Adaptive Hybrid Application Protocol for IoT

Erdal ÖZDOĞAN[1], Osman Ayhan ERDEM[2] and Ahmet Nusret ÖZALP[3]

[1] IT Department, Gazi University, 06560 Ankara, Turkey, erdal.ozdogan@gazi.edu.tr

[2] Department of Computer Engineering, Gazi University, 06560 Ankara, Turkey, ayerdem@gazi.edu.tr

[3] Republic of Turkey Ministry Education, 06640 Ankara, Turkey, ahmetnusretozalp@karabuk.edu.tr

Abstract: The Internet of Things (IoT) solutions are demonstrating a significant impact in various sectors. These solutions encompass diverse applications, ranging from machine-to-machine communication to human-to-machine interaction within the same system. Various types of IoT application protocols have been developed to address these needs. While some IoT protocols enable direct communication between devices, others rely on server-based communication. Both forms of communication can coexist within the same IoT solution. However, utilizing multiple protocols in low-capacity IoT nodes can result in performance degradation. In this study, an adaptive protocol is proposed, which combines server-based and direct access protocols. This study also explains the working principle of the proposed protocol and compares its functionality with MQTT using OPNET and real environment. It was observed that the proposed protocol remained functional even in the event of a server failure. Furthermore, the protocol was found to be efficient in terms of bandwidth usage. Conversely, it exhibited lower latency in the direct communication method. Although the proposed protocol had slightly higher latency in server-based communication, it remained operational.

Keywords: Adaptive IoT Protocol; Application Layer IoT Protocols; IoT Messaging Protocols; IoT Protocol Design; Protocol Simulation

1 Introduction

Internet of Things (IoT) environments are becoming more and more common every day and are used in many areas such as smart cities, smart campuses, smart homes and smart buildings. Developments in sensor and communication technologies and low cost of hardware have caused many segments, from amateurs to professionals, to use IoT applications. The heterogeneous nature of IoT and the diversity of applications have caused to complex solutions.

The existence of multiple IoT hardware types, especially in environments such as smart factories and smart campuses, has also led to the diversity of communications. Some communications are established directly between machines (M2M), while others occur through servers or the cloud. It is possible to see both types of communication in a complex IoT solution. On the other hand, the hardware used in IoT applications are generally have limited resources. It is obvious that low-capacity IoT nodes cannot provide adequate support in environments where too much data is produced, transferred and processed. Therefore, new application protocols have been developed for IoT environments Application protocols such as MQTT, CoAP, and AMQP have been developed for this purpose and can run on low-resource devices in IoT environments. Considering the communications in IoT solutions, it is possible to divide them into two categories: directly communication protocols and server-based protocols [1] MQTT and AMQP need a server (broker) for communication. CoAP, on the other hand, is capable of communicating directly between two nodes. In a complex IoT solution, both direct communication and server-based communication can be seen in the same environment. Direct communication method can be used especially in automation applications, M2M communications and applications that require high speed and low latency [2]. Both methods have their pros and cons[3]. Server-based communication may cause single point of failure (SPoF - Single Point of Failure[4]. In addition, the use of a central server in data transmission is against the distributed nature of IoT, where resources are distributed to things [5] [6]. Also, the presence of an additional device between source and destination can cause additional delay [7]. There may be performance degradation such as exponential increase in processor consumption in direct communication method. In addition, the direct communication method is inadequate, especially when data needs to be stored or analysed, and when decision-making mechanisms are needed [8]. Despite their pros and cons, we may see more IoT applications in the near future where both approaches are required in the same solution. But constrained devices in IoT environments are a major challenge for running two different protocols in the same solution [9]. Adding an additional protocol, to a complex solution, creates more complexity. It also requires an intermediary translator for the two protocols to communicate with each other, which also means additional complexity.

Choosing the most appropriate application layer protocol depends on the purpose of the solution and the changing states of the network. These states include bandwidth usage, packet loss probability, end-to-end delay, throughput and the size of the transmitted packet [10]. However, these parameters change frequently in the network environment [11]. When the state of the network changes, the selected protocol may then experience performance issues. Therefore, there is a need for a protocol that can adaptively switch between communication methods according to the changing state of the network. In this study, a new adaptive application layer protocol has been developed that can support both communication approaches. It is possible to see many performance comparisons

of IoT application protocols in the literature. The protocol developed in this study has not been compared with all IoT protocols. But to evaluate the performance of the developed protocol, it is compared with the MQTT protocol as benchmark [12]. In this study, a new protocol is proposed that combines both direct communication and server-based communication into a single approach. Thus, a single protocol can be used instead of two different protocols that support different approaches in the same environment. The contribution of the article can be summarized as follows:

- Introduction of a new hybrid protocol that supports different IoT approaches
- Development of a light protocol that can work in complex IoT solutions
- Introduction of a new adaptive mechanism that combines the advantages of server-based operating protocols with the advantages of directly communicating protocols into a single protocol

Some limitations were taken into account in the design of the protocol. Accordingly, the environment in which the protocol will work is assumed to be safe, therefore security principles are out of scope. Another limitation is related to the size of the data. In IoT environments, especially considering the small size of the data produced from the sensors, environments where the amount of data carried in a single package is at most 1024 Bytes. It is designed to be used in the transfer process of data obtained from sensors, especially in environments such as campus networks. The other parts of this article are organized as follows: In the second part, related work on this subject, is included in Chapter 2. The working phases, components, diagrams, and packet structures of the protocol are explained in the Chapter 3. In Chapter 4, the behaviors of the protocol in OPNET simulation and its performances in real-world scenarios have been evaluated. In Chapter 5, results and discussions focus on the academic contributions made by the study and evaluations that will provide insights for the future are presented. Chapter 6 provides related conclusions.

2 Related Work

In studies on existing IoT application protocols, these protocols can be superior in different scenarios and under different conditions. In the study of Thangavel et al. [13], MQTT and CoAP protocols is examined in terms of packet loss, message delays and data transferred per message through middleware implementation. In the study, it is stated that MQTT has a lower delay in low packet losses and COAP has a lower delay in high packet losses. Çorak et al. [10] evaluate the well-known IoT protocols in real world testbed. In this study where packet creation time and packet delivery time were evaluated as metrics, it was stated that XMPP performed worst, while CoAP and MQTT performed approximately the same.

Moraes et al. [14] compare AMQP, CoAP and MQTT protocols for throughput, message size and packet loss, and the authors state that CoAP showed the best results. In the study of Maksymyuk et al. [15], payload transmission of MQTT and CoAP is examined in NB-IoT environment. The results show that MQTT performs better for in-band deployment, while CoAP shows higher throughput. In Naik's study [16], IoT protocols are examined from various aspects. It is stated in the research that AMQP is applied more successfully in big projects and MQTT is used by many organizations. De Caro et al. [17] perform a qualitative and quantitative comparison between MQTT and CoAP in their study. The authors state that performance may vary depending on the circumstances. For example, CoAP is successful in terms of bandwidth usage, whereas MQTT is successful in 20% packet losses. Chen and Kunz [18] evaluate the MQTT, CoAP, DDS and Custom UDP protocols in the medical environment using a network emulator. In the study, the observed performance of the protocols is reported, and it is stated that DDS uses higher bandwidth than MQTT. Mijovic et al. [19] compare the performance of CoAP, webSocket and MQTT protocols. Study shows that CoAP achieves the highest protocol efficiency and the lowest average RTT. Collina et al. [20], provide a quantitative analysis of MQTT and CoAP for various traffic conditions. The results show that MQTT provides the smallest delays, but CoAP performs better in heavy traffic. In Bandyopadhyay and Bhattacharyya study [21], it is demonstrated that DDS and MQTT protocols do not experience any packet loss in networks with an average 25% packet loss and 400 ms delay.

The heterogeneous nature of IoT environments has also revealed the need for multiple protocols to work together in the same environment. For this reason, studies involving various hybrid approaches are also conducted. In the study [9], conducted by Bellavista and Zanni to support the scenarios of having multiple IoT application protocols in the same environment, an architecture is developed in order to enable CoAP and MQTT protocol to work together. In the study, it is state that the architecture offers high scalability in network environments with high traffic and device density. In the study of Derhamy et al. [22], It is state that using middleware software creates a scaling problem, alternatively, SOA-based protocol transformation is proposed. The authors stated that the method they developed showed low delays. Lee et al. [23], designed SDN based hybrid IoT communication framework to achieve bi-directional data exchange without further modification on existing protocols. In the study of Desai et al. [24], conversion of XMPP, CoAP, and MQTT messages is achieved through the multi-protocol proxy architecture. Khaled et all. [6], introduce “Atlas Framework” for the interaction of different communication protocols. The results show the feasibility of enabling seamless heterogeneous communication between things with an acceptable energy cost.

As can be seen from the studies, the performances of the protocols that show different approaches according to the dynamic variables in the network also change. Considering that using multiple protocols in the same environment causes

additional delays and scalability problems, protocols that act differently according to the dynamic variables in the network are needed.

3 Adaptive Hybrid IoT Protocol

As a new approach to the Internet of Things, we designed an adaptive IoT protocol (hIoT). This section summarizes the quick overview of the developed protocol. There are three components in the adaptive IoT Protocol: IoT Coordinator, Client and IoT Gateway (IoTGW).

The IoT Gateway undertakes the task of presenting the data, which is the source of the data and obtained from the sensors to the network environment. The client component represents the user, hardware or application that wants to access the data. The coordinator component, on the other hand, acts as an intermediary server (broker) for data transfer in server-based communication. Likewise, the records of the services offered in the network through the IoT Gateway components are also kept in the coordinator component. The services offered by IoTGW have been developed in a "topic" format to comply with the MQTT protocol.

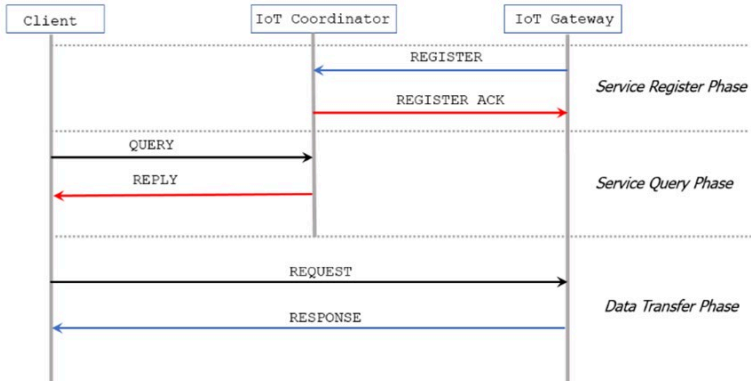

Figure 1
Hybrid IoT protocol has three phases: Service register, Service query and Data transfer

The working principle of the protocol is examined in three phases (Service Register, Service Query and Data Transfer) as shown in Figure 1. In the Service Registration phase, the IoT Gateway component performs the task of registering the services (temperature, humidity, motion, light intensity, etc.) provided by the directly connected objects (sensors) to the coordinator component. The queries made by client applications about how to access the services, and the response by the coordinator, which includes the access information to the service, constitutes the service query phase of the architecture. At this phase, the server can respond to the access method, that is, direct access or access through the server. The data

exchange between the client and the gateway that provides the service, either directly or through the server constitutes the data transfer or data transmission phase.

3.1 Packet Header

The packet header specifies the types of packets used in the hybrid application protocol, encompassing flags, addressing, and additional payload for communication. The header details of the developed protocol are illustrated in Figure 2.

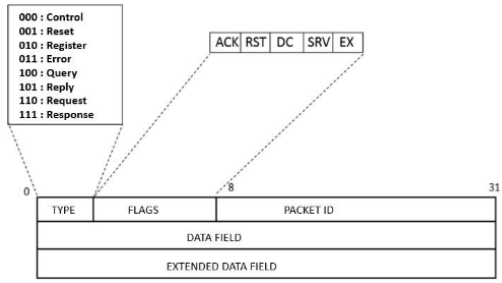

Figure 2
The packet header of the developed protocol

The header consists of 4 fixed bytes (32 bits) and data fields. The first field in the header is the "Type" field, which consists of 3 bits. Various bit sequences in this field represent values that indicate the packet type. The "Flags" is composed of a total of 5 bits. These flags store information that is necessary for communication to occur. The 24-bits "Packet Identifier" field, which ensures the uniqueness of each packet. 32-byte "Data Field" in the packet header used for transporting data in the protocol. The "Extended Data Field" has been developed to carry larger data sizes when needed. This field can carry additional payload of up to 1024 bytes, and its usage is determined based on the state of the "EX" bit (flag), which is optional.

The first bit in the 5-bit flag field is the "ACK" flag. This one-bit field is used to identify acknowledgment packets during the process of responding to packets. The second flag is used in the reset packet to change the access type of the service provided through the IoT Gateway. If this flag is marked as "1," the Gateway hardware that receives the packet loses the direct access feature provided for the service. In this case, the IoT Gateway hardware will not consider any packets sent from devices other than the coordinator. Therefore, to access the relevant service, communication through the server is mandatory. This allows for instant changes in access methods based on the network's status, as the access method is determined by the DC flag. If this flag is set, direct access to the service is

possible. The SRV bit within the flags indicates that the traffic consists of packets originating from the server. In other words, the SRV bit is set to "1" in every packet generated by the server. The use of the additional data-carrying field in the packet (Extended Data Field) is determined based on the EX-bit.

3.2 Packet Types

In the developed model, there are different types of communication occurring at each phase. These communication types are determined by the packet types of the hybrid application protocol. Each packet type corresponds to different steps in the communication process. To fulfill these steps in the protocol, there are eight packet types. The names and functions of these packets are specified in Table 1.

Table 1
Types of packets and their descriptions

Type	Value	Description
Control	0	It is used to verify whether the IoT Coordinator and IoT Gateway are accessible.
Reset	1	It is used to reset the access type of the service provided through the IoT Gateway.
Register	2	It is used during the registration process of services provided through the IoT Gateway to the IoT Coordinator server and in the response given to this registration process.
Error	3	It is the packet sent in various error situations during communication and contains an error code.
Query	4	It is the packet used by the client wishing to access data during the service discovery process, requesting access information for the service.
Reply	5	It is the response provided by the coordinator to the service query packet.
Request	6	It is the packet sent by the client to request information about the service provided at the IoT Gateway.
Response	7	It contains the data requested by the client and is sent by the IoT Gateway in response to the request packet

These types are defined in the Type field of the packet header.

3.3 Diagrams of Components

In IoT applications, the M2M (Machine-to-Machine) and M2P (Machine-to-Person) connectivity approaches involve either humans or machines as the requesting party for data. In the context of this study, the term "Client" is used to refer to the application, software, or hardware that makes data requests and can be used in both types of connections.

At the initial stage, the Client generally has two main states. If access information related to the required topic is cached in memory, it transitions directly to the *send request* state where data requests can be made without the need for further queries. Otherwise, it transitions to the *send query* state to proceed to the stage where the service will be provided. In this stage, a *query packet* is sent to the IoT Coordinator component, inquiring about how to access the desired service. If a previous query has been made regarding the topic, the records related to that query are stored in the Client's cache. In this case, the IP address of the device where the service is provided, or more precisely, the device where a specific topic is located, is obtained directly without the need for another query, ensuring minimal delay. With access information now known for the device providing the service (IoT Gateway or IoT Coordinator), it transitions to the *request state*. The query state (indicated as *send query* in the flowchart) corresponds to the stage of requesting access information from the IoT Coordinator and aligns with the Service query phase of the hybrid application protocol. After sending the query packet containing topic information, there is a waiting period for a response to the query. During this time, if no notification is received from the IoT Coordinator, the process is repeated two more times (for a total of three times). If, after the maximum number of retries, there is still no response to the query, the Client is informed that the query has not been answered, and the process comes to an end. When a query response arrives from the Coordinator, the Client retrieves the Service IP address (access information for the component providing the service) from the response, and it proceeds to the request stage. Upon request, data from the IoT Gateway or Coordinator is incorporated into the Response packets. Similar to the query state, the request is reiterated three times, with an idle time for each operation. Upon receiving the Response packet, information is extracted and transmitted to the client software. If no Response packet is received, the software is notified with a "no response" message.

The most important component in the architecture of the developed application protocol is the IoT Coordinator component. The role of the IoT Coordinator is to host the information that clients will need during the service discovery process, register resources associated with the topic provided through the IoT Gateway in the local resource directory, and ensure coordination in communication. Additionally, the IoT Coordinator is responsible for monitoring the accessibility of services within the IoT ecosystem and determining the method of data transfer, whether direct or through a server. The coordinator oversees service resources and guides the process of accessing these services. It routes client traffic to the appropriate service resource by analyzing service discovery queries from clients. This device serves not just as a resource for service discovery, but also operates as an intermediary for specific service types defined during the registration process.

The IoT Gateway is one of the critical hardware components in the IoT ecosystem. The role of the IoT Gateway component in this architecture is to receive data from objects and transform it into a format that can be used on the internet or in a network environment. The IoT Gateway can respond directly to

data requests from clients and can also respond through the server (IoT Coordinator). For each service provided through the IoT Gateway and associated with a topic, the access method is made possible through the packets sent in the Service Registration Phase. After this process is approved by the coordinator, the gateway component can respond to data transfers from clients. The IoT Gateway component uses Control packets to verify whether it is in communication with the IoT Coordinator. In the event that the Coordinator becomes unavailable or for any reason is unreachable, the IoT Gateway switches itself to direct access mode to be able to respond to Request and Query packets sent by clients and to maintain connectivity.

4 Experimental Analysis

To evaluate the performance and functionality of the protocol developed within the scope of this study, a network topology was prepared in an environment isolated from other network traffic. In this study, 10 Mbps Ethernet is simulated, MQTT and our IoT protocol are evaluated from various aspects. In topology, three different LANs are connected by a router. Devices connected to a switch act as "Publishers" devices in MQTT, and act as "IoTGW" devices for adaptive IoT protocol. Devices connected to the switch act as "Subscriber" and "Client" for MQTT and hIoT, respectively. By changing the number of devices connected to the switches, bandwidth consumption and latency values were compared for both protocols. The server acts as “Broker” in MQTT and act as “IoT Coordinator” in hIoT. In this experimental topology, the task of IoTGW is to capture hIoT packets. For this reason, the IoTGW Node Model is customized to receive only UDP messages in the OPNET simulation. In comparison, QoS-0, the fastest level of service quality for MQTT, was considered. Payload values of the packages used in the MQTT protocol are given in the Table 2. All packet sizes are fixed, except for Publish as seen in Table 2. The size of publish messages has been defined to normal distribution between 25-75 Bytes. The information of the MQTT protocol used for performance comparison was determined according to the values of the packets captured with the Wireshark software.

Table 2
MQTT Packets and Payloads

MQTT Packets	***Payload Size (Byte)***
Connect	39
Connect ACK	4
Subscribe	18
Publish	25-75
Ping Request	2
Ping Response	2

According to the MQTT protocol, the traffic is divided into phases and the payload values in each phase are created in the simulation with "*Task Configuration*". The payload values in the phases of the developed hIoT protocol are shown in Table 3. These values are taken from the fixed size header information. The transferred data size is the same size as in MQTT.

Table 3
Adaptive Hybrid IoT Packets and Payloads

Hybrid Application Protocol Phase	**Payload Size (Byte)**
Service Register	32
Service Query	32
Data Transfer	-75

4.1 Ethernet LAN Delay

The results of the Ethernet LAN delay performed in the environment of 100 nodes are shown in Figure 3. Direct methods of the hIoT protocol and the MQTT-QoS0 are compared.

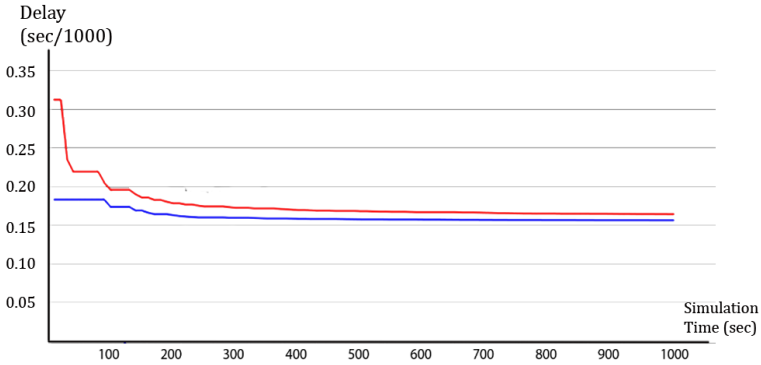

Figure 3
MQTT and hIoT Ethernet delay comparison

Compared to the delays in Ethernet, the difference between MQTT and hIoT was higher at the beginning, while the difference between this delay decreased in the following phases. There are significant differences between the delays at the beginning, because the MQTT's additional payload in the Connect phase and the response from the server also have a relatively high load amount. However, in the later stages (MQTT Publish, Subscribe and Ping), the difference between the lags has closed as these overhead values fall to an average of 2 Bytes.

4.2 Server Traffic Comparison

Performance results regarding the traffic on the servers in the topology proposed with the MQTT server are shown in Figure 4. Since the server is the center of communication in MQTT, its delay continues at a fixed value. However, in the proposed protocol, when the direct access method is used, the server only plays a role in the service discovery process. Therefore, it exhibits lower latency in the direct access method.

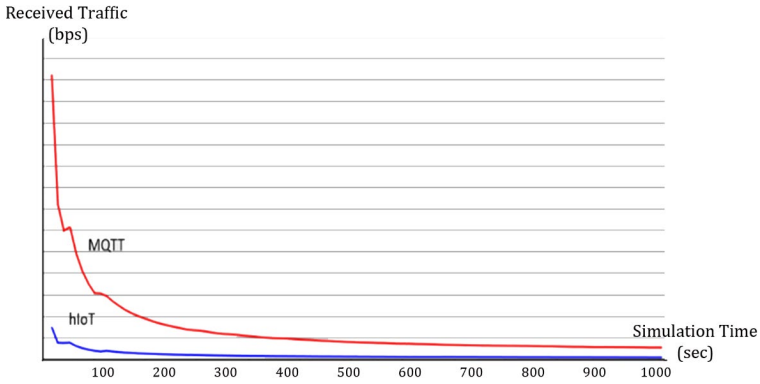

Figure 4
Server traffic comparison for MQTT and hIoT

Since the payload of the hIoT protocol is less, the number of packets received per second decreases over time. When the number of packets received by the server and the number of packets on the network is compared, it appears that the server does not receive traffic on the network after its task is completed.

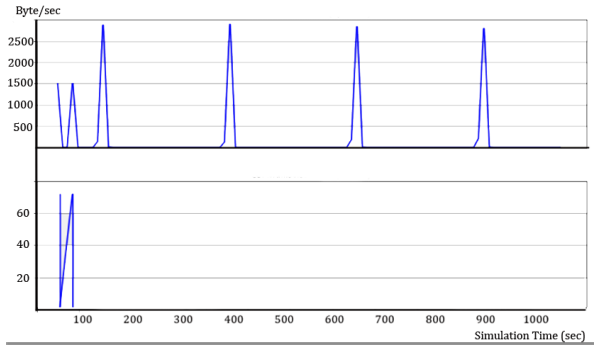

Figure 5
IoT Gateway traffic (above) and IoT Coordinator traffic (below)

A comparison of the messages received by the server and IoTGW nodes is given in Figure 5. In the simulation, one hIoT packets were sent every 5 minutes. It is seen that data transfer between IoTGW and Clients continues but the server does not have an active role in data transfer phase. Thus, the SPoF problem is reduced in the data transfer process.

4.3 Bandwidth Consumption and Functionality

Another performance comparison in the simulation is the total bandwidth usage in the campus network where the IoT Gateways, Clients and Server are modelled. The comparison results of the total throughput values in the IoTGW network with 10 nodes are shown in Figure 6.

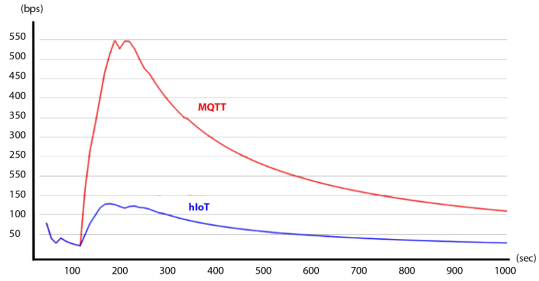

Figure 6
Bandwidth usage comparison in Publisher LAN

Bandwidth usage for MQTT and hIoT protocols decreases over time in the ethernet network where the publishers (IoTGW) are located. When looking at the total bandwidth usage in both the server network and the publisher network, it is seen that the recommended protocol consumes lower average bandwidth. Low bandwidth usage affects not only network performance but also low power consumption and extends sensor lifetime. One of the performance evaluations is related to the total bandwidth consumption by the number of devices. In a scenario with 100 Clients (MQTT Subscriber) and 100 IoT Gateways (MQTT Publisher), (all clients communicate with all gateways) 50±25 bytes of data were sent and the hIoT protocol and MQTT protocols were compared.

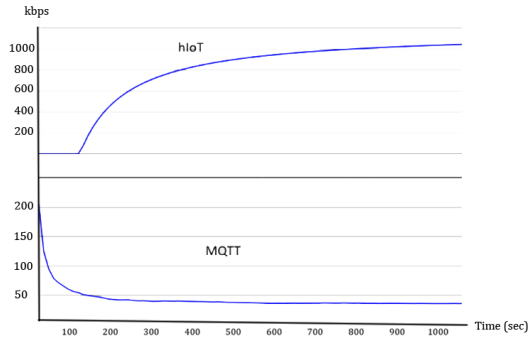

Figure 7

The number of packets to be transmitted increases in hIoT, while there is no change in MQTT

As the number of devices increases, the number of end-to-end hIoT communications increases (Figure 7). Therefore, Pub / Sub communications such as MQTT perform better in terms of bandwidth consumption. However, the risk of SPoF in server-based communication reduces functionality. To evaluate the functionality of the developed protocol, the server-based communication of the hIoT protocol and the communications of MQTT were compared. In this scenario, each client is connected with only one IoTGW. In the hIoT server-based approach, MQTT has higher performance in terms of bandwidth usage. However, in the use of the MQTT protocol, the data flow between publishers and subscribers will be interrupted in case of server failure. In the simulation, it is assumed that the server was disabled for 50 seconds.

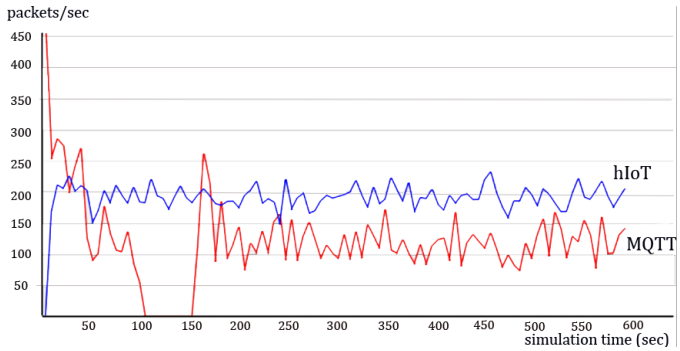

Figure 8

Comparison of packet flows per second in the event of server failure

In the scenario where the functionality of the hIoT is evaluated, it is shown in Figure 8 that the network where MQTT subscribers cannot receive any packets when the server fails. However, although it consumes higher bandwidth in direct communication, there is no packet receiving problem in the network where the hIoT protocol is used, and data transfer still occurs.

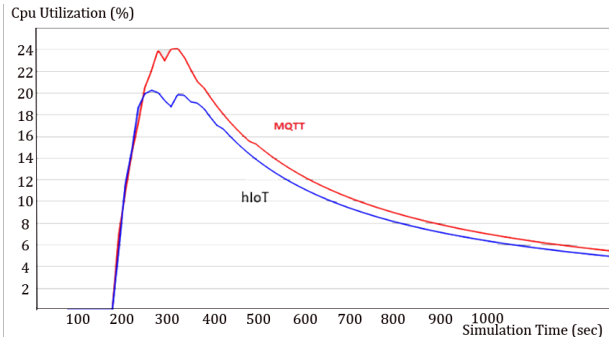

Figure 9
CPU Usage comparison

In the publisher network with 10 nodes, the evaluation of CPU usage is shown in Figure 9. Although the Adaptive Application Protocol shows less CPU usage, there is no big difference between MQTT and hIoT.

Determining the application layer protocol to be used in IoT ecosystems consisting of complex solutions is an extremely important issue in order to benefit from the advantages that the Internet of Things promises to offer. Although the applications in which IoT devices are used are similar to each other in terms of basic concepts, they may have different characteristics.

Within the scope of this study, the protocol developed in the evaluations in the simulation environment was compared with the MQTT protocol. When examined in terms of latency, it is seen that hIoT has a lower latency value than MQTT in end-to-end direct access method. However, it has been observed that server-based communications have higher latency than MQTT. In the tests carried out in the simulation environment, it is seen that the bandwidth consumption increases in the direct accesses that a hundred users want to make at random times to one hundred IoT Gateways. This can be seen as a disadvantage of hIoT. To overcome this disadvantage, it is a good practice to change access methods according to the type of services gateways provide.

One of the performance evaluations of the developed protocol is the delay times depending on the amount of load carried and the bandwidth consumed depending on the amount of load. Evaluations show that hIoT performs better at low overheads. Since it is designed to transfer data from hardware that is expected to produce data at low dimensions, such as sensors, the data transport area is limited to 1024 bytes in hIoT and the package structure has been developed accordingly.

4.4 Empirical Experiment Results

In the scope of this study, to evaluate the performance and functionality of the developed protocol, a closed network topology, as shown in Figure 10, has been set up in an environment isolated from other network traffic.

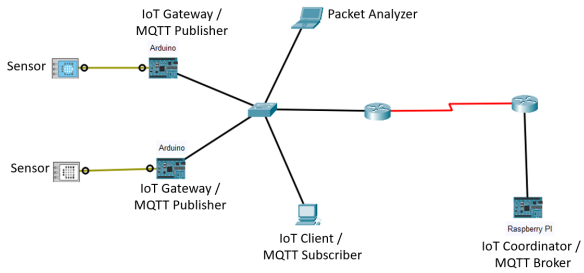

Figure 10
The topology of the experiment conducted in a real-world environment

In this study, a closed network topology as shown in Figure 9 has been designed to evaluate the performance of the developed protocol with real hardware. The Hybrid Application Protocol is compared to the MQTT protocol, commonly used in IoT applications, in terms of functionality, bandwidth usage, latency, and various other aspects. The topology includes two Arduinos, a PC, a laptop and a Raspberry PI. Additionally, a laptop computer with Wireshark software installed is used to capture and analyze all generated traffic. In the MQTT protocol, a Raspberry Pi acts as the broker server. Arduino devices function as MQTT publishers, while the PC serves as an MQTT subscriber. In the developed protocol, the Raspberry Pi device acts as the IoT Coordinator, the PC as the client component, and the Arduino as the IoT Gateway. In both scenarios, the PC and Arduino devices are connected to each other through a standard Ethernet switch that provides IEEE 802.3 Ethernet connectivity. Furthermore, two routers are connected to analyze the impact of various bandwidths. The effects are studied by altering the bandwidths between the routers. Each action performed in the real environment is repeated 30 times to obtain average latency values. The values obtained for different bandwidths are presented in the Figure 11.

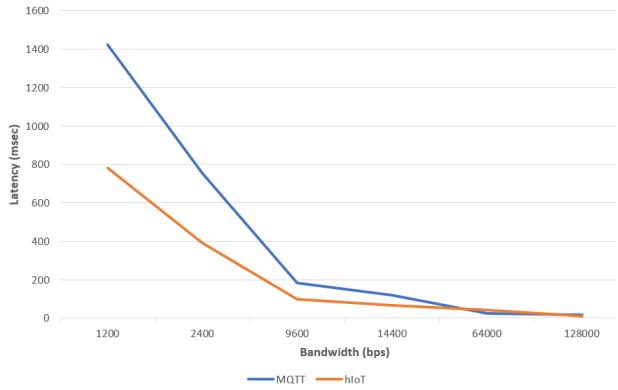

Figure 11
End-to-end delay values at different bandwidths

According to this graph, the Hybrid Application Protocol exhibits lower latency when the bandwidth used is relatively low. To assess the behavior of the developed protocol in carrying different payload sizes, latency times were evaluated based on payload sizes in the topology mentioned in Figure 9. The comparisons of average latency times obtained in the experimental study based on the transmitted data size is shown in Figure 12.

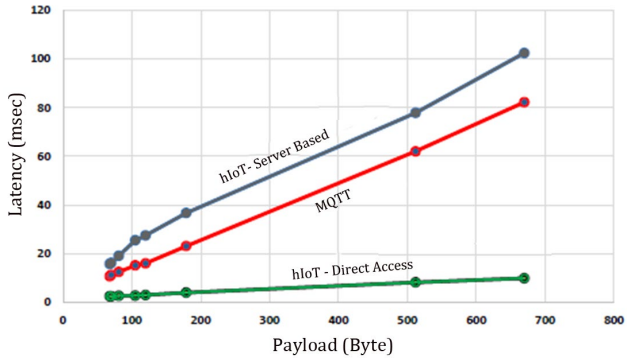

Figure 12
Latency based on different payloads on data transmission

It is observed that in the direct access method of the developed protocol, where there are no additional devices in between, the latency times are lower for each amount of data transmitted. However, in server-based data communication, additional delays occur due to the presence of the IoT Coordinator component, resulting in higher latencies compared to the MQTT protocol. In IoT ecosystems, when access time to data is critical, it appears that the direct access method would be more suitable. In applications where a certain amount of latency is acceptable

and direct access to the IoT Gateway is not desired or access control is required, server-based communication is recommended. Additionally, adaptive switching based on real-time network conditions can also be implemented. The average latency values experienced depending on bandwidths are provided in Table 5.

Table 5
Average latency values on service query

Band width (bps)	Average discovery time (msec)
1200	890
2400	384
9600	99
14400	67
64000	18
125000	15

As can be seen from the table, the average discovery time is inversely proportional to the bandwidth. In high bandwidths, this time is very short, around 15 msec, while it increases when the bandwidth decreases.

5 Results and Discussion

In Internet of Things (IoT) solutions, various protocols are used, each with their advantages in various aspects. In the scope of this study, the proposed hybrid protocol stands out due to its ability to perform both server-based and direct communication. Furthermore, in empirical and simulation experiments, it is observed that the direct access feature consumes lower bandwidth. Accordingly, in the comparison between the hIoT protocol and MQTT, it can be seen that hIoT requires lower bandwidth. The features of the protocol we have developed are summarized comparatively in the table below. The table includes a comparison with the most used MQTT and CoAP protocols.

Table 6
Comparison of the protocol from various aspects

Feature	hIoT	MQTT	CoAP
Access model	Request/Response	Pub/Sub	Request/Response
Topic Usage	Yes	Yes	No
Serive Discovery	Yes	No	Yes
L4 Protocol	UDP	TCP/UDP	UDP
Server based access	Yes	Yes	No
Direct access	Yes	No	Yes
Adaptive access	Yes	No	No

The term "adaptive access" as indicated in the table refers to the ability to switch between direct access and server-based access. One of the performance evaluations of the developed protocol is the latency depending on the amount of load carried and the bandwidth consumed depending on the amount of load. Evaluations show that hIoT performs better at low overhead. Since it is designed to transfer data from hardware that is expected to produce data in low sizes, such as sensors, the data transfer area is limited to 1024 bytes in hIoT and the packet structure has been developed accordingly.

Conclusions

The protocol developed in this study provides a new approach, to the mix of existing approaches. The protocol offers both server-based access and direct access. The ability to determine the access method of IoT Gateway devices is important. Bandwidth consumption is a critical issue in the IoT ecosystem. Protocols such as MQTT operate in the Pub/Sub architecture. In most cases, the broadcaster sends data to the subscriber periodically, even if the broadcaster does not need the data at the moment, which is a waste of bandwidth. The "on-demand access" method, is applied in the developed protocol and unnecessary bandwidth consumption is reduced. The protocol developed in simulation was compared with the MQTT protocol. When examined, in terms of latency of the two access methods offered by adaptive application protocol, direct access method is faster than MQTT, while server-based method is slower. When examined in terms of the bandwidth consumed, the adaptive application protocol consumes less bandwidth. In the evaluations carried out in the simulation environment, as the number of nodes increases, the bandwidth consumption for the direct access method also increases. Adaptive Application Protocol has both server-based communication and direct communication capability. The potential to switch between these two methods, is an important advantage.

Acknowledgement

This study was produced from the doctoral thesis titled "*General Architecture and Protocol Design For The Internet Of Things In Campus Networks*".

References

[1] A. Talaminos-Barroso, M. Estudillo-Valderrama, L. Roa, J. Reina-Tosina, and F. Ortega-Ruiz, "A Machine-to-Machine Protocol Benchmark for eHealth Applications - Use Case: Respiratory Rehabilitation," *Comput. Methods Programs Biomed.*, Vol. 129, Sep. 2016, doi: 10.1016/j.cmpb.2016.03.004

[2] L. Šikić *et al.*, "A Comparison of Application Layer Communication Protocols in IoT-enabled Smart Grid," in *2020 International Symposium ELMAR*, 2020, pp. 83-86, doi: 10.1109/ELMAR49956.2020.9219030

[3] R. Roman, J. Zhou, and J. Lopez, "On the features and challenges of security and privacy in distributed internet of things," *Comput. Networks*,

Vol. 57, No. 10, pp. 226-2279, 2013, doi: https://doi.org/10.1016/j.comnet.2012.12.018

[4] J. Dizdarević, F. Carpio, A. Jukan, and X. Masip-Bruin, "A Survey of Communication Protocols for Internet of Things and Related Challenges of Fog and Cloud Computing Integration," *ACM Comput. Surv.*, Vol. 51, No. 6, Jan. 2019, doi: 10.1145/3292674

[5] M. Aziez, S. Benharzallah, and H. Bennoui, "A Comparative Analysis of Service Discovery Approaches for the Internet of Things," *Int. Res. J. Electron. Comput. Eng. (ISSN Online 2412-4370)*, Vol. 3, p. 17, Sep. 2017, doi: 10.24178/irjece.2017.3.1.17

[6] 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, doi: 10.1016/j.future.2017.12.042

[7] B. Bendele and D. Akopian, "A study of IoT MQTT control packet behavior and its effect on communication delays," *Electron. Imaging*, vol. 2017, pp. 120-129, Sep. 2017, doi: 10.2352/ISSN.2470-1173.2017.6.MOBMU-311

[8] N. Al-Nabhan, N. Al-Aboody, and A. B. M. Alim Al Islam, "A hybrid IoT-based approach for emergency evacuation," *Comput. Networks*, Vol. 155, pp. 87-97, 2019, doi: https://doi.org/10.1016/j.comnet.2019.03.015

[9] P. Bellavista and A. Zanni, "Towards better scalability for IoT-cloud interactions via combined exploitation of MQTT and CoAP," *2016 IEEE 2nd Int. Forum Res. Technol. Soc. Ind. Leveraging a better tomorrow*, pp. 1-6, 2016 [Online] Available: https://api.semanticscholar.org/CorpusID:635592

[10] B. H. Çorak, F. Y. Okay, M. Güzel, Ş. Murt, and S. Ozdemir, "Comparative Analysis of IoT Communication Protocols," in *2018 International Symposium on Networks, Computers and Communications (ISNCC)*, 2018, pp. 1-6, doi: 10.1109/ISNCC.2018.8530963.

[11] B. Jia, L. Hao, C. Zhang, and D. Chen, "A Dynamic Estimation of Service Level Based on Fuzzy Logic for Robustness in the Internet of Things," *Sensors*, Vol. 18, No. 7, 2018, doi: 10.3390/s18072190

[12] M. O. Al Enany, H. M. Harb, and G. Attiya, "A Comparative analysis of MQTT and IoT application protocols," in *2021 International Conference on Electronic Engineering (ICEEM)* 2021, pp. 1-6, doi: 10.1109/ICEEM52022.2021.9480384

[13] M. Gill and D. Singh, "A Comprehensive Study of Simulation Frameworks and Research Directions in Fog Computing," *Comput. Sci. Rev.*, Vol. 40, No. C, May 2021, doi: 10.1016/j.cosrev.2021.100391

[14] T. Moraes, B. Nogueira, V. Lira, and E. Tavares, "Performance Comparison of IoT Communication Protocols," Sep. 2019, pp. 3249-3254, doi: 10.1109/SMC.2019.8914552

[15] T. Maksymyuk, M. Brych, S. Dumych, and H. Al-Zayadi, "Comparison of the IoT Transport Protocols Performance over Narrowband-IoT Networks," 2017 [Online] Available: https://api.semanticscholar.org/CorpusID:198233983

[16] N. Naik, "Choice of effective messaging protocols for IoT systems: MQTT, CoAP, AMQP and HTTP," in *2017 IEEE International Systems Engineering Symposium (ISSE)* 2017, pp. 1-7, doi: 10.1109/SysEng.2017.8088251

[17] N. Caro, W. Colitti, K. Steenhaut, G. Mangino, and G. Reali, "Comparison of two lightweight protocols for smartphone-based sensing," Sep. 2013, pp. 1-6, doi: 10.1109/SCVT.2013.6735994

[18] Y. Chen and T. Kunz, "Performance evaluation of IoT protocols under a constrained wireless access network," Sep. 2016, pp. 1-7, doi: 10.1109/MoWNet.2016.7496622

[19] S. Mijovic, E. Shehu, and C. Buratti, "Comparing application layer protocols for the Internet of Things via experimentation," Sep. 2016, pp. 1-5, doi: 10.1109/RTSI.2016.7740559

[20] M. Collina, M. Bartolucci, A. Vanelli-Coralli, and G. Corazza, "Internet of Things application layer protocol analysis over error and delay prone links," Sep. 2014, pp. 398-404, doi: 10.1109/ASMS-SPSC.2014.6934573

[21] A. Bhattacharyya and S. Bandyopadhyay, "Lightweight Internet Protocols for Web Enablement of Sensors Using Constrained Gateway Devices," in *Proceedings of the 2013 International Conference on Computing, Networking and Communications (ICNC)*, 2013, pp. 334-340, doi: 10.1109/ICCNC.2013.6504105

[22] H. Derhamy, J. Eliasson, and J. Delsing, "IoT Interoperability - On-Demand and Low Latency Transparent Multiprotocol Translator," *IEEE Internet Things J.*, Vol. 4, No. 5, pp. 1754-1763, 2017, doi: 10.1109/JIOT.2017.2697718

[23] C. Lee, Y. Chang, C. Chuang, and Y. H. Lai, "Interoperability enhancement for Internet of Things protocols based on software-defined network," in *2016 IEEE 5th Global Conference on Consumer Electronics*, 2016, pp. 1-2, doi: 10.1109/GCCE.2016.7800510

[24] P. Desai, A. Sheth, and P. Anantharam, "Semantic Gateway as a Service Architecture for IoT Interoperability," in *2015 IEEE International Conference on Mobile Services*, 2015, pp. 313-319, doi: 10.1109/MobServ.2015.51