MESSAGE BROADCAST METHODS FOR THE INTERNET OF THINGS

Jan Čapek, Dawit Dejene Bikila

University of Pardubice
Faculty of Economics and Administration
capek@upce.cz, dawitdejene.bikila@student.upce.cz

DOI: 10.35011/IDIMT-2024-257

Keywords

Internet of Things (IoT); messaging techniques; message queuing; communication models

Abstract

Ambient Intelligence and the Internet of Things (IoT) have shown noticeable growth in recent years in changing people's daily lifestyles. The Internet of Things creates a vast network of interconnected devices, sensors, and actuators that can collect, transmit, analyse, and use data to make intelligent decisions and take action. We can say that the IoT is the materialisation or manifestation of AmI that integrates this imaginary world with the real world through a common platform. IoT organises an environment which can be considered intelligent and independent, such as smart cities, things, health, or even intelligent life. The IoT communication model refers to several ways, in which IoT devices connect and share data. In this regard, we have observed from the literature that only the two methods have been commonly used for IoT messaging. Hence, in this work, we propose a messaging broadcast system using Producer-Consumer and Reader-Writer methods. A study with discussion and analysis provided and indicates the reasonability and applicability of this proposed approach in IoT-based smart environments.

1. Introduction

Ambient Intelligence (AmI) and IoT have shown a noticeable growth in recent years that changes people's daily lifestyles. AmI is an environment that uses technology to create intelligence and intelligent environments that are aware of our needs and can respond to them automatically (Anastasopoulos, M. et al., 2005; Kissoum, Y. et al., 2014). The Internet of Things (IoT) can be denoted using the following general representation: IoT = Data (or information) + Sensors + Networks + Services. We can say that the IoT is the materialisation or manifestation of AmI. This makes our physical surroundings be seamless and personalised environment.

IoT devices are usually based on affordable wireless communication interfaces through which they can communicate and transmit information to other IoT devices or a centralised system. With the rapid development of these technologies, everyday routines of life are centred on virtual space, a virtual world as indicated by Razzaq, M. A. et al. (2017). As a result, users can then shop, work, plant, or keep animals in this virtual world while physically living in the real world.

The interconnection and interaction of IoT devices have altered the way we live by providing increased flexibility and convenience through IoT applications. With the rapid growth and a huge number of devices connecting to IoT, 75 billion things are expected to be connected by 2025. As a result, a large volume of real-time data will be generated and transmitted via the Internet (Khashan & Khafajah, 2023). Without IoT, it would be difficult or impossible to create smart and intelligent environments that AmI envisions.

However, replacing human activities is difficult to fully automated way. IoT is used to organise an environment that is intelligent and independent, such as intelligent cities, things, health, or even intelligent life (Abomhara & Køien, 2014), Figure 1 presents the general concept of the IoT, including its capabilities.

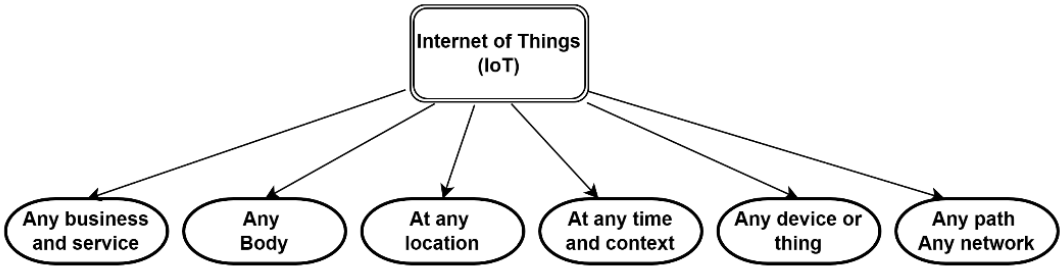

Figure 1. General concept of IoT modified Source: (Razaq M.A. et al. (2017))

Communication with IoT devices can be done via Wi-Fi, Bluetooth, or communication protocols. Hence, IoT messaging methods are widely used to enable message sharing among these devices. Consequently, several communication methods have been proposed and used frequently. However, employed methods at any layer of IoT architecture resulted in synchronisation and interoperability issues that need greater concern (Abdelouahid et. al., 2021). The main aim of this paper is to show the possibilities of message broadcast methods for IoT.

The paper is organised as follows. Section 2 presents related works regarding IoT messaging systems. Section 3 explains the background and methodology. Then, section 4 will discuss the result and analysis of the proposed work. Finally, section 5 will present the discussion and conclusion.

2. Related works

Tallberg (2020) defined message queuing systems are used to store data in IoT applications and allow devices to connect to the queueing system. These systems can improve stream processing using asynchronous processing, and decoupling of producers and consumers, another study by Nguyen et al. (2019) concluded that compatibilities, such as storage and synchronised processing, in IoT big data platforms and with other tools should be considered.

In another study by Fu et al. (2021) two modes, Push and Pull, were briefly explained. In Push mode, messages are pushed continuously to the consumer with better real-time performance, but it demands a flow control mechanism. In pull mode, message requests are based on a time interval but setting a reasonable pull interval is difficult. Continuous pull requests will impose a significant load on the queueing system. However, if the pull requests are not continuous and timely, they will cause latency problems. Thus, synchronisation will play a great role in solving such a problem.

IoT message queueing models are divided into the following categories based on their communication models: Request and response, Publisher subscriber, and polling (Domínguez-Bolaño et al., 2022). Choosing the right communication protocol that ensures security, reliability, and efficient data

transmission is a challenge in IoT applications (El Ouadghiri et al., 2020). Generally, Amjad et al. (2021) discussed the two basic categories of IoT communication protocols Device to Device(D2D); protocols such as Data Distribution Service (DDS) used for independent smart device communication, and Device to Server (D2S) protocols.

Ghotbou and Khansari (2021) stated that Constrained Application Protocol (CoAP) uses a request-response architecture to exchange messages in IoT. Still, Seoane et al. (2021) reported that CoAP has no mechanism for congestion control, confirming message delivery, which results in possible errors. In another study by Amjad et al. (2021) Message Queuing Telemetry Transport (MQTT) is a protocol that uses the publisher-subscriber model. It is currently used in several IoT applications. According to Mishra and Kertesz (2020), it is suitable for resource-constrained devices with reduced delay and low bandwidth. However, it doesn't support any encryption method.

Authors da Cruz et al. (2019) introduced a gateway for the application layer in HTTP, called MiddleBridge. The MiddleBridge uses packets smaller than those sent by an IoT device. However, simultaneous one-to-many communications is not supported (Šikić, L. et al., 2020). Authors Uy and Nam (2019) noted that the Advanced Message Queueing Protocol (AMQP) provides long-lasting queues and low-latency transmission. However, this model has limitations such as high bandwidth, unsuitable for real-time systems, and not supporting automatic resource discovery (Bang et al., 2022).

Authors Lohitha and Pounambal (2023) proposed a cloud-based IoT framework by integrating push-pull and publisher-subscriber methods for real-time IoT data sharing. They stated that streaming audio and video using this model does not support synchronous end-to-end communication. Further, Longo and Redondi (2023) state that the publisher-subscriber is a single broker model. Hence, a single point of failure and managing real-time data generated by IoT is a problem. They proposed a modified distributed broker resulting in reduced latency and network traffic. But they didn't consider the synchronisation problem.

Lu and Da Xu (2018) show capabilities in messaging within IoT in the context of smart house solutions. These solutions pose a challenge for the elderly in creating a safe and secure home environment to reduce stress, fear, falls, or social isolation. Anastasopoulos et. al. (2005) indicated that IoT devices are connected to the Internet within an ambient intelligent home environment using different communication protocols. Each communication protocol shall be suitable for every use, depending on several key factors.

IoT message queuing techniques need to be interoperable, power-efficient, fast, low latency, reliable, scalable, and robust in security. Further, based on the literature overview the response-request and publisher-subscriber schemes are mostly used. Thus, this work will propose message queueing models using Producer-Consumer and Reader-Writer.

3. Materials and Methods

3.1. IoT Messaging Techniques

IoT messaging techniques are the various methods by which IoT devices can communicate and exchange data with each other. This research reliability was ensured by using model triangulation. Some of the most common IoT communication models are displayed in Table 1.

Table 1. Internet of Things Requirements and Protocol

Protocol	*Transport*	*Messaging*	*2G,3G,4G (1000's)*	*LowPower and Lossy (1000's)*	*Compute Resources*	*Security*	*Success Stories*	*Arch*
CoAP	UDP	Rqst/Rspnse	Excellent	Excellent	10Ks/RAM Flash	Medium - Optional	Utility field area ntwks	Tree
Continua HDP	UDP	Pub/Subsrb Rqst/Rspnse	Fair	Fair	10Ks/RAM Flash	None	Medical	Star
DDS	UDP	Pub/Subsrb Rqst/Rspnse	Fair	Poor	100Ks/RAM Flash +++	High-Optional	Military	Bus
DPWS	TCP		Good	Fair	100Ks/RAM Flash ++	High-Optional	Web Servers	Client Server
HTTP/ REST	TCP	Rqst/Rspnse	Excellent	Fair	10Ks/RAM Flash	Low-Optional	Smart Energy Phase 2	Client Server
MQTT	TCP	Pub/Subsrb Rqst/Rspnse	Excellent	Good	10Ks/RAM Flash	Medium - Optional	IoT Msging	Tree
SNMP	UDP	Rqst/Response	Excellent	Fair	10Ks/RAM Flash	High-Optional	Network Monitoring	Client-Server
UPnP		Pub/Subscrb Rqst/Rspnse	Excellent	Good	10Ks/RAM Flash	None	Consumer	P2P Client Server
XMPP	TCP	Pub/Subsrb Rqst/Rspnse	Excellent	Fair	10Ks/RAM Flash	High-Manditory	Rmt Mgmt White Gds	Client Server
ZeroMQ	UDP	Pub/Subscrb Rqst/Rspnse	Fair	Fair	10Ks/RAM Flash	High-Optional	CERN	P2P

Source: (Kim and David (2016))

For further investigation, according to Table 1, we focused on message broadcast methods. Table 1 shows different communication protocols for message transmission, the Publish-Subscriber and/or Request-Response techniques are used preferably. The Producer-Consumer and Reader-Writer that originate from operating systems will be discussed later.

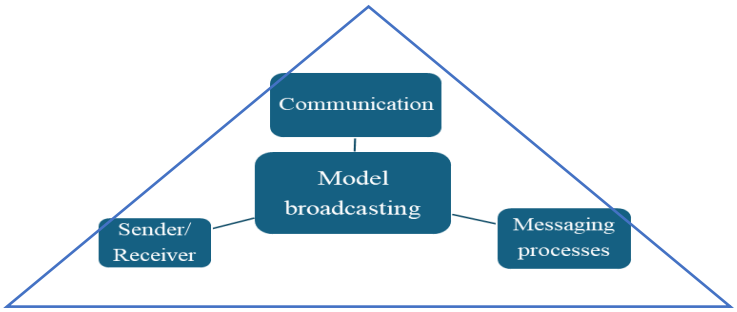

Figure 2. The triangulation model of broadcasting messages between IoTs

3.2. Publisher-Subscriber model

This messaging method involves one or more publishers, one or more subscribers, and a broker. In this technique, publishers produce messages and send them to a broker, which distributes the messages to one or more subscribers according to their interests. The Publisher-Producer model works using the following steps:

1. The publisher sends its message to the broker,
2. The broker receives the message and stores it,
3. The subscriber expresses interest by subscribing to specific topics or channels. A topic is a label or tag publishers use to categorise their messages,
4. The broker delivers messages to subscribers after it checks the subscribers' interests.

The subscriber processes the message and performs some other action based on the content of the message. This model is widely used in distributed messaging systems and event-driven architectures asynchronously. It decouples publishers and subscribers and allows them to operate independently and at different rates. The Publisher sends the messages to the broker, and then the broker forwards

them to everyone who requested them (subscribers). No messages are shared among devices directly without a broker (message distributer).

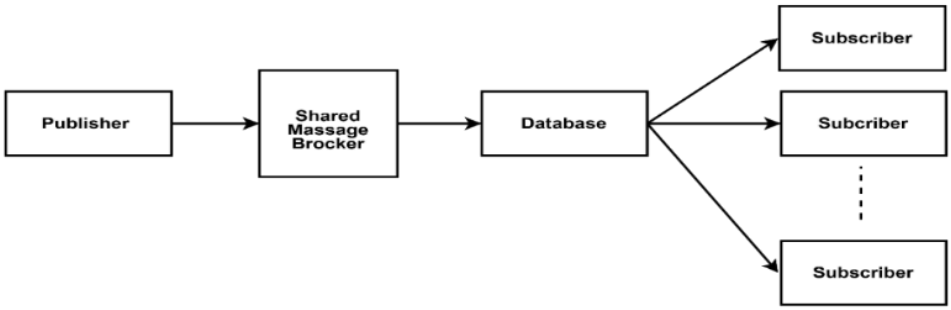

Figure 3. General communication model: Publisher-Subscriber

3.3. Request-Response Model

This model is a common way of communication between two entities. It consists of two parts, the client and the server. Messages are requested by the client and responses are generated and sent to the client by the server. The request-response messaging model works using the following steps:

1. The client sends the request to the system to trigger the communication. The request typically contains information about what the client wants the server to do or what data it needs,
2. The server processes the request and generates a response. The server may need to perform some computations, access a database or other resources, or perform some other operations to generate the response,
3. The server sends a response. The response typically contains the requested data or information about whether the requested operation was successful,
4. The client processes the response and the client may proceed with the next step.

In a request-response communication process, one sender and one receiver typically exist. This model is widely used in client-server systems, such as web applications. The request-response model allows efficient communication between the client and server, as each entity can focus on its specific roles and responsibilities.

The number of senders and receivers in a request-response communication process can be increased if multiple clients need to send requests to the same server. In this case, the server can handle multiple requests simultaneously using techniques such as multithreading or asynchronous programming.

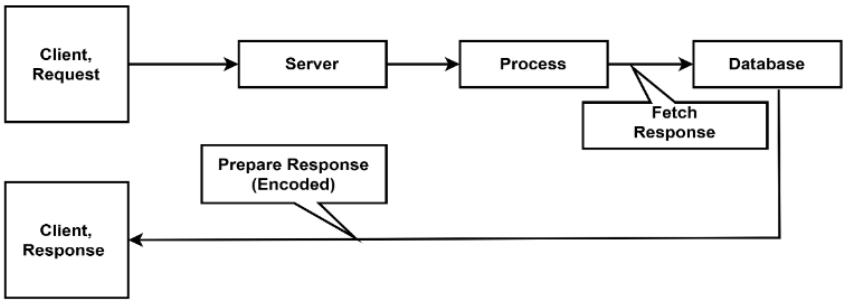

Figure 4. Request-response relationship of the general messaging model

As additional messaging techniques, methods originating from the operating system environment can be used. They are, for example, producers-consumers or writers-readers.

4. Result and Analysis

4.1 Producer-Consumer

The producer-consumer model is a common design pattern used in computer programming (operating systems). In this model, one or more "producer" devices generate data, and one or more "consumer" devices process data. The producers and consumers are decoupled so that each can operate independently and at its own pace. They are usually separate threads or processes, and communication between them is typically through the shared queue. The consumer retrieves the data or tasks from the queue and processes them.

The IoT producer-consumer model is a framework in which data is collected by IoT devices and then consumed by various applications. In this model, IoT devices act as data producers, while applications or systems act as consumers. The producer-consumer messaging model works using the following steps:

1. Date generation by IoT devices,
2. Data Transmission,
3. Data Storage,
4. Consumer applications,
5. Data Consumption.

The producer-consumer model in IoT environments can be implemented in various ways. One common approach is to use a message queue, where producers publish messages and consumers subscribe to those messages. This allows producers and consumers to operate asynchronously without requiring them to be connected at the same time.

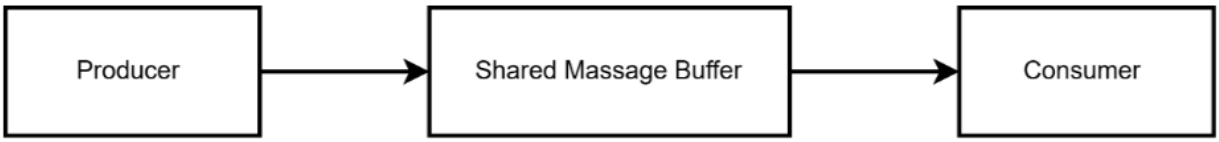

Figure 5. General Messaging Model of Producer-Consumer

4.2. Reader-Writer Model

The IoT reader-writer model is another method used in IoT environments to manage access to shared resources. Readers are entities that only read data and writers are entities that read and write data from the shared resource. The goal of this model is to ensure that multiple readers can access resources that are shared simultaneously. At a time only one writer can access the resource to prevent conflicts and ensure consistency of data.

The reader-writer messaging model works using the following steps:

1. The writer creates the message,
2. Multiple readers can access the message parallelly,
3. No readers can access the message when a writer is waiting for the resource,
4. No writer can access the resource when a reader is waiting for the message.

Two processes that share the same buffer, are not allowed to read the buffer at the same time. Such two processes are parallel mutual exclusion and sequential mutual exclusion. To implement the reader-writer model in an IoT environment, we need to synchronize the reading and writing process. This ensures that only one writer can publish messages on the topic at a time while allowing multiple readers to receive messages simultaneously. The IoT reader-writer model is a powerful tool for managing shared resources in IoT systems.

Table 2. Show how many senders and receivers one can use in each messaging method.

Basic messaging process	Senders	Receivers
Request-Response	One sender	One receiver
Publish-Subscribe	One or multiple publishers	Multiple Subscribers
Producer-Consumer	One or multiple producers	One or multiple consumers
Reader-Writer	One writer	Multiple readers

As indicated by Sethi and Sarangi (2017), the IoT is a three-layer architecture, namely the perception, network, and application layers. They are described as follows:

- The perception layer involves collecting data from various sensors, devices, and actuators. It includes devices embedded with sensors to sense, connect, and interact with the real world,
- The network layer is responsible for data messaging or transmission. If message transmission and receiving are not handled properly, a synchronisation problem occurs. This problem should be solved according to operating systems concepts semaphores, critical section, mutual exclusion, etc. (Stallings, 2021),
- The application layer defines various applications in which the IoT can be deployed. This layer is responsible for application-specific service delivery to the user.

5. Conclusion

Message queuing is a technique that plays an important role in data sharing among IoT devices in smart environments, where efficient data sharing is required. Due to the heterogeneous nature of IoT, message queuing techniques need to be interoperable, power-efficient, fast and low latency, reliable, scalable, and robust in security. The key findings of this study indicate the possibilities of IoT messaging techniques using operating system concepts Producer-consumer and reade-writer schemes. These messaging techniques allow flexibility in handling varying data generation and processing speeds, and asynchronous communication. This helps in managing resource constraints, when a fast producer generates data faster than a slow consumer can process, or vice versa. Moreover decoupling between the producer and consumer enables modularity and enabling scalable and loosely coupled architectures.

It's important to note that these messaging techniques mentioned in this article, can be used in combination or tailored to the specific requirements of an IoT system, considering factors such as device capabilities, network constraints, security considerations, and scalability needs. These messaging techniques facilitate efficient communication and data exchange between IoT devices, applications, and services, enabling seamless integration and interoperability in IoT ecosystems.

Acknowledgement

This paper was supported by grant No. SGS_2024_017, Faculty of Economics and Administration, University of Pardubice, Czech Republic.

References

Abdelouahid, R. A., Debauche, O., & Marzak, A. (2021). Internet of Things: a new interoperable IoT platform. Application to a smart building. Procedia Computer Science, 191, 511-517.

Abomhara, M., & Køien, G. M. (2014, May). Security and privacy in the Internet of Things: Current status and open issues. In 2014 international conference on privacy and security in mobile systems (PRISMS) (pp. 1-8). IEEE.

Amjad, A., Azam, F., Anwar, M. W., & Butt, W. H. (2021). A systematic review on the data interoperability of application layer protocols in industrial IoT. IEEE Access, 9, 96528-96545.

Anastasopoulos, M., Niebuhr, D., Bartelt, C., Koch, J., & Rausch, A. (2005, October). Towards a reference middleware architecture for ambient intelligence systems. In ACM conference on object-oriented programming, systems, languages, and applications.

Bang, A. O., Rao, U. P., Visconti, A., Brighente, A., & Conti, M. (2022). An IoT inventory before deployment: a survey on iot protocols, communication technologies, vulnerabilities, attacks, and future research directions. Computers & Security, 123, 102914.

da Cruz, M. A., Rodrigues, J. J., Lorenz, P., Solic, P., Al-Muhtadi, J., & Albuquerque, V. H. C. (2019). A proposal for bridging application layer protocols to HTTP on IoT solutions. Future Generation Computer Systems, 97, 145-152.

Domínguez-Bolaño, T., Campos, O., Barral, V., Escudero, C. J., & García-Naya, J. A. (2022). An overview of IoT architectures, technologies, and existing open-source projects. Internet of Things, 20, 100626.

El Ouadghiri, M., Aghoutane, B., & El Farissi, N. (2020). Communication model in the Internet of Things. Procedia Computer Science, 177, 72-77.

Fu, G., Zhang, Y., & Yu, G. (2020). A fair comparison of message queuing systems. IEEE Access, 9, 421-432.

Ghotbou, A., & Khansari, M. (2021). VE-CoAP: A constrained application layer protocol for IoT video transmission. Journal of Network and Computer Applications, 173, 102855.

Kim, R., & David, L. (2016),” Internet of Things Requirements and Protocol”, IEEE Standards University E-magazine.

Khashan, O. A., & Khafajah, N. M. (2023). Efficient hybrid centralized and blockchain-based authentication architecture for heterogeneous IoT. Journal of King Saud University-Computer and Information Sciences, 35(2), 726-739.

Kissoum, Y., Kerraoui, S., & Boughaouas, M. L. (2014). Smart Home for Elderly: Modeling and Simulation. In ICAASE (pp. 148-155).

Lohitha, N. S., & Pounambal, M. (2023). Integrated publish/subscribe and push-pull method for cloud based IoT framework for real-time data processing. Measurement: Sensors, 27, 100699.

Longo, E., & Redondi, A. E. (2023). Design and implementation of an advanced MQTT broker for distributed pub/sub scenarios. Computer Networks, 224, 109601.

Lu, Y., & Da Xu, L. (2018). Internet of Things (IoT) cybersecurity research: A review of current research topics. IEEE Internet of Things Journal, 6(2), 2103-2115.

Mishra, B., & Kertesz, A. (2020). The use of MQTT in M2M and IoT systems: A survey. IEEE Access, 8, 201071-201086.

Nguyen, C. N., Lee, J., Hwang, S., & Kim, J. S. (2019). On the role of message broker middleware for many-task computing on a big-data platform. Cluster Computing, 22, 2527-2540.

Razzaq, M. A., Gill, S. H., Qureshi, M. A., & Ullah, S. (2017). Security issues in the Internet of Things (IoT): A comprehensive study. International Journal of Advanced Computer Science and Applications, 8(6), 383.

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.

Sethi, P., & Sarangi, S. R. (2017). Internet of Things: Architectures, protocols, and applications. Journal of Electrical & Computer Engineering.

Šikić, L., Janković, J., Afrić, P., Šilić, M., Ilić, Ž., Pandžić, H., ... & Džanko, M. (2020, September). A comparison of application layer communication protocols in IoT-enabled smart grid. In 2020 International symposium ELMAR (pp. 83-86). IEEE.

Stallings, W. (2021). Operating systems: internals and design principles. Pearson.

Tallberg, S. (2020). A Comparison of Data Ingestion Platforms in Real-Time Stream Processing Pipelines.

Uy, N. Q., & Nam, V. H. (2019, December). A comparison of AMQP and MQTT protocols for Internet of Things. In 2019 6th NAFOSTED Conference on Information and Computer Science (NICS) (pp. 292-297). IEEE.