University, Islamabad, Pakistan.

Moneeb Gohar He received B.S. degree in Computer Science from University of Peshawar, Pakistan, and M.S. degree in Technology Management from Institute of Management Sciences, Pakistan, in 2006 and 2009, respectively. He also received Ph.D degree from the School of Computer Science and Engineering in the Kyungpook National University, Korea, in 2012. From September 2012 to September 2014, he worked as a Post-Doctoral researcher for Software Technology Research Center (STRC) in Kyungpook National University, Korea. He has been as an International Research Professor with the Department of Information and Communication Engineering in the Yeungnam University since September 2014. Currently, He is a Full Professor in Bahria University Islamabad. His current research interests include Network Layer Protocols, Wireless Communication, Mobile Multicasting, Wireless Sensors Networks, TRILL,Security and Internet Mobility. Email: moneebgohar@yu.ac.kr.

Hanen karamti Hanen karamti is with the Department of computer sciences, College of Computer and Information Sciences, Princess Nourah bint Abdulrahman University.

Walid Karamti Walid Karamti is with the Department of Computer Science, College of Computer, Qassim University, Buraydah, 51452, Saudi Arabia.

Seok-Joo Koh He received the B.S. and M.S. degrees in Management Science from KAIST in 1992 and 1994, respectively. He also received Ph.D. degree in Industrial Engineering from KAIST in 1998. From August 1998 to February 2004, he worked for Protocol Engineering Center in ETRI. He has been as a professor with the school of Computer Science and Engineering in the Kyungpook National University since March 2004. His current research interests include mobility management in the future Internet, IP mobility, multicasting, and SCTP. He has so far participated in the international standardization as an editor in ITU-T SG13 and ISO/IEC JTC1/SC6. Email: sjkoh@knu.ac.kr.

Jin-Ghoo Choi Jin-Ghoo Choi received his Ph.D. degree from the School of Electrical Engineering & Computer Science, Seoul National University in 2005. From 2006 to 2007, he worked for Samsung Electronics as a senior engineer. In 2009, he was with the Department of Electrical & Computer Engineering in The Ohio State University as a visiting scholar. He joined the Department of Information and Communication Engineering in Yeungnam University as a faculty member in 2010. His research interests include performance analysis of communication networks, resource management in wireless networks, and wireless sensor network. Email: jchoi@yu.ac.kr.