

Smart Home Connectivity: Identifying the Best IoT Application Layer Protocols

Hossein Shahinzadeh

Department of Electrical Engineering,
Amirkabir University of Technology
Tehran, Iran
h.s.shahinzadeh@ieee.org

S. Mohammadali Zanjani*

Smart Microgrid Research Center,
Najafabad Branch, Islamic Azad University,
Najafabad, Iran
sma_zanjani@pel.iaun.ac.ir

Zohreh Azani

Department of Electrical Engineering
Amirkabir University of Technology
Tehran, Iran
z.azani@ieee.org

Sundus F. Al-Hameedawi

Department of Electrical Engineering
University of Shahrood
Shahrood, Iran
falihsundus@gmail.com

Saiyedeh Mehrabani-Najafabadi

Faculty of Computer Engineering,
Najafabad Branch, Islamic Azad University,
Najafabad, Iran
s.mehraban@off.iaun.ac.ir

Mohammadreza Hemmati

Faculty of Computer Engineering,
Najafabad Branch, Islamic Azad University,
Najafabad, Iran
Mr.hemmati@sco.iaun.ac.ir

Abstract— The Internet of Things (IoT) bridges the physical and digital worlds by utilizing sensors, actuators, communication technologies, computing power, and data analytics to enable precise monitoring and control of the surrounding environment. Leveraging the data derived from IoT can lead to optimal decision-making for system management. In smart homes, IoT has ushered in a new generation known as connected homes. Given the diverse range of protocols available for the IoT application layer, selecting the appropriate protocol to connect smart home devices (based on their specific requirements) to the internet gateway is a critical issue. This paper first identifies the key factors influencing the choice of application layer protocols in smart homes. Then, it examines and analyzes some of the most commonly used IoT application layer protocols, including HTTP, MQTT, CoAP, WebSocket, DDS, XMPP, AMQP, STOMP, LwM2M, Zigbee, Z-Wave, BLE, and 6LoWPAN, based on these factors. Finally, recommendations for protocol selection in various sections of a smart home are provided based on the analysis conducted.

Keywords— Smart Home, Internet of Things, Application Layer, Protocol, HTTP, MQTT, CoAP, WebSocket, DDS, XMPP, AMQP, STOMP, LwM2M, Zigbee, Z-Wave, BLE, 6LoWPAN.

I. INTRODUCTION

The Internet of Things (IoT) connects various objects around us, enabling seamless communication and service delivery. Objects within this network are equipped with sensors, actuators, and communication technologies, allowing them to receive and transmit information about their status and environmental conditions. This connectivity facilitates the collection of comprehensive and accurate data from devices and surroundings, which can then be used to make optimized decisions for system management. A smart home is equipped with a communication network, connected sensors, and appliances that can be monitored, accessed, and controlled remotely. These homes offer benefits such as enhanced security, safety, comfort, energy savings, and increased efficiency [1]. Modern smart homes, also known as connected homes, are developed based on IoT. In these homes, devices communicate with an internet-connected gateway through IoT application layer protocols. These protocols handle tasks such as facilitating data transmission from the receiver to the sender for specific applications, providing features like encryption and authentication, abstracting network details from the programmer, and formatting data at the sender's end and decoding it at the receiver's end [2]. Common IoT application layer protocols include HTTP, MQTT, CoAP, WebSocket, DDS, XMPP, AMQP, STOMP, LwM2M, Zigbee, Z-Wave, BLE, and 6LoWPAN [3]. In a smart home, various devices with different functions and specific data transmission and reception needs are present. Evaluating these needs and matching them with the appropriate application layer

protocol can significantly enhance communication quality in smart homes. Numerous studies have compared these protocols for applications like smart parking, smart cities, and the transmission of video and images from surveillance cameras [4]. However, there is limited research focused on comparing these protocols specifically for smart home applications, and those that do exist often consider only a few protocols without addressing the diverse needs of different smart home devices. In [5], the protocols are evaluated based on their efficiency, scalability, and suitability for various IoT applications. Performance metrics such as latency, throughput, and message delivery reliability under different network conditions and data payload sizes are analyzed. In [6], MQTT, WebSocket, and HTTP protocols are implemented in a Node-RED environment for smart room applications. The study compares their performance in terms of latency, throughput, and ease of integration, providing practical insights into their deployment in smart home scenarios. In [7], a detailed comparison of MQTT and CoAP protocols is provided, focusing on architectural differences, message overhead, and energy consumption. The suitability of these protocols for constrained environments typical of many IoT deployments is evaluated. In [8], the performance of MQTT, HTTP, and WebSocket protocols is examined by measuring message delivery time, reliability, and bandwidth usage in various IoT scenarios. The study provides a comparative analysis highlighting the strengths and weaknesses of each protocol. In [9], CoAP, MQTT, and HTTP protocols are evaluated in smart home environments, focusing on their efficiency and reliability, particularly in handling real-time data and ensuring secure communications, which are critical for smart home applications. In [10], HTTP, MQTT, CoAP, and XMPP protocols are compared based on performance metrics such as latency, throughput, and security features. Their suitability for different IoT use cases is assessed, providing a broad overview of their capabilities. In [11], CoAP, MQTT, AMQP, DDS, and HTTP protocols are explored for various IoT applications. The study examines factors like message size, delivery guarantees, and support for asynchronous communication, providing insights into their performance in different scenarios. In [12], the messaging protocols MQTT, CoAP, AMQP, and XMPP are compared based on their performance in handling different IoT traffic patterns. The study evaluates their suitability for real-time applications and their ability to manage data efficiently. In [13], the protocols are analyzed for their performance, scalability, and ease of deployment in smart home systems. The study focuses on energy efficiency, interoperability, and the ability to handle diverse IoT devices. In [14], MQTT and CoAP are compared for smart home automation, evaluating their performance in terms of latency, reliability, and energy consumption. Practical insights into

* Assistant Professor, Department of Electrical Engineering, Najafabad Branch, Islamic Azad University, Najafabad, Iran.

their deployment in real-world smart home environments are provided. In [15], the performance of MQTT, HTTP, and WebSocket protocols is evaluated in smart home networks by measuring response times, data transfer rates, and reliability under different network conditions. In [16], a comprehensive comparison of CoAP, MQTT, AMQP, and DDS protocols for smart home applications is provided. The study focuses on their ability to support various IoT devices, data integrity, and communication efficiency.

This paper aims to cover a broader range of application layer protocols, including HTTP, MQTT, CoAP, WebSocket, DDS, XMPP, AMQP, STOMP, LwM2M, Zigbee, Z-Wave, BLE, and 6LoWPAN. It should be noted that IoT application layer protocols are not limited to these, but these have been chosen for their prevalence and recognition. This study identifies factors that influence protocol selection for smart home applications, evaluates various smart home devices with different communication needs, and matches these needs with the appropriate protocols. The article is structured as follows: Section 2 extracts and examines the factors influencing the selection of application layer protocols for smart homes. Section 3 studies the features and compares thirteen common IoT application layer protocols. Finally, Section 4 summarizes the findings and proposes suitable protocols for different parts of the smart home.

II. FACTORS INFLUENCING THE SELECTION OF APPLICATION LAYER PROTOCOLS FOR SMART HOMES

Application layer protocols provide services to application programs, making them crucial in the communication framework of smart homes. The main objective of this section is to identify and discuss the key factors that should be considered when selecting an appropriate application layer protocol for smart home devices. These factors include:

- *Security*: Security is a critical factor, especially in smart homes where personal and sensitive data are transmitted [17]. Protocols should offer robust security features such as encryption, authentication, and data integrity to protect against cyber threats.

- *Energy Consumption*: Devices can be categorized based on their energy requirements into energy-constrained and non-energy-constrained devices. Devices connected to the main power supply, such as air conditioners, heaters, coffee makers, stoves, and televisions, fall into the non-energy-constrained category. Conversely, devices that rely on batteries or need to remain operational during power outages, like central controllers, gateways, door sensors, and smart locks, are energy-constrained [18].

- *Compatibility with Existing Devices*: The selected protocol should be compatible with existing smart home devices to ensure seamless integration and operation [19]. This avoids additional costs and complexities associated with replacing or upgrading devices.

- *Scalability*: The protocol should support the addition of new devices without significant performance degradation. Scalability is essential as smart homes expand with more devices and applications over time [20].

- *Real-Time Requirements*: Applications requiring real-time data transmission, such as fire and smoke sensors, need protocols that minimize latency to ensure prompt response and action [21].

- *High Reliability Requirements*: Some applications, such as sending control commands, need high reliability. The chosen protocol and communication network must guarantee that data is delivered accurately and timely to the intended recipient [22].

- *Bandwidth Requirements*: Depending on the application, devices may need high or low bandwidth. Generally, multimedia applications demand high bandwidth, whereas data transfer and command transmission between devices can function with lower bandwidth requirements [23].

- *Cost*: The cost of implementing the protocol, including licensing fees, deployment costs, and maintenance expenses, should be considered. Affordable protocols can promote wider adoption and implementation in various smart home scenarios [24].

III. APPLICATION LAYER PROTOCOLS IN IoT FOR SMART HOME REQUIREMENTS

This section analyzes a selection of application layer protocols used in IoT, including HTTP, MQTT, CoAP, WebSocket, DDS, XMPP, AMQP, STOMP, LwM2M, Zigbee, Z-Wave, BLE, and 6LoWPAN, focusing on the specific requirements identified in Section 2. The analysis involved reviewing official documentation from the respective standards organizations (e.g., IETF) and studying supplementary research articles on each protocol's features. This approach ensures a comprehensive understanding of how each protocol meets the necessary factors for smart home applications.

A. HTTP

The HTTP protocol currently stands as the most significant and widely used protocol on the web, supported by the majority of web browsers and web servers. In the context of a smart home system based on the IoT, HTTP plays a crucial role at the application layer. HTTP is a RESTful, request/response protocol that employs four primary commands: PUT, GET, POST, and DELETE [25]. These commands are used to create, read, update, and delete resources or files on the server. For smart home applications, this means that various IoT devices can communicate with a central server to retrieve or manipulate data, such as sensor readings, device statuses, or control commands. For instance, a smart thermostat can use HTTP GET requests to fetch the current weather data from an external server or use POST requests to update its settings based on user preferences. In a smart home environment, the use of HTTP ensures interoperability and ease of integration across various devices and platforms. Since HTTP is a stateless protocol, each request from a device to the server is processed independently, which simplifies the management of numerous IoT devices within a home network. Additionally, the widespread support for HTTP means that developers can leverage existing tools and frameworks to build and maintain smart home applications efficiently. Moreover, the security features of HTTP, such as HTTPS for encrypted communication, can be employed to ensure the secure transmission of sensitive data, such as user credentials or personal information, between IoT devices and the central server. This makes HTTP a robust and reliable choice for managing and controlling smart home devices in an IoT ecosystem [26-28].

- *Security*: HTTP alone does not provide specific security, but HTTPS can be used to ensure security. HTTPS, by utilizing SSL/TLS, enables secure and encrypted data transmission, which is essential for many applications.

- *Energy Consumption*: The HTTP protocol is not suitable for applications that require low energy consumption due to the overhead of request and response through TCP and the large header size (at least 25 bytes). Each time an HTTP request is sent, a TCP connection is established, which requires more energy. This issue is particularly problematic in devices with limited resources (such as sensors and IoT devices).

- *Compatibility with Existing Devices*: HTTP is compatible with almost all existing devices and is widely used on the web. This protocol is supported by most operating systems and devices, and numerous libraries and tools are available for working with it.

- *Scalability*: This protocol is highly scalable due to its simple structure and RESTful model. HTTP can easily be used in large environments with numerous users, and its

RESTful design makes management and development straightforward.

- *Real-Time Requirements:* Due to its dependency on TCP, this protocol is not recommended for real-time applications that require frequent connection and disconnection of TCP connections. HTTP's inherent nature of stable connections and request-response cycles does not meet real-time requirements effectively. Protocols like WebSocket or RTP are more suitable for such applications.

- *High Reliability Requirements:* HTTP, by leveraging TCP and ensuring 100% packet delivery, is considered a highly reliable protocol. Using TCP as the transport protocol ensures that data is transferred completely and without errors, which is crucial for many applications.

- *Bandwidth Requirements:* HTTP is applicable for various types of data, from textual data to video data, and does not impose specific bandwidth limitations. This protocol can handle different data volumes, but its bandwidth efficiency is not optimized and may be less efficient than more specialized protocols when transferring large data volumes.

- *Cost:* The implementation and usage cost of HTTP is generally low, and there are many tools available for its use. Many HTTP servers and clients are available for free, and using it does not require high costs.

B. MQTT

The MQTT protocol is a lightweight messaging protocol optimized for use in scenarios where bandwidth and power consumption are critical, such as with small sensors and mobile devices. Operating on top of the TCP/IP protocol, MQTT was introduced by IBM in 1999 and later standardized by OASIS in 2013, making it a cornerstone of IoT applications [29]. The protocol employs a publish/subscribe model, where devices (clients) can either publish data to a central broker or subscribe to receive data from the broker. The broker is responsible for managing the messages, ensuring they are delivered to the appropriate subscribers, making MQTT ideal for real-time communication between numerous devices. This is particularly useful in environments with unreliable networks or high-latency, where MQTT's features like Quality of Service (QoS) levels, persistent sessions, and the Last Will and Testament (LWT) ensure reliable and efficient data transmission [30].

- *Security:* MQTT includes several security mechanisms, including username/password authentication and Transport Layer Security (TLS) for encrypted communication. However, since MQTT is a lightweight protocol, it does not natively provide extensive security features like message-level encryption or intrusion detection, necessitating additional security measures for sensitive applications in smart homes.

- *Energy Consumption:* MQTT is highly energy-efficient, particularly in scenarios where devices wake up periodically to send data and then go back to sleep. The protocol's low overhead and support for QoS make it ideal for battery-powered devices in a smart home, minimizing energy consumption.

- *Compatibility with Existing Devices:* MQTT is widely supported across various platforms and devices, including many off-the-shelf IoT products used in smart homes. Its simplicity and standardized nature contribute to its compatibility, making it easy to integrate with existing devices and systems.

- *Scalability:* MQTT's publish/subscribe architecture supports a highly scalable system. It can manage numerous devices and handle a large number of messages without significant performance degradation, making it suitable for smart homes with many connected devices.

- *Real-Time Requirements:* MQTT is well-suited for real-time applications due to its low latency and efficient message delivery. The protocol's ability to operate over unreliable networks also ensures that real-time data is delivered

consistently, making it appropriate for time-sensitive smart home applications like security and lighting control.

- *High Reliability Requirements:* With its QoS levels, MQTT ensures that messages are delivered with varying degrees of reliability. This makes it suitable for smart home applications that require high reliability, such as security systems and critical automation tasks.

- *Bandwidth Requirements:* MQTT is designed to operate efficiently in environments with limited bandwidth. Its small message size and low overhead allow it to function well even in constrained network conditions, making it ideal for smart home applications where bandwidth may be limited.

- *Cost:* The MQTT protocol is open-source and free to use, making it a cost-effective solution for smart home applications. Its lightweight nature also reduces the need for high-end hardware, further lowering costs associated with implementing and maintaining smart home systems.

C. CoAP

The CoAP is specifically designed for constrained devices and networks in the IoT. Developed by the Constrained RESTful Environments (CoRE) working group, CoAP is built on top of the UDP protocol, which reduces overhead and is more suitable for low-latency, lossy networks. CoAP's RESTful architecture is similar to HTTP, utilizing familiar methods like GET, POST, PUT, and DELETE, but it is optimized for environments where resources such as bandwidth and energy are limited [31]. The protocol supports both synchronous and asynchronous message exchanges and features such as multicast support, low header overhead, and proxying/caching capabilities, making it highly efficient for IoT applications. The combination of lightweight design and asynchronous communication capabilities makes CoAP a strong choice for smart homes and other IoT scenarios where devices need to interact in a low-power, efficient manner [32].

- *Security:* CoAP includes built-in security features using DTLS (Datagram Transport Layer Security) to provide encryption, integrity, and authentication. However, as CoAP is designed for constrained environments, its security mechanisms are less robust compared to protocols like HTTPS. Additional security layers may be required to fully protect smart home systems.

- *Energy Consumption:* CoAP is highly energy-efficient, primarily due to its lightweight design and use of UDP. It is ideal for battery-operated devices in a smart home, as it minimizes the energy required for communication, enabling longer device lifespans.

- *Compatibility with Existing Devices:* CoAP's RESTful nature and its similarity to HTTP make it relatively easy to integrate with existing devices and systems. Many IoT devices that support lightweight communication protocols can use CoAP, making it a good choice for smart homes with diverse device ecosystems.

- *Scalability:* CoAP supports multicast communication and has a low overhead, allowing it to scale efficiently in networks with many devices. Its ability to handle a large number of devices with minimal resource consumption makes it suitable for smart homes as they grow and include more IoT devices.

- *Real-Time Requirements:* CoAP is designed to support real-time communication, thanks to its use of UDP and asynchronous messaging capabilities. It is well-suited for smart home applications that require immediate response times, such as security systems and real-time monitoring.

- *High Reliability Requirements:* While CoAP is designed to be lightweight and efficient, its reliance on UDP can sometimes lead to issues with packet loss, especially in unreliable networks. For critical smart home applications where high reliability is essential, CoAP might require additional mechanisms like message acknowledgments or retransmissions to ensure reliability.

- *Bandwidth Requirements:* CoAP is optimized for low bandwidth scenarios, making it an excellent choice for smart home environments where bandwidth may be constrained. Its small message size and minimal header overhead ensure that it uses bandwidth efficiently, which is critical in IoT deployments.

- *Cost:* CoAP is an open standard, and its implementation is cost-effective due to its simplicity and minimal resource requirements. This makes it an attractive option for smart home systems where cost considerations are important, particularly in large-scale deployments.

D. WebSocket

WebSocket, a communication protocol introduced in 2008 and standardized by the IETF in 2011, addresses the limitations of HTTP by enabling full-duplex communication over a single TCP connection [33]. Unlike HTTP's request/response model, WebSocket allows simultaneous, bidirectional data transmission between a client and a server, which is particularly advantageous in applications requiring real-time data exchange, such as live chat, online gaming, and IoT environments [34]. The protocol begins with an HTTP handshake and then switches to a persistent WebSocket connection, which reduces latency by avoiding the overhead of repeated HTTP requests. WebSocket's ability to maintain an open connection and its lightweight messaging format make it an efficient solution for scenarios where rapid, ongoing communication is essential [35].

- *Security:* WebSocket supports security through (WSS), the WebSocket Secure protocol, which is essentially WebSocket over TLS. This provides encryption, data integrity, and protection against various security threats like man-in-the-middle attacks. However, the persistent nature of WebSocket connections can make them more susceptible to prolonged attacks if vulnerabilities are exploited.

- *Energy Consumption:* While WebSocket is efficient in terms of bandwidth, its energy consumption can be higher compared to more constrained protocols like MQTT or CoAP, especially in devices that need to maintain a persistent connection. For battery-powered devices in smart homes, this could be a concern if the connection remains open for extended periods without significant data exchange.

- *Compatibility with Existing Devices:* WebSocket is widely supported by modern web browsers and many IoT platforms, making it highly compatible with existing devices in smart homes. Its reliance on TCP/IP also ensures it works over common network infrastructures like Wi-Fi and Ethernet.

- *Scalability:* WebSocket can handle a large number of concurrent connections, making it scalable for smart home environments with many connected devices. However, managing these connections efficiently on the server side requires robust infrastructure to avoid bottlenecks, especially as the number of devices increases.

- *Real-Time Requirements:* WebSocket excels in scenarios where real-time communication is crucial. Its low latency and bidirectional nature make it ideal for applications like live video streaming, real-time notifications, or instant control commands in smart home systems.

- *High Reliability Requirements:* WebSocket's reliance on TCP ensures reliable data transmission, as TCP handles error correction and packet reordering. This makes WebSocket suitable for applications where data integrity is critical. However, in highly unreliable networks, the connection might drop, requiring reestablishment, which could impact reliability.

- *Bandwidth Requirements:* WebSocket is efficient in terms of bandwidth usage compared to HTTP, especially for continuous data streams. The protocol's lightweight nature means that only the necessary data is transmitted without the overhead of repeated HTTP headers, making it suitable for smart home applications that require frequent data exchange.

- *Cost:* WebSocket is an open standard and is supported by many free and open-source libraries, which helps reduce the implementation cost. However, maintaining a server capable of handling numerous persistent WebSocket connections could increase operational costs, particularly if high availability and fault tolerance are required.

E. DDS

DDS is a middleware protocol and API standard designed by the Object Management Group (OMG) in 2004 for data-centric connectivity, particularly in real-time systems [36]. Widely used in aerospace, defense, industrial automation, and increasingly in IoT applications, DDS operates on a publish/subscribe model that decouples data producers (publishers) from consumers (subscribers) through a global data space. This model enables efficient, scalable communication among devices without requiring direct knowledge of each other [37]. One of DDS's key strengths lies in its extensive QoS policies, which allow fine-tuned control over data delivery aspects like timing, reliability, and resource usage, making it adaptable to various application needs. Additionally, DDS offers robust security mechanisms, including encryption, authentication, and access control, making it a suitable choice for applications that demand high reliability and data integrity [38].

- *Security:* DDS offers comprehensive security features, including encryption, authentication, and access control. This ensures that data exchanged within the smart home environment is protected against unauthorized access and tampering, making it highly suitable for applications involving sensitive information, such as health data or security systems.

- *Energy Consumption:* DDS is not specifically optimized for energy-constrained devices, which might be a consideration for battery-powered sensors or actuators in a smart home. However, its efficiency in managing data flows and the ability to fine-tune QoS settings can help optimize energy usage in specific scenarios, such as reducing transmission frequency or controlling data packet sizes.

- *Compatibility with Existing Devices:* While DDS is a powerful protocol, it may not be natively supported by many existing smart home devices, particularly consumer-grade products that are typically designed to work with more widely adopted protocols like MQTT or CoAP. Integration might require additional middleware or gateways, which could add complexity.

- *Scalability:* DDS is highly scalable, capable of supporting a large number of devices and high data throughput. Its architecture is designed to handle distributed systems with dynamic network topologies, making it suitable for smart homes with numerous interconnected devices, including those in multi-residence buildings or large estates.

- *Real-Time Requirements:* One of the core strengths of DDS is its ability to meet real-time communication requirements. With its low latency and deterministic data delivery, DDS is well-suited for smart home applications that require immediate response, such as security alarms, live video feeds, or real-time environmental monitoring.

- *High Reliability Requirements:* DDS is designed with reliability in mind, offering various QoS settings that allow for fine-tuning the level of data reliability. This includes options for ensuring data is delivered once and only once, as well as mechanisms for managing network congestion and data loss. This makes DDS ideal for critical smart home functions where data accuracy and reliability are non-negotiable.

- *Bandwidth Requirements:* DDS is efficient in its use of bandwidth, particularly in environments where large volumes of data need to be transmitted quickly and reliably. The protocol allows for the prioritization of data streams, ensuring that essential information is transmitted with minimal delay, while less critical data can be scheduled for transmission during periods of lower network activity.

- *Cost:* Implementing DDS can be more expensive compared to simpler protocols like MQTT or CoAP, both in terms of deployment and maintenance. This is due to its advanced features, scalability, and the need for potentially complex configurations. However, for applications where high performance, reliability, and security are paramount, the investment in DDS can be justified.

F. XMPP

XMPP is an open-standard communication protocol initially developed for instant messaging (IM) and presence information [39]. It operates on a decentralized client-server architecture, enabling multiple clients to connect to servers within the same domain, and allowing inter-server communication to deliver messages across different domains. This design makes XMPP highly scalable and flexible. The protocol relies on XML (Extensible Markup Language) for its message format, using "stanzas" like '`<message>`', '`<presence>`', and '`<iq>`' (info/query) to handle data transmission between clients. XMPP is real-time and supports a range of functionalities, including chat, voice and video calls, and data transfer. It operates over the TCP (Transmission Control Protocol) and supports various extensions (XEPs - XMPP Extension Protocols) to enhance its capabilities [40-41].

- *Security:* XMPP supports multiple layers of security, including TLS for encrypting communications, and Simple Authentication and Security Layer (SASL) for secure authentication. Additionally, the protocol's support for XEP-0027 (Current Jabber OpenPGP) and XEP-0384 (OMEMO Encryption) enables end-to-end encryption for messages, making it robust for smart home applications where security of transmitted data is critical.

• *Energy Consumption:* XMPP was not initially designed with energy efficiency in mind, which can be a drawback for battery-powered IoT devices in smart homes. However, efforts have been made to optimize XMPP for constrained devices, such as through the use of XEP-0313 (Message Archive Management) to reduce unnecessary transmissions. Nevertheless, it may still consume more energy compared to lightweight protocols like CoAP or MQTT.

• *Compatibility with Existing Devices:* XMPP is widely adopted and supported across a variety of platforms, making it compatible with many existing devices, particularly those used in communication applications like messaging and VoIP. Its extensibility allows it to be adapted to various IoT devices, although integration might require additional development for specific smart home use cases.

• *Scalability:* XMPP's decentralized architecture allows it to scale efficiently. Servers can handle numerous clients and can federate with other servers to distribute load and manage large-scale deployments. This makes XMPP suitable for smart homes with many interconnected devices, especially in larger or multi-building environments.

• *Real-Time Requirements:* XMPP is inherently designed for real-time communication, making it well-suited for applications requiring immediate data exchange, such as real-time monitoring, alerts, and control in smart homes. The protocol's use of TCP ensures reliable, ordered delivery of messages, which is critical for time-sensitive operations.

• *High Reliability Requirements:* XMPP is highly reliable due to its use of TCP and its ability to support features like message acknowledgments, offline message storage, and message delivery confirmations through various XEPs. This makes it appropriate for smart home systems where reliable delivery of commands and status updates is necessary.

• *Bandwidth Requirements:* XMPP's use of XML stanzas can result in relatively higher bandwidth consumption compared to more lightweight protocols. However, various extensions and optimizations can be applied to minimize bandwidth usage, such as compression techniques and the use of more efficient message formats like JSON over XML.

- *Cost:* The cost of implementing XMPP is relatively low, especially considering its open-source nature and the availability of numerous free and open-source libraries and tools. However, depending on the scale and complexity of the smart home system, there may be costs associated with maintaining and scaling the XMPP servers and customizing the protocol for specific use cases.

G. AMQP

AMQP, on the other hand, is an open standard application layer protocol designed for message-oriented middleware. Originally developed for the financial industry, AMQP is now widely used across various sectors due to its reliable, secure, and interoperable messaging features [42]. Operating over TCP, AMQP facilitates reliable communication with features like message orientation, queuing, routing, reliability, and security. It follows a client-server model, where the client sends messages to a broker, which ensures the message is delivered to the correct queue, even in the event of network disruptions. AMQP supports multiple messaging patterns, including point-to-point, publish/subscribe, and request/reply, making it a versatile choice for complex, distributed IoT systems [43].

- *Security:* AMQP provides robust security features, including TLS for encryption and SASL for authentication. These features ensure that messages are securely transmitted, making AMQP suitable for smart homes where data privacy and integrity are crucial.

• *Energy Consumption:* AMQP is more resource-intensive compared to lightweight protocols like MQTT or CoAP. Its advanced features, such as guaranteed delivery and extensive queuing mechanisms, require more processing power and memory, leading to higher energy consumption. This makes it less ideal for battery-powered IoT devices in a smart home environment.

• *Compatibility with Existing Devices:* AMQP is well-supported by many enterprise-level systems and can be integrated with existing messaging infrastructures. However, its complexity and resource requirements may limit its compatibility with more constrained devices often found in smart homes. Integration may require additional customization or middleware to bridge the gap between AMQP and simpler IoT devices.

• *Scalability:* AMQP is highly scalable and designed to handle large-scale, distributed systems. Its ability to manage complex message flows and ensure reliable delivery across multiple devices makes it suitable for smart home environments with numerous interconnected devices. However, the overhead associated with AMQP may require more powerful hardware and infrastructure as the system scales.

• *Real-Time Requirements:* While AMQP offers reliable messaging with guaranteed delivery, it is not inherently optimized for low-latency, real-time communication. The protocol's focus on reliability and message persistence can introduce delays, making it less suitable for applications requiring instantaneous responses, such as real-time monitoring and control in smart homes.

• *High Reliability Requirements:* AMQP excels in environments where high reliability is paramount. Its message queuing, persistent storage, and guaranteed delivery mechanisms ensure that messages are not lost, even in the event of network failures. This makes AMQP a strong choice for smart home applications that require high reliability, such as security systems and critical alerts.

• *Bandwidth Requirements:* AMQP's messaging overhead, including its extensive protocol headers and queuing mechanisms, results in higher bandwidth consumption compared to more lightweight protocols like MQTT or CoAP. This could be a limitation in bandwidth-constrained environments, although it may be acceptable in smart homes with robust network infrastructure.

- *Cost:* The implementation of AMQP can be costlier compared to simpler protocols due to its complexity and the need for more powerful hardware to support its advanced features. Additionally, the maintenance and scaling of an AMQP-based system may incur higher operational costs. However, for smart homes requiring reliable and secure messaging, the benefits may justify the costs.

H. STOMP

STOMP is a lightweight, simple, and text-based protocol that allows easy interaction with message brokers. Initially created to enable scripting languages like Ruby, Python, and Perl to communicate with message brokers, STOMP has evolved into a general-purpose protocol for messaging in various applications [44]. It operates over TCP, allowing clients to connect to a message broker, send messages to a destination, and subscribe to receive messages from a destination. STOMP is protocol-agnostic in terms of message content, which can be in any format (e.g., text, JSON, XML), making it flexible and suitable for scenarios that require a lightweight messaging protocol without the complexity of more feature-rich protocols like AMQP [45].

- *Security:* STOMP itself does not include advanced security features such as encryption or authentication. However, since it runs over TCP, it can leverage TLS for secure transmission. Security needs to be implemented at the application level or through the use of secure transport layers. This could be a drawback for smart home applications where security is a critical concern.

- *Energy Consumption:* STOMP is relatively lightweight, which makes it suitable for devices with limited resources. Its simplicity means it doesn't require much processing power, which helps in minimizing energy consumption. This makes STOMP a viable option for battery-powered IoT devices in smart homes.

- *Compatibility with Existing Devices:* STOMP's simplicity and text-based format make it highly compatible with a wide range of existing devices and programming environments. It can easily be integrated with various platforms, making it flexible for use in smart homes that utilize diverse hardware and software systems.

- *Scalability:* While STOMP is simple and easy to scale in small to medium environments, it lacks the advanced features required for large-scale deployments. For a smart home environment, it can scale effectively within the limits of the infrastructure, but it may struggle with very large-scale, complex systems that require more robust messaging capabilities.

- *Real-Time Requirements:* STOMP provides basic messaging capabilities and can be configured to support near real-time communication. However, it lacks the sophisticated QoS controls found in more advanced protocols, which could limit its effectiveness in scenarios requiring strict real-time performance, such as time-sensitive smart home automation tasks.

- *High Reliability Requirements:* STOMP does not natively support advanced reliability features such as message acknowledgment, persistent messaging, or guaranteed delivery. Reliability must be managed by the application or broker, making STOMP less suitable for smart home applications where high reliability is critical, such as security alerts or emergency notifications.

- *Bandwidth Requirements:* STOMP is efficient in terms of bandwidth usage due to its lightweight nature and simple text-based protocol. This efficiency makes it a good choice for smart home environments where bandwidth may be limited, especially in scenarios with multiple devices transmitting data simultaneously.

- *Cost:* STOMP is open-source and has a low overhead in terms of implementation, making it a cost-effective solution for smart home applications. The simplicity of the protocol reduces the need for complex infrastructure and development, which can further minimize costs. However,

the need to implement additional features like security and reliability at the application level might increase development costs.

I. LwM2M

LwM2M is a protocol designed by the Open Mobile Alliance (OMA) for device management and service enablement in IoT applications, specifically optimized for constrained devices, making it suitable for smart home environments [46]. It uses CoAP as its transport layer and provides a framework for remote management and monitoring of IoT devices. LwM2M supports various essential functionalities, such as device registration, management (e.g., firmware updates), and resource monitoring, ensuring efficient operation of IoT devices. Its lightweight design ensures low overhead in terms of processing, memory, and bandwidth usage, and it includes built-in security features, which are critical for maintaining data integrity and confidentiality in IoT systems [47-48].

- *Security:* LwM2M provides robust security mechanisms, including support for DTLS, which ensures secure communication between devices and the server. The protocol also includes mechanisms for authentication, encryption, and integrity protection, making it highly suitable for smart home applications where security is paramount.

- *Energy Consumption:* LwM2M is designed with energy efficiency in mind, making it ideal for devices with limited power resources, such as battery-operated sensors in smart homes. Its lightweight nature and efficient use of CoAP reduce the processing and communication overhead, which helps in minimizing energy consumption.

- *Compatibility with Existing Devices:* LwM2M's reliance on CoAP and its lightweight design make it compatible with a wide range of existing IoT devices, especially those designed for constrained environments. It can be implemented on various hardware and software platforms, ensuring broad compatibility within a smart home ecosystem.

- *Scalability:* LwM2M is scalable, supporting a large number of devices within a network. Its efficient use of resources allows it to handle the demands of a growing smart home environment, where the number of connected devices might increase over time.

- *Real-Time Requirements:* While LwM2M is not designed explicitly for real-time applications, its use of CoAP over UDP allows for relatively low-latency communication, which can meet the real-time requirements of many smart home applications, such as environmental monitoring and automation control.

- *High Reliability Requirements:* LwM2M ensures reliability through features like retransmissions, acknowledgments, and message queue management, all of which are essential for maintaining high reliability in smart home applications. These features make it suitable for scenarios where consistent and reliable data delivery is required.

- *Bandwidth Requirements:* LwM2M is optimized for low-bandwidth environments, making it ideal for smart homes where multiple devices might need to communicate simultaneously over limited network resources. Its use of CoAP further reduces bandwidth consumption by minimizing message size and using efficient encoding.

- *Cost:* LwM2M is an open standard and can be implemented with minimal licensing costs, which makes it a cost-effective solution for smart home IoT applications. Additionally, its efficiency in terms of resource usage reduces the overall operational costs, especially in large-scale deployments.

J. Zigbee

Zigbee is a wireless communication protocol designed specifically for low-power, low-data-rate, and short-range communication, making it well-suited for IoT applications,

particularly in smart homes. Introduced by the Zigbee Alliance, Zigbee operates on the IEEE 802.15.4 standard and is known for its simplicity, reliability, and scalability [49]. This protocol is often employed in scenarios where data transfer requirements are minimal, such as controlling lights, sensors, and other home automation devices. One of Zigbee's significant advantages is its ability to support mesh networking, where devices can communicate directly with each other or through intermediate devices, which allows for broad network coverage and redundancy. This mesh topology enhances the reliability of the network by enabling multiple pathways for data transmission, making the system robust against single points of failure [50].

- **Security:** Zigbee incorporates several security features, including AES-128 encryption, network-level security, and device authentication, making it reasonably secure for smart home applications. However, its security may be considered weaker compared to more robust protocols like LwM2M or AMQP, as some vulnerabilities have been identified in certain Zigbee implementations.

- **Energy Consumption:** Zigbee is designed for low power consumption, making it highly suitable for battery-operated devices in smart homes. The protocol's power-saving mechanisms, such as sleep modes and low duty cycles, help extend the battery life of connected devices, making it ideal for applications like smart lighting and sensors.

- **Compatibility with Existing Devices:** Zigbee is widely supported by a range of smart home devices, including lights, thermostats, and security systems, which makes it compatible with many existing devices. The Zigbee Alliance ensures interoperability between certified devices, which enhances its compatibility across different brands and products.

- **Scalability:** Zigbee is highly scalable, supporting up to 65,000 devices in a single network. Its mesh networking capability allows for easy addition of new devices without compromising network performance, making it suitable for expanding smart home ecosystems.

- **Real-Time Requirements:** Zigbee is not optimized for real-time applications that require high-speed data transfer. However, for many smart home applications, such as lighting control or sensor data transmission, Zigbee's latency is within acceptable limits. For real-time video or audio streaming, other protocols like WebSocket or MQTT may be more appropriate.

- **High Reliability Requirements:** The mesh networking capability of Zigbee enhances its reliability, as data can be routed through multiple paths to reach its destination. This redundancy helps in maintaining communication even if some devices in the network fail. However, Zigbee's reliance on the 2.4 GHz frequency band, which is often crowded, can lead to interference and occasional reliability issues.

- **Bandwidth Requirements:** Zigbee operates with low data rates, typically up to 250 kbps, which is sufficient for transmitting small amounts of data such as sensor readings or control signals. This makes it suitable for smart home applications that do not require high bandwidth, but it is not ideal for applications needing high data throughput.

- **Cost:** Zigbee is a cost-effective solution for smart home automation, with low implementation costs due to its open standard and widespread adoption. The availability of affordable Zigbee-compatible devices further reduces the overall cost of deploying and maintaining a smart home network based on this protocol.

K. Z-Wave

Z-Wave is another wireless communication protocol designed primarily for home automation, providing a low-power, low-latency solution for monitoring and controlling smart home devices [51]. Developed by Zensys (now owned by Silicon Labs) in 2001, Z-Wave operates on a sub-GHz frequency band (usually 908.42 MHz in the U.S. and 868.42 MHz in Europe), which helps reduce interference with Wi-Fi and Bluetooth devices that typically operate in the 2.4 GHz

band [52]. Like Zigbee, Z-Wave also supports a mesh network topology, allowing devices to communicate directly or through intermediate nodes, thereby enhancing the network's reliability and range. Z-Wave is highly scalable, supporting up to 232 devices per network, making it ideal for typical home automation scenarios [53]. This protocol is widely adopted in smart home products, such as lighting controls, security systems, and thermostats, and is known for its simplicity, reliability, and extensive device compatibility.

- **Security:** Z-Wave incorporates strong security features, including AES-128 encryption, which is standard in modern smart home communications. The introduction of the Z-Wave Security 2 (S2) framework further enhances security with additional measures like QR code authentication and secure key exchange, making it a reliable option for securing smart home devices.

- **Energy Consumption:** Z-Wave is designed to be energy-efficient, making it ideal for battery-powered devices such as sensors and smart locks. The protocol's low data rate and optimized communication mechanisms contribute to reduced energy consumption, allowing devices to operate for extended periods without frequent battery replacements.

- **Compatibility with Existing Devices:** Z-Wave has a high degree of compatibility with existing devices due to its widespread adoption in the smart home market. The Z-Wave Alliance, which oversees the standard, ensures interoperability across certified devices from different manufacturers, making it easy to integrate new devices into an existing Z-Wave network.

- **Scalability:** Z-Wave networks can support up to 232 devices, which is sufficient for most smart home applications. Its mesh networking capability allows for the expansion of the network without significant performance degradation, although large-scale deployments may face challenges if the maximum device limit is approached.

- **Real-Time Requirements:** Z-Wave is not optimized for high-speed data transmission but provides low-latency communication suitable for many smart home applications, such as lighting control and security alerts. It meets the real-time requirements for most non-critical smart home functions, though it might not be ideal for time-sensitive applications like live video streaming.

- **High Reliability Requirements:** The mesh network topology of Z-Wave enhances reliability by enabling multiple communication paths between devices. If one path fails, the network can reroute traffic through other nodes, ensuring consistent and reliable communication. Additionally, the sub-GHz frequency band reduces interference, further improving reliability.

- **Bandwidth Requirements:** Z-Wave operates at a relatively low data rate (up to 100 kbps), which is sufficient for controlling devices and transmitting small amounts of data, such as sensor readings. However, it is not suitable for applications requiring high bandwidth, such as video streaming or large data transfers.

- **Cost:** Z-Wave devices are generally more expensive than those using some other protocols like Zigbee, largely due to the proprietary nature of the technology. However, the higher cost can be justified by the protocol's reliability, security features, and extensive device compatibility, making it a worthwhile investment for users seeking a robust smart home solution.

L. BLE

BLE, also known as Bluetooth Smart, is a wireless communication protocol introduced as part of the Bluetooth 4.0 specification in 2010 [54]. BLE was specifically designed for applications that require low power consumption, allowing devices to operate on small batteries for extended periods, making it particularly suitable for IoT devices in smart homes. BLE operates in the 2.4 GHz ISM band and is widely used in various applications, including fitness trackers, smart locks, and lighting systems [55]. Unlike

classic Bluetooth, BLE is optimized for short bursts of data exchange rather than continuous data streaming, making it significantly more energy-efficient. BLE supports various communication topologies, including point-to-point, broadcast, and mesh networks, providing flexibility in how devices communicate. Additionally, BLE's backward compatibility with classic Bluetooth ensures smooth integration with existing devices, further enhancing its utility in diverse IoT environments [56].

- **Security:** BLE includes several security features such as AES-128 encryption, which ensures that data transmitted between devices is secure. BLE 4.2 and later versions introduced improved privacy features, including address randomization and enhanced pairing mechanisms, reducing the risk of tracking and unauthorized access. However, the security of BLE devices can be compromised if not implemented correctly, so proper configuration is essential.

- **Energy Consumption:** BLE is specifically designed for low energy consumption, making it one of the most energy-efficient protocols for smart home applications. Devices using BLE can run on small batteries for months or even years, depending on the usage scenario. This low power consumption makes BLE ideal for battery-operated devices like sensors and remote controls.

- **Compatibility with Existing Devices:** BLE is highly compatible with existing devices, as it is part of the broader Bluetooth standard. Most modern smartphones, tablets, and computers support BLE, enabling easy integration with various smart home devices. The backward compatibility with classic Bluetooth also allows BLE devices to communicate with a wide range of older Bluetooth devices, enhancing its utility in mixed environments.

- **Scalability:** BLE supports mesh networking, which allows for the creation of large, scalable networks by connecting multiple BLE devices in a distributed manner. This scalability makes BLE suitable for extensive smart home setups, where numerous devices need to communicate with each other. However, in very large networks, managing and maintaining connections can become complex.

- **Real-Time Requirements:** BLE provides relatively low-latency communication, which is adequate for most smart home applications like lighting control, door locks, and thermostats. While it may not be suitable for ultra-low latency requirements or high-speed data applications, BLE's performance is sufficient for the real-time control of typical smart home devices.

- **High Reliability Requirements:** BLE is reliable in environments with low to moderate interference, and its mesh networking capability adds redundancy, enhancing overall reliability. However, the 2.4 GHz band can become congested, especially in environments with many Wi-Fi and other wireless devices, which may lead to occasional communication issues. Still, BLE's low power and robust communication protocols generally ensure consistent performance.

- **Bandwidth Requirements:** BLE is optimized for low data rate applications, with a maximum throughput of around 2 Mbps in BLE 5.0. This bandwidth is sufficient for transmitting small amounts of data, such as sensor readings or control commands, but it may not be suitable for applications requiring high data rates, such as streaming video or audio.

- **Cost:** BLE is a cost-effective solution for smart home devices due to its low power requirements and widespread adoption. The cost of BLE modules and development kits is relatively low, making it accessible for manufacturers and developers. Additionally, the backward compatibility with classic Bluetooth reduces the need for new hardware investments, further lowering the overall cost.

M. 6LoWPAN

6LoWPAN stands for "IPv6 over Low-Power Wireless Personal Area Networks" and is a communication protocol

that enables the transmission of IPv6 packets over low-power, low-data-rate wireless networks, such as those defined by the IEEE 802.15.4 standard. Developed by the IETF, 6LoWPAN was designed to extend the use of IPv6 to small, low-power devices commonly found in IoT environments, such as sensors and actuators in smart homes [57]. One of the key features of 6LoWPAN is its ability to compress IPv6 headers and fragment packets to fit the smaller payload sizes of IEEE 802.15.4 frames, enabling constrained devices to communicate directly with other devices on the Internet using standard Internet protocols. This capability ensures seamless integration of IoT devices with the broader Internet, facilitating more efficient and scalable smart home ecosystems [58].

- **Security:** 6LoWPAN inherits IPv6 security features, including IPsec, which provides encryption and authentication. However, due to resource constraints, implementing these security features on low-power devices can be challenging. Additional security mechanisms, such as link-layer encryption provided by IEEE 802.15.4, are often used to secure communications. The security of 6LoWPAN-based systems largely depends on the underlying implementations and additional security layers that are applied.

- **Energy Consumption:** 6LoWPAN is designed for low-power operations, making it suitable for battery-powered IoT devices in a smart home. The protocol optimizes energy consumption by using lightweight headers and efficient data compression techniques, reducing the amount of data that needs to be transmitted. This ensures that devices can operate for long periods without frequent battery replacements.

- **Compatibility with Existing Devices:** 6LoWPAN enables IoT devices to communicate directly with IPv6 networks, which means it can integrate well with existing IP-based infrastructure. Devices using 6LoWPAN can be part of a larger IPv6 network, making it highly compatible with modern networking technologies. However, it may require a gateway or bridge to communicate with older IPv4 networks or non-IP-based devices.

- **Scalability:** 6LoWPAN supports scalable networks by allowing a large number of devices to be interconnected. It supports mesh networking, where devices can relay data through other devices, extending the network's range and robustness. This scalability is particularly beneficial in smart home environments where multiple sensors, actuators, and controllers need to communicate effectively.

- **Real-Time Requirements:** 6LoWPAN can meet basic real-time requirements, but it may not be ideal for applications that require ultra-low latency. The protocol's low power and low data rate characteristics may introduce some delays, making it suitable for non-critical real-time applications like environmental monitoring or home automation where slight delays are acceptable.

- **High Reliability Requirements:** The reliability of 6LoWPAN is enhanced by its support for mesh networking, which allows multiple communication paths between devices. This redundancy improves the reliability of the network by ensuring that data can still be transmitted even if some nodes fail. However, in environments with high interference or high-density networks, reliability may be affected.

- **Bandwidth Requirements:** 6LoWPAN is designed for low data rate applications, with typical data rates around 250 kbps as defined by IEEE 802.15.4. This bandwidth is sufficient for many IoT applications in smart homes, such as sensor data transmission or control commands. However, it may not be suitable for applications requiring high bandwidth, such as streaming video or large data transfers.

- **Cost:** 6LoWPAN is cost-effective due to its use of low-power, low-cost hardware. The protocol is designed to operate on inexpensive microcontrollers with limited processing power and memory, which helps to keep the overall cost of devices low. Additionally, the ability to

integrate seamlessly with existing IPv6 networks reduces the need for specialized networking infrastructure, further lowering deployment costs.

Table 1 summarizes the results of the protocol evaluations discussed in this section.

Table I. COMPARISON OF APPLICATION LAYER PROTOCOLS IN SMART HOME ENVIRONMENTS

Protocol	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)
HTTP	✓	✗	✓	✗	✗	✗	✗	✗
MQTT	✓	✓	✓	✓	✓	✓	✓	✓
CoAP	✓	✓	✓	✓	✓	✓	✓	✓
WebSocket	✓	✗	✓	✓	✓	✓	✓	✓
DDS	✓	✗	✓	✓	✓	✓	✓	✗
XMPP	✓	✗	✓	✗	✓	✓	✓	✗
AMQP	✓	✗	✓	✓	✓	✓	✓	✗
STOMP	✓	✗	✓	✓	✓	✓	✓	✓
LwM2M	✓	✓	✓	✓	✓	✓	✓	✓
Zigbee	✓	✓	✓	✓	✓	✓	✓	✓
Z-Wave	✓	✓	✓	✓	✓	✓	✓	✓
BLE	✓	✓	✓	✓	✓	✓	✓	✓
6LoWPAN	✓	✓	✓	✓	✓	✓	✓	✓
(1): Security	(5): Real-Time Requirements							
(2): Energy Consumption	(6): High Reliability Requirements							
(3): Compatibility with Existing Devices	(7): Bandwidth Requirements							
(4): Scalability	(8): Cost							

IV. SELECTING THE APPROPRIATE PROTOCOLS FOR DIFFERENT SMART HOME COMPONENTS

In this section, a selection of smart home devices, including Camera, Smart Door Lock, Fire Extinguishing System, Air Conditioning System, Lighting System, Watering the Pot, Smart Curtain, Switch/Socket/Relay, Smart Thermostat, and Motion and Presence Sensors, are chosen for analysis. The communication requirements of these devices are evaluated based on the factors identified in Section 2 of this paper. While the selected devices do not cover all possible smart home devices, an effort has been made to include at least one device from each category with varying communication needs. The communication requirements for these smart home devices are summarized in Table 2.

Table II. FACTORS FOR SELECTING APPLICATION LAYER PROTOCOLS FOR SMART HOME DEVICES

Device	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)
Camera	✓	✗	✓	✓	✓	✓	✗	✗
Smart Door Lock	✓	✓	✓	✓	✓	✓	✓	✓
Fire Extinguishing System	✓	✓	✓	✓	✓	✓	✓	✓
Air Conditioning System	✓	✗	✓	✓	✓	✓	✗	✗
Lighting System	✓	✓	✓	✓	✓	✓	✓	✓
Watering the Pot	✓	✓	✓	✓	✓	✓	✓	✓
Smart Curtain	✓	✓	✓	✓	✓	✓	✓	✓
Switch/Socket/Relay	✓	✓	✓	✓	✓	✓	✓	✓
Smart Thermostat	✓	✓	✓	✓	✓	✓	✓	✓
Motion Sensors	✓	✓	✓	✓	✓	✓	✓	✓
(1): Security	(5): Real-Time Requirements							
(2): Energy Consumption	(6): High Reliability Requirements							
(3): Compatibility with Existing Devices	(7): Bandwidth Requirements							
(4): Scalability	(8): Cost							

In Section 3 of this paper, thirteen application layer protocols were analyzed and compared based on the factors identified in Section 2. Subsequently, based on the requirements of the selected smart home devices and their alignment with the features of the protocols, efforts are made to identify suitable protocols for each device.

• *Cameras*: Cameras in smart home environments often require protocols that can handle significant data loads due to the high-resolution video streams they produce. The need for high bandwidth is paramount to ensure smooth, real-time video transmission. Protocols like MQTT and CoAP, while typically used for lightweight messaging, can be adapted for video data through efficient compression techniques, although they might not always be ideal for this purpose due to their original design for smaller data packets. WebSocket, on the other hand, is well-suited for continuous, bi-directional communication, making it highly effective for real-time

video feeds. DDS is particularly robust for handling large-scale, real-time data distribution, making it an excellent choice for video surveillance systems that require high reliability and low latency across multiple devices.

• *Smart Door Locks*: Security is the foremost concern when it comes to smart door locks. These devices need to ensure that communication is encrypted, secure, and resistant to interference or hacking attempts. MQTT is a lightweight protocol that supports secure transmission through TLS, making it a good choice for smart door locks that require secure communication with minimal overhead. Zigbee and Z-Wave are specifically designed for low-power, secure communication in home automation systems, with strong encryption standards (like AES-128) to prevent unauthorized access. BLE also supports secure connections through encryption and pairing mechanisms, making it a reliable choice for devices where both security and low power consumption are important.

• *Fire Extinguishing Systems*: For fire extinguishing systems in a smart home, reliability and real-time communication are non-negotiable. These systems must react instantly to emergency signals to prevent catastrophic damage. MQTT, with its support for QoS levels, ensures that messages are delivered reliably, even in the case of network failures. CoAP is designed for IoT devices and supports reliable messaging through its confirmable messages feature, which ensures that messages are acknowledged by the recipient. LwM2M (Lightweight Machine to Machine) is optimized for device management and monitoring in constrained environments, making it ideal for fire extinguishing systems that need to reliably communicate status updates and receive remote commands in real-time.

• *Air Conditioning Systems*: Air conditioning systems in smart homes benefit from protocols that support scalability and real-time responsiveness, particularly in larger setups with multiple units or zones. MQTT's lightweight nature and support for efficient message routing through its broker-based architecture make it suitable for managing multiple air conditioning units. DDS, known for its high performance in distributed environments, is ideal for real-time control of air conditioning systems, ensuring that temperature adjustments and system updates occur without noticeable delays, even in complex setups involving many interconnected devices.

• *Lighting Systems*: In smart home lighting systems, energy efficiency and scalability are crucial, especially in homes with numerous light fixtures that need to be controlled individually or in groups. Zigbee and Z-Wave are both highly energy-efficient, making them ideal for battery-powered lighting devices. Their mesh networking capabilities allow for the seamless extension of the network, ensuring that even lights far from the central hub remain responsive. BLE, while not originally designed for large-scale networks, has been adapted for smart lighting systems due to its low power consumption and sufficient range for smaller setups.

• *Watering the Pot*: Smart watering systems for plants require protocols that prioritize low energy consumption and compatibility with a variety of sensors and control units. MQTT is a strong contender due to its ability to function efficiently on low-power devices, ensuring that the system can run for extended periods without frequent battery changes. Zigbee, with its low-power mesh networking capability, ensures that communication between the watering system and the central hub is reliable, even over larger garden areas or in setups with multiple sensors.

• *Smart Curtains*: Smart curtains share similar requirements with lighting systems, where energy efficiency and reliable wireless communication are key. Zigbee and Z-Wave stand out as top choices due to their ability to create robust, low-power mesh networks that ensure consistent performance across various smart devices in the home. These protocols also support the necessary responsiveness and integration required for seamless operation with other smart home systems, such as lighting and security.

- Switch/Socket/Relay:** Smart switches, sockets, and relays need to be both energy-efficient and compatible with a wide range of devices. Zigbee and Z-Wave, with their low energy requirements and strong device compatibility, are well-suited for these applications. BLE can also be considered, particularly in smaller setups where low power consumption and direct device communication are priorities.

- Smart Thermostat:** For smart thermostats, scalability and real-time performance are critical to ensuring that temperature adjustments happen quickly and efficiently across the entire home. MQTT, with its lightweight messaging and support for real-time data exchange, is well-suited for these devices. Zigbee, known for its low power consumption and reliable mesh networking, ensures that the thermostat can communicate effectively with both the central hub and other devices in the home, maintaining optimal temperature settings.

- Motion and Presence Sensors:** Motion and presence sensors in smart homes require protocols that offer real-time communication and high reliability, ensuring that the sensors can instantly trigger responses, such as turning on lights or alerting security systems. MQTT, with its low latency and reliable message delivery, is an excellent choice for these sensors. CoAP, designed for low-power devices, also provides the necessary real-time responsiveness and reliability. Zigbee's mesh networking further enhances reliability by ensuring that sensor signals can always reach their intended destination, even in complex home layouts.

A summary of the analysis for protocol selection is provided in Table 3.

Table III. APPLICATION LAYER PROTOCOL SELECTION FOR SMART HOME DEVICES

Protocol	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)	(10)
HTTP	x	x	x	x	x	x	x	x	x	x
MQTT	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
CoAP	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
WebSocket	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
DDS	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
XMPP	✓	x	x	✓	x	x	x	x	x	x
AMQP	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
STOMP	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
LwM2M	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Zigbee	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Z-Wave	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
BLE	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
6LoWPAN	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
(1): Camera	(6): Watering the Pot									
(2): Smart Door Lock	(7): Smart Curtain									
(3): Fire Extinguishing System	(8): Switch/Socket/Relay									
(4): Air Conditioning System	(9): Smart Thermostat									
(5): Lighting System	(10): Motion Sensors									

Based on Table 3, it can be concluded that the MQTT stands out as a highly adaptable protocol in the realm of smart home devices, largely due to its low energy consumption, scalability, and robust real-time capabilities. This makes it suitable for a wide range of devices, from high-bandwidth applications like cameras to low-energy systems like smart thermostats and lighting. Its efficiency in handling various types of communication, from simple commands to more complex data streams, makes it a universal choice for diverse smart home environments. Meanwhile, Zigbee, Z-Wave, and BLE are particularly advantageous for devices where low power consumption and the ability to scale across multiple devices are crucial. Their mesh networking capabilities further enhance their compatibility, allowing for seamless integration and reliable operation across various smart home systems. In contrast, HTTP, despite its widespread use in general web applications, is less suitable for most smart home devices. Its higher energy consumption and lack of inherent real-time capabilities make it inefficient for the quick, low-power operations typically required in a smart home setting. On the other hand, protocols like CoAP and LwM2M are designed with low power and efficient bandwidth usage in mind, making them well-suited for devices that need to operate on minimal energy while still maintaining reliable

communication. These protocols excel in environments where lightweight communication is critical, ensuring that even the most constrained devices can function effectively within a smart home ecosystem. Overall, the choice of protocol in a smart home setup should be carefully aligned with the specific energy, scalability, and real-time requirements of each device.

V. CONCLUSION

In modern smart homes, IoT communication technologies are used to establish connections between various home components and the external environment. In the IoT protocol stack, different application layer protocols have been introduced, each with distinct characteristics suitable for various applications. To optimize communication efficiency in a smart home, it is crucial to select the appropriate protocol for each device based on its specific requirements. This paper initially reviewed related works on the topic, then studied the influencing factors for protocol selection. Subsequently, several widely used IoT application layer protocols were analyzed and compared in terms of fulfilling the specified requirements for smart homes. In the fourth section of the paper, the focus was on matching the protocols with the requirements of smart home devices. For each smart home device, key factors were identified based on its requirements (as shown in Table 2). Then, by aligning the features of application layer protocols with the requirements of each device, recommended protocols were identified, with a summary provided in Table 3. The analysis results indicated that MQTT is the most versatile protocol for smart home devices due to its low energy consumption, scalability, and real-time performance, making it ideal for nearly all applications. Zigbee, Z-Wave, and BLE are also highly compatible, particularly for devices where low power and scalability are essential. CoAP and LwM2M are optimal for low-power devices with efficient bandwidth requirements.

REFERENCES

- [1] Gupta, Rajesh, et al. "Blockchain and AI-based secure onion routing framework for data dissemination in IoT environment underlying 6g networks." *2022 Sixth International Conference on Smart Cities, Internet of Things and Applications (SCIoT)*. IEEE, 2022.
- [2] Hosseiniyan, Heliasadat, et al. "Blockchain outlook for deployment of IoT in distribution networks and smart homes." *International Journal of Electrical and Computer Engineering* 10.3 (2020): 2787.
- [3] Bang, A. O., Rao, U. P., & Husari, A. A. (2022). A Comprehensive Study of Security Issues and Research Challenges in Different Layers of Service-Oriented IoT Architecture. *Cyber Security and Digital Forensics*, 1-43.
- [4] Shahinzaadeh, G., Shahinzaadeh, H., & Tanwar, S. (2024, May). Security and Privacy Issues in the Internet of Things: A Comprehensive Survey of Protocols, Standards, and the Revolutionary Role of Blockchain. In *2024 8th International Conference on Smart Cities, Internet of Things and Applications (SCIoT)* (pp. 59-67). IEEE.
- [5] Al-Masri, Eyhab, et al. "Investigating messaging protocols for the Internet of Things (IoT)." *IEEE Access* 8 (2020): 94880-94911.
- [6] Kaur, S., & Khanna, V. (2022). Implementation and comparison of MQTT, WebSocket, and HTTP protocols for smart room iot application in node-RED. In *IoT for Sustainable Smart Cities and Society* (pp. 165-193). Cham: Springer International Publishing.
- [7] Seoane, V., Garcia-Rubio, C., Almenares, F., & Campo, C. (2021). Performance evaluation of CoAP and MQTT with security support for IoT environments. *Computer Networks*, 197, 108338.
- [8] Gemirter, C. B., Şenturca, C., & Baydere, S. (2021, September). A comparative evaluation of AMQP, MQTT and HTTP protocols using real-time public smart city data. In *2021 6th International Conference on Computer Science and Engineering (UBMK)* (pp. 542-547). IEEE.
- [9] Bansal, M., & Priya. (2021). Performance comparison of MQTT and CoAP protocols in different simulation environments. *Inventive Communication and Computational Technologies: Proceedings of ICICCT 2020*, 549-560.
- [10] Nikolov, N. (2020, September). Research of MQTT, CoAP, HTTP and XMPP IoT communication protocols for embedded systems. In *2020 XXIX International Scientific Conference Electronics (ET)* (pp. 1-4). IEEE.
- [11] Seleznev, S., & Yakovlev, V. (2019). Industrial Application Architecture IoT and protocols AMQP, MQTT, JMS, REST, CoAP,

- XMPPI, DDS. *International Journal of Open Information Technologies*, 7(5), 17-28.
- [12] Naik, N. (2017, October). Choice of effective messaging protocols for IoT systems: MQTT, CoAP, AMQP and HTTP. In *2017 IEEE international systems engineering symposium (ISSE)* (pp. 1-7). IEEE.
- [13] Yan, Wenyao, et al. "Survey on recent smart gateways for smart home: Systems, technologies, and challenges." *Transactions on Emerging Telecommunications Technologies* 33.6 (2022): e4067.
- [14] Seoane, Victor, et al. "Performance evaluation of CoAP and MQTT with security support for IoT environments." *Computer Networks* 197 (2021): 108338.
- [15] Bayılmış, C., Ebleme, M. A., Çavuşoğlu, Ü., Küçük, K., & Sevin, A. (2022). A survey on communication protocols and performance evaluations for Internet of Things. *Digital Communications and Networks*, 8(6), 1094-1104.
- [16] Sidna, Jeddou, et al. "Analysis and evaluation of communication Protocols for IoT Applications." *Proceedings of the 13th international conference on intelligent systems: theories and applications*. 2020.
- [17] Touqeer, Haseeb, et al. "Smart home security: challenges, issues and solutions at different IoT layers." *The Journal of Supercomputing* 77.12 (2021): 14053-14089.
- [18] Andrade, Sérgio HMS, et al. "A smart home architecture for smart energy consumption in a residence with multiple users." *IEEE Access* 9 (2021): 16807-16824.
- [19] Phan, L. A., & Kim, T. (2020). Breaking down the compatibility problem in smart homes: A dynamically updatable gateway platform. *Sensors*, 20(10), 2783.
- [20] Plantevin, Valère, et al. "Towards a more reliable and scalable architecture for smart home environments." *Journal of Ambient Intelligence and Humanized Computing* 10 (2019): 2645-2656.
- [21] Hui, T. K., Sherratt, R. S., & Sánchez, D. D. (2017). Major requirements for building Smart Homes in Smart Cities based on Internet of Things technologies. *Future Generation Computer Systems*, 76, 358-369.
- [22] Safaei, Bardia, et al. "Reliability side-effects in Internet of Things application layer protocols." *2017 2nd International Conference on System Reliability and Safety (ICSRS)*. IEEE, 2017.
- [23] Islam, Md Milon, et al. "Internet of things: Device capabilities, architectures, protocols, and smart applications in healthcare domain." *IEEE Internet of Things Journal* 10.4 (2022): 3611-3641.
- [24] Sahrab, A. A., & Marhoon, H. M. (2022). Design and fabrication of a low-cost system for Smart Home Applications. *Journal of Robotics and Control (JRC)*, 3(4), 409-414.
- [25] da Cruz, Mauro AA, et al. "A proposal for bridging application layer protocols to HTTP on IoT solutions." *Future Generation Computer Systems* 97 (2019): 145-152.
- [26] Gupta, B. B., & Quamara, M. (2020). An overview of Internet of Things (IoT): Architectural aspects, challenges, and protocols. *Concurrency and Computation: Practice and Experience*, 32(21), e4946.
- [27] Tripathi, Shivam, et al. "IoMT-Enabled Smart Healthcare: State-of-the-Art, Security and Future Directions." *2023 14th International Conference on Information and Knowledge Technology (IKT)*. IEEE, 2023.
- [28] Lombardi, M., Pascale, F., & Santaniello, D. (2021). Internet of things: A general overview between architectures, protocols and applications. *Information*, 12(2), 87.
- [29] Patni, Hrishita, et al. "SmartGuardML: ML-based MQTT Data Analysis Approach for Threat Prediction in Smart Homes." *2024 8th International Conference on Smart Cities, Internet of Things and Applications (SCIoT)*. IEEE, 2024.
- [30] Zanjani, S. Mohammadali, et al. "Securing the internet of things via blockchain-aided smart contracts." *2022 13th International Conference on Information and Knowledge Technology (IKT)*. IEEE, 2022.
- [31] Bhattacharjya, Aniruddha, et al. "CoAP—application layer connection-less lightweight protocol for the Internet of Things (IoT) and CoAP-IPSEC Security with DTLS Supporting CoAP." *Digital twin technologies and smart cities* (2020): 151-175.
- [32] Dave, Madhavi, Jyotika Doshi, and Harshal Arolkar. "MQTT-CoAP interconnector: IoT interoperability solution for application layer protocols." *2020 Fourth International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud)(I-SMAC)*. IEEE, 2020.
- [33] Oliveira, Guilherme MB, et al. "Comparison between MQTT and WebSocket protocols for IoT applications using ESP8266." *2018 Workshop on Metrology for Industry 4.0 and IoT*. IEEE, 2018.
- [34] Saritha, S., & Sarasvathi, V. (2017, December). A study on application layer protocols used in IoT. In *2017 International Conference on Circuits, Controls, and Communications (CCUBE)* (pp. 155-159). IEEE.
- [35] Thakur, Rohit Kumar, and Raj Kumari. "A Comparison of Various IoT Application Layer Protocol." *American Journal of Electronics & Communication* 3.1 (2022): 28-34.
- [36] Nebbione, G., & Calzarossa, M. C. (2020). Security of IoT application layer protocols: Challenges and findings. *Future Internet*, 12(3), 55.
- [37] Donta, Praveen Kumar, et al. "Survey on recent advances in IoT application layer protocols and machine learning scope for research directions." *Digital Communications and Networks* 8.5 (2022): 727-744.
- [38] Jienan, D., Xiangning, C., & Shuai, C. (2021, April). Overview of Application Layer Protocol of Internet of Things. In *2021 IEEE 6th International Conference on Computer and Communication Systems (ICCCS)* (pp. 922-926). IEEE.
- [39] Iqbal, F., Akhtar, S. M., & Anwar, R. (2021). A survey of application layer protocols of Internet of Things. *International Journal of Computer Science & Network Security*, 21(11), 301-311.
- [40] Bansal, Malti. "Application layer protocols for internet of healthcare things (IoHT)." *2020 fourth international conference on inventive systems and control (ICISC)*. IEEE, 2020.
- [41] Zanjani, S. Mohammadali, et al. "Big data analytics in iot with the approach of storage and processing in blockchain." *2022 6th Iranian Conference on Advances in Enterprise Architecture (ICAEA)*. IEEE, 2022.
- [42] Al Enany, Marwa O., Hany M. Harb, and Gamal Attiya. "A Comparative analysis of MQTT and IoT application protocols." *2021 International Conference on Electronic Engineering (ICEEM)*. IEEE, 2021.
- [43] Amjad, Anam, et al. "A systematic review on the data interoperability of application layer protocols in industrial IoT." *Ieee Access* 9 (2021): 96528-96545.
- [44] Bhowmik, R., & Riaz, M. H. (2023). An extended review of the application layer messaging protocol of the internet of things. *Bulletin of Electrical Engineering and Informatics*, 12(5), 3134-3141.
- [45] Win, Su Pyae, Win Lelt Lelt Phy, and Tin Zar Thaw. "Shedding Light on Technological Treasures within IoT Application Layer Protocol." *2023 IEEE Conference on Computer Applications (ICCA)*. IEEE, 2023.
- [46] Basu, Subho Shankar, et al. "An end-to-end LwM2M-based communication architecture for multimodal NB-IoT/BLE devices." *Sensors* 20.8 (2020): 2239.
- [47] Pappalardo, M., Virdis, A., & Mingozzi, E. (2022). An Edge-Based LWM2M Proxy for Device Management to Efficiently Support QoS-Aware IoT Services. *IoT*, 3(1), 169-190.
- [48] Dvorak, Radim, et al. "LWM2M for Cellular IoT: Protocol Implementation and Performance Evaluation." *2023 15th International Congress on Ultra Modern Telecommunications and Control Systems and Workshops (ICUMT)*. IEEE, 2023.
- [49] Zohourian, Alireza, et al. "IoT Zigbee device security: A comprehensive review." *Internet of Things* 22 (2023): 100791.
- [50] Dymora, P., Mazurek, M., & Smalara, K. (2021). Modeling and fault tolerance analysis of zigbee protocol in iot networks. *Energies*, 14(24), 8264.
- [51] Lilli, M., Braghin, C., & Riccobene, E. (2021). Formal Proof of a Vulnerability in Z-Wave IoT Protocol. In *SECRYPT* (pp. 198-209).
- [52] Kim, Kyoonggon, et al. "What's your protocol: Vulnerabilities and security threats related to Z-Wave protocol." *Pervasive and Mobile Computing* 66 (2020): 101211.
- [53] Braghin, C., Lilli, M., & Riccobene, E. (2023). A model-based approach for vulnerability analysis of IoT security protocols: The Z-Wave case study. *Computers & Security*, 127, 103037.
- [54] Spachos, P., & Plataniotis, K. (2020). BLE beacons in the smart city: Applications, challenges, and research opportunities. *IEEE Internet of Things Magazine*, 3(1), 14-18.
- [55] Lacava, Andrea, et al. "Securing Bluetooth Low Energy networking: An overview of security procedures and threats." *Computer Networks* 211 (2022): 108953.
- [56] Ghori, M. R., Wan, T. C., & Sodhy, G. C. (2020). Bluetooth low energy mesh networks: Survey of communication and security protocols. *Sensors*, 20(12), 3590.
- [57] Thungon, Leki Chom, et al. "A Survey on 6LoWPAN Security for IoT: Taxonomy, Architecture, and Future Directions." *Wireless Personal Communications* (2024): 1-45.
- [58] Abood, Albahlool M., Waleed K. Hasan, and Haitham Khaled. "6LoWPAN-Technical Features and Challenges in IoT: A Review." *2024 IEEE 4th International Maghreb Meeting of the Conference on Sciences and Techniques of Automatic Control and Computer Engineering (MI-STAC)*. IEEE, 2024.