PEOPLE'S DEMOCRATIC REPUBLIC OF ALGERIA

Ministry of Higher Education and Scientific Research

Mohamed Khider University - BISKRA

Faculty of Exact Sciences, Natural Sciences and Life

Computer Science department

Order N°: :1/M2/2021

Memory Thesis

presented to obtain the academic master's degree in

Computer Science

Option: Networks and Technologies of Information and Telecommunications

publish/subscribe communications in the contextof IoT-connected e-healthcare: A prototype

By:

BOUDERHEM MOHAMED

Soutenu Defended on September 27, 2020, in front of the jury composed of:

Aloui Imen	MCB	President/chairwoman
Sahraoui Somia	MCB	Rapporteur
Belaiche Hamza	MCA	Examiner

year university 2020-2021

knowledgements

My thanks go first to the almighty ALLAH, for providing me with courage and patience for guiding my steps towards the path of knowledge.

I would like to extend my sincere thanks to my supervisor Dr. SAHRAOUI Somia of the Computer Science faculty at Mohamed Khider. The door to Prof. SAHRAOUI office was always open whenever I ran into a trouble spot or had a question about my research or writing. He consistently allowed this paper to be my own work, but steered me in the right direction whenever he thought I needed it.

I would also like to thank my friends for their support. Last but not the least, I would like to thank my family: my parents and to my brothers and sisters and my cousin BOUDERHEM RABIE like a brother to me, like an older brother for supporting me spiritually throughout writing this thesis and my life in general. Last but not the least, I would also like to thank my friends for their support.

Abstract

With the increased technological development, the use of the Internet of Things in various fields has become a necessity. Where Internet of Things plays the role of technical tool by empowering physical resources into smart entities through mixture of various communications and embedded technologies in its architecture. MQTT is a machine-to-machine internet of things connectivity protocol. It is an extremely lightweight and publish-subscribe messaging transport protocol. This protocol is useful for the connection with the remote location where the bandwidth is a premium.

Our work is about an e-healthcare prototype that carries out communications between publisher (patient) and subscriber (doctor) using MQTT. The solution is lightweight and efficient and was tested by raspberry Pi and sensor medical and laptop and smart phone.

الملخص

ومع زيادة التطور التكنولوجي ، أصبح استخدام شبكة إنترنت الأشياء في مختلف الميادين أمرا ضروريا. حيث تقوم شبكة إنترنت الأشياء بدور الأداة التقنية من خلال تمكين الموارد المادية إلى كيانات ذكية من خلال خليط من مختلف الاتصالات والتكنولوجيات المدمجة في بنيتها.MQTT هو بروتوكول ربط بين الآلات .. وهو بروتوكول نقل خفيف الوزن للغاية ومشترك في النشر. هذا البروتوكول مفيد للاتصال مع الموقع البعيد حيث عرض النطاق الترددي هو قسط.

ويتعلق عملنا بنموذج الرعاية الصحية الإلكترونية الذي ينفذ الاتصالات بين الناشر (المريض) والمشترك (الطبيب) باستخدام MQTT. والحل خفيف الوزن وفعال وتم اختباره بواسطة جهاز "راسبري باي" وجهاز استشعار طبي وحاسوب محمول وهاتف ذكي.

Table of Contents

List of figures

General Introduction

The first principle of IoT (Internet of Things) is to connect smart objects - things- to the Internet in a transparent way. This leads to an exchange of data between all things, and bring users information in a more secure way. Cisco Systems estimates that IoT will consist of 50 billion devices connected to the Internet by 2020 and it is predictable that many physical objects, like computers, sensor actuators, will be distributed with unique addresses and the ability to transfer data, from the common daily activities to restricted medical records, in a secure way.

Among various messaging protocols for IoT, Message Queuing Telemetry Transport (MQTT) protocol plays a vital role in M2M communications. MQTT is an application layer protocol which is used to transmit messages by connecting devices through constrained approach. It is lightweight protocol when compared to all other protocols. It is based on publish and subscribes messaging pattern through MQTT broker. The main advantage of MQTT is that it will buffer the unsent messages during broker disconnection and will send to the subscriber when it is connected again.

E-Health is one of the most important application areas of IoT. It provides opportunities to several medical applications such as mobile and remote health monitoring. The integration of wearable devices and systems in IoT provides better m-health services. Medical devices generate data and send to designated computer servers.

This work, which aims to Embellishment a prototype e-health system communication, is organized into three chapters:

1. The 1 st chapter introduces generalities on internet of things, including their characteristics and their context.
2. The 2 nd chapter studies firstly, the publish/subscribe communication and more specifically MQTT protocol. Then, we devote to the topic e-healthcare.
3. The 3rd chapter we explain the realized prototype step by step.

chapterOne. **Generalities**

Interne of Things

1.1 Introduction

The Internet of Things (IoT) is the network of physical objects-devices, instruments, vehicles, buildings and other items embedded with electronics, circuits ,software, sensors and network connectivity that enables these objects to collect and exchange data The Internet of Things allows objects to be sensed and controlled remotely across existing network infrastructure, creating opportunities for more direct integration of the physical world into computer-based systems, and resulting in improved efficiency and accuracy The concept of a network of smart devices was discussed as early as 1982, with a modified Coke machine at Carnegie Mellon University becoming the first internet-connected appliance , able to report its inventory and whether newly loaded drinks were cold Kevin Ashton (born 1968) is aBritish technology pioneer who is known for inventing the term "the Internet of Things" to describe a system where the Internet is connected to the physical world via ubiquitous sensors IoT is able to interact without human intervention Some preliminary IoT applications have been already developed in healthcare, transportation, and automotive industries IoT technologies are at their infant stages; however, many new developments have occurred in the integration of objects with sensors in the Internet The development of IoT involves many issues such as infrastructure, communications, interfaces, protocols, and standards The objective of this paper is to give general concept of IoT, the architecture and layers inIoT, some basic terms associated with it and the services provided Introduction to IOT Pradyumna Gokhale1 , Omkar Bhat2 , Sagar Bhat3 IT Dept , Smt Kashibai Navale College of Engineering.[1]

1.2 IoT History

Ever since the birth of the internet in 1989, connecting “Things” in the internet began widely Trojan Room coffee pot is possibly the first application of this kind In 1990 John Romkey created the first Internet ‘device’, a toaster that could be turned on and off over the Internet WearCam was invented in 1994 by Steve Mann It had a near-real-time performance using a 64-processor system Paul Saffo’s gave the first brief description about sensors and their future course of action in 1997 In 1999 The Internet of Things term was coined by Kevin Ashton, executive director of the AutoIDCentre, MIT They also invented a global RFID-based item identification system in the same year As a major leap in commercializing, in 2000 electronics giant LG announced its plans of revealing a smart refrigerator that would determine itself whether or not the food items stored in it are replenished In 2003 RFID was deployed at a massive level in US army in their Save program The same year saw retail giant Walmart to deploy RFID in all its shops across the globe to a greater extent In 2005 main stream publications like The Guardian, Scientific American and Boston Globe cited many articles about IoT and its future course In 2008 a group of companies launched the IPSO Alliance to promote the use of Internet Protocol (IP) in networks of "smart objects" and to enable the Internet of Things In 2008 the FCC approved the usage of the “white space spectrum” Finally the launch of IPv6 in 2011 triggered massive growth and interests in this field Later IT giants like Cisco, IBM, Ericson took a lot of educational and commercial initiatives with IoT.[2]

1.3 IoT Definition

The Internet of Things (IoT) is the network of physical objects accessed through the internet, as defined by technology analysts and visionaries These objects contain embedded technologies such as wireless sensor networks (WSN) and Radio frequency identification (RFID) to interact with internal state or external environment.

In general, we can say IoT allows people and things to be connected Anytime, Anyplace, with anything and anyone using any network and any service as shown in (Figure 1.1). [3]

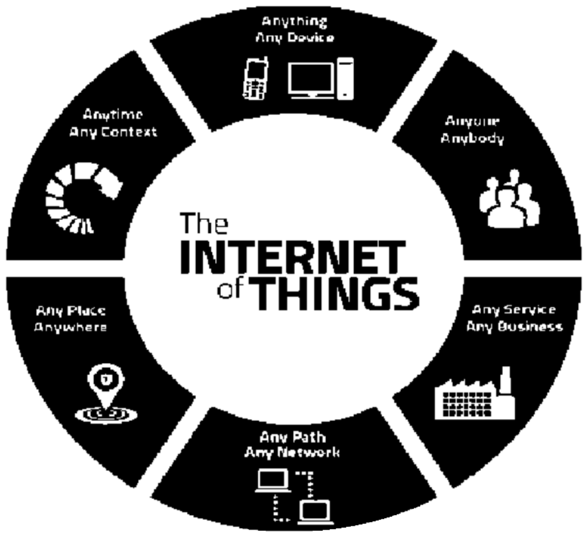


Figure 1.1 internet of things

1.4 Architecture

Today's Internet is using TCP/IP protocol stack for communication between network hosts which was proposed long time ago. However, the IoT connects billions of objects which will create much larger traffic and much more data storage is needed. Also, IoT will face many challenges specially related to privacy and security. Thus, the new proposed architecture for IoT needs to address many factors like scalability, interoperability, reliability, QoS, etc. Since IoT connects everything and everyone to exchange information among themselves, the traffic and storages in the network will also increase in the exponential way. Thus, IoT development depends on the technology progress and design of various new applications and business models. The basic architecture of IoT is proposed.

Generally, the structure of IoT is divided into five layers as shown in Figure 1.2. These layersare briefly described below: [4]

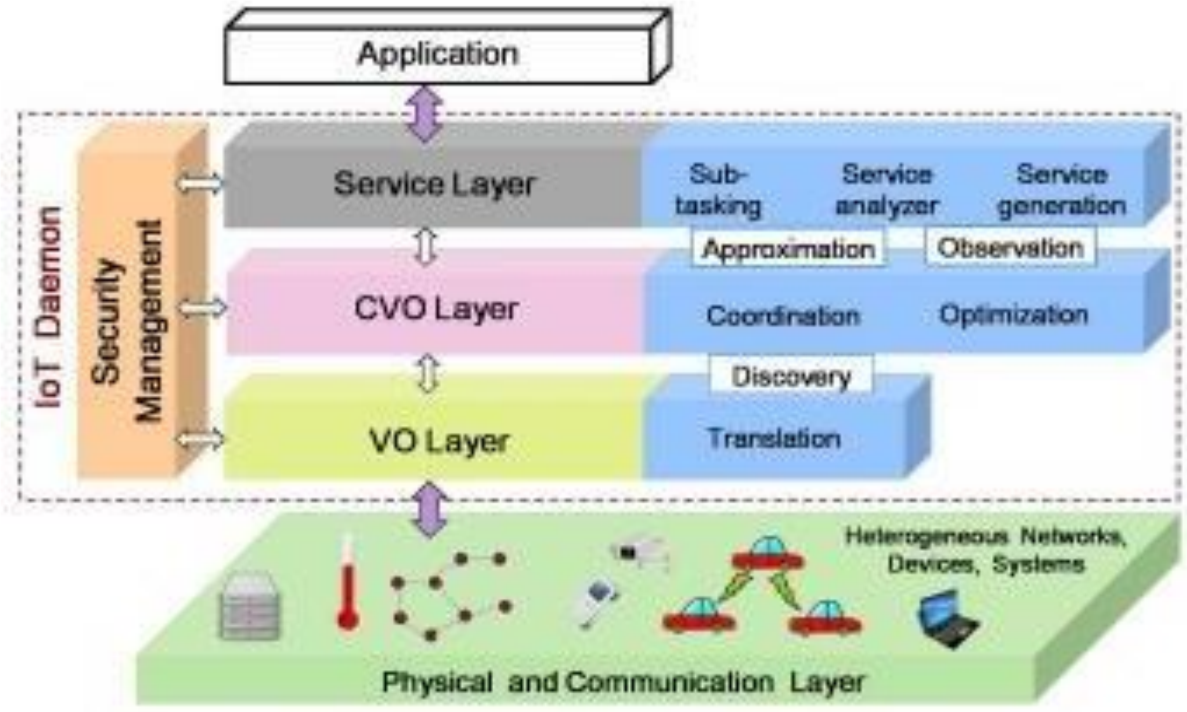


Figure 1.2 A-layered-architecture-for-the-Internet-of-Things

1.4.1 Perception Layer

The Perception layer is also known as 'Device Layer'. It consists of the physical objects and sensor devices. The sensors can be RFID, 2D-barcode, or Infrared sensor depending upon objects identification method. This layer basically deals with the identification and collection of objects specific information by the sensor devices. Depending on the type of sensors, the information can be about location, temperature, orientation, motion, vibration, acceleration, humidity, chemical changes in the air etc. The collected information is then passed to Network layer for its secure transmission to the information processing system.

1.4.2 Network Layer

The Network layer can also be called 'Transmission Layer'. This layer securely transfers theinformation from sensor devices to the information processing system. The transmission medium can be wired or wireless and technology can be 3G, UMTS, WIFI, Bluetooth, infrared, ZigBee, etc. depending upon the sensor devices. Thus, the Network layer transfersthe information from Perception layer to Middleware layer.

1.4.3 Middleware Layer

The devices over the IoT implement different type of services. Each device connects and communicates with only those other devices which implement the same service type. This layer is responsible for the service management and has link to the database. It receives theinformation from Network layer and store in the database. It performs information processing and ubiquitous computation and takes automatic decision based on the results.

1.4.4 Application Layer

This layer provides global management of the application based on the object's informationprocessed in the Middleware layer. The applications implemented by IoT can be smart health, smart farming, smart home, smart city, intelligent transportation, etc.

1.4.5 Business Layer

This layer is responsible for the management of overall IoT system including the applications and services. It builds business models, graphs, flowcharts etc. based on the data received from Application layer. The real success of the IoT technology also depends on the good business models. Based on the analysis of results, this layer will help to determine the future actions and business strategies.

1.5Enabling technologies

IoT can be realized with several enabling technologies: [15]

1.5.1RFID

Generally speaking, RFID, as a non-contact communication technology, is used to identify and track objects without contact. It supports data exchange via radiosignals over a short distance. The RFID-based system consists of RFID tag, RFID reader, and antenna. RFID tag can be a microchip attached to an antenna. Each RFID tag is attached in an object and has its unique identification number. A RFIDreader can identify an object and obtain the corresponding information by querying to the attached RFID tag through appropriate signals. An antenna is used to transmit signals between RFID tag and RFID reader. In comparison with other technologies, RFID has the following benefits (fast scanning, durability, reusability, large storage, noncontact reading, security, small size, low cost, etc.).Because of these benefits, RFID can be useful in the perception layer of IoT to identify and track objects and exchange information.

1.5.2Wireless Sensor Networks (WSN)

WSN can play a very important role in IoT. WSN can monitor and track the status of devices, and transmit the status data to the control center or sink nodes via multiple hops. Thus, WSN can be considered as the further bridge between the real world and the cyber world. In comparison with other technologies, WSN has a number of benefits, including scalability, dynamic reconfiguration, reliability, small size, low cost, and low energy consumption. Allthese benefits help WSN to be integrated in various areas with diverse requirements.

Notice that both RFID and WSN can be used for data acquisition in IoT, and the difference is that RFID is mainly used for object identification, while WSN is mainly used for the perception of real-world physical parameters associated withthe surrounding environment.

In addition, RFID sensor network (RSN) is an integration of RFID system and sensor network. In an RSN, sensor network can cooperate with RFID system to identify and track the status of objects. In an RSN, small RFID-based sensing devices and RFID reader are implemented, where the RFID reader works as a sinknode to generate data and provides power for network operations.

1.5.3 Barcode

Barcode, also denoted one-dimensional code, stores the information in several black lines and white spacings. These lines and spacings have different widths, organized in a linear or one-dimensional direction, and are arranged with special encoding rules. The information included in the barcode can be read by a machine that scans the barcode with an infrared beam. A two-dimensional code record the information by using black and white pixels laid out on the plane, in which black pixel represents a binary of '1' and white pixel represents a binary of'0'. With special encoding rules, the black and white pixels can store a significantamount of information. In comparison with barcode, two-dimensional code has the benefit of high information content, high reliability, high robustness, etc.

1.6Protocol

1.6.1type of protocol IoT

This section presents the four widely accepted and emerging messaging protocols for IoT systems: MQTT, CoAP, AMQP and HTTP, which are shown at the top of the protocolstack for IoT systems in Fig1.3. [5]

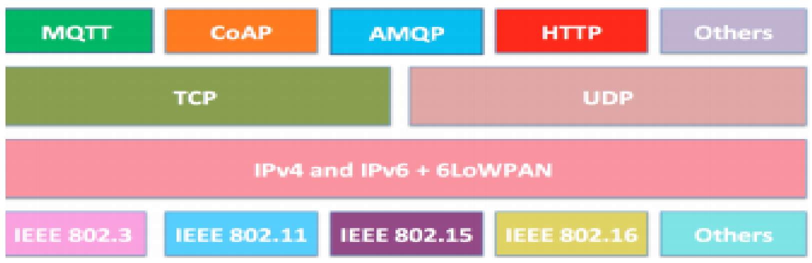


Figure 1.3 Protocol Stack for IoT Systems

1.6.1.1MQTT (Message Queuing Telemetry Transport Protocol)

MQTT is one of the oldest M2M communication protocols, which was introduced in 1999. It was developed by Andy Stanford-Clark of IBM and Arlen Nipper of Arcom ControlSystems Ltd (Eurotech). It is a publish/subscribe messaging protocol designed for lightweight M2M communications in constrained networks. MQTT client publishes messages to an MQTT broker, which are subscribed by other clients or may be retained for the future subscription. Every message is published to an address, known as a topic. Clients can subscribe to multiple topics and receives every message published to each topic. MQTT is a binary protocol and normally requires fixed header of 2-bytes with small message payloads up to maximum size of 256 MB. It uses TCP as a transport protocol and TLS/SSL for security. Thus, communication between client and broker is a connection oriented. Another great feature of MQTT is its three levels of Quality of Service (QoS) for reliable delivery of messages. MQTT is most suitable for large networks of small devices that need to be monitored or controlled from a back-end server on the Internet. It is neither designed for device-to-device transfer nor for multicast data to many receivers. It is a very basic messaging protocol offering only a few control options.

1.6.1.2CoAP (Constrained Application Protocol)

CoAP is a lightweight M2M protocol from the IETF Core (Constrained RESTful Environments) Working Group. CoAP supports both request/response and resource/observe (a variant of publish/subscribe) architecture. CoAP is mainly developed to interoperate with HTTP and the RESTful Web through simple proxies.

Publisher publishes data to the URI and subscriber subscribes to a particular resource Unlike MQTT, CoAP uses Universal Resource Identifier (URI) instead of topics [9]. Publisher publishes data to the URI and subscriber subscribes to a particular resource indicated by the URI. When a publisher publishes new data tothe URI, then all the subscribers are notified about the new value as indicated by the URI. CoAP is a binary protocol and normally requires fixed header of 4-bytes with small message payloads up to maximum size dependent on the web server or the programming technology. CoAP uses UDP as a transport protocol and DTLS for security. Thus, clients and servers communicate through connectionless datagrams with less reliability. However, it uses “confirmable” or“non-confirmable” messages to provide two different levels of QoS. Where, confirmable messages must be acknowledged by the receiver with an ACK packet and nonconformable messages are not. CoAP offers more functionality than MQTT such as it supports content negotiation to express a preferred representation of a resource; this allows client and server to evolve independently, adding new representations without affecting each other.

1.6.1.3AMQP (Advanced Message Queuing Protocol)

AMQP is a lightweight M2M protocol, which was developed by John O’Hara at JPMorgan Chase in London, UK in 2003. It is a corporate messaging protocol designed for reliability, security, provisioning and interoperability. AMQP supports both request/response and publish/subscribe architecture. It offers a wide range of features related to messaging such as a reliable queuing, topic-based published-subscribe messaging, flexible routing and transactions. AMQP communication system requires thateither the publisher or consumer creates an “exchange” with a given name and then broadcasts that name. Publishers and consumers use the name of this exchange to discover each other. Subsequently, a consumer creates a “queue” and attaches it to the exchange at the same time. Messages received by the exchange have to be matched to the queue via a process called “binding”. AMQP exchanges messages in various ways: directly, in fanout form, by topic, or based on headers. AMQP is a binary protocol and normally requires fixed header of 8-bytes with small message payloads up to maximum size dependent on the broker/server or the programming technology. AMQP uses TCP as a default transport protocol and TLS/SSL and SASL for security. Thus, the communicationbetween client and broker is a connection-oriented. Reliability is one of the core features of AMQP, and it offers two preliminary levels of Quality of Service (QoS) for delivery of messages: Unsettle Format (not reliable) and Settle Format (reliable).

1.6.1.4HTTP (Hyper Text Transport Protocol)

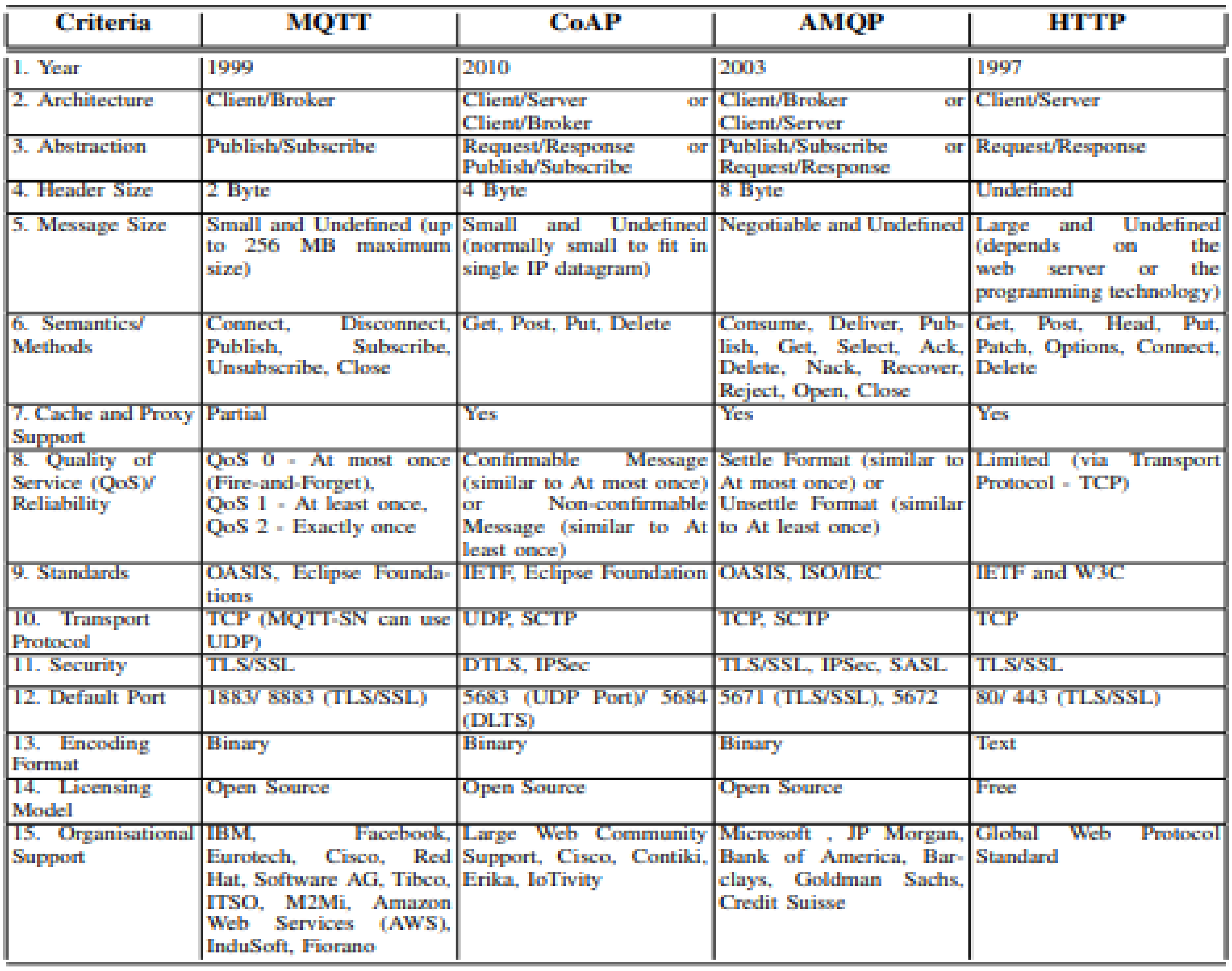
HTTP is predominantly a web messaging protocol, which was originally developed by Tim Berners-Lee. Later, it was developed by IETF and W3C jointly and first published as a standard protocol in 1997. HTTP supports request/response RESTful Web architecture. Analogous to CoAP, HTTP uses Universal Resource Identifier (URI) instead of topics.

Server sends data through the URI and client receives data through particular URI. HTTP isa text-based protocol and it does not define the size of header and message payloads rather it depends on the web server or the programming technology. HTTP uses TCP as a default transport protocol and TLS/SSL for security. Thus, communication between client and server is a connection-oriented. It does not explicitly define QoS and requires additional support for it. HTTP is a globally accepted web messaging standard offers several features such as persistent connections, request pipelining, and chunked transfer encoding.

1.6.2COMPARATIVE ANALYSIS OF MESSAGING PROTOCOLSFOR IOT SYSTEMS: HTTP, COAP, AMQP AND MQTT

This section presents a comparative analysis of the four widely accepted and emerging messaging protocols for IoT systems MQTT, CoAP, AMQP and HTTP based on several criteria to introduce their characteristics comparatively. This complete comparative studyis shown in Table.

Table 1 Comparative Analysis of Messaging Protocols for IoT Systems: MQTT,CoAP, AMQP and HTTP



Criteria	MQTT	CoAP	AMQP	HTTP
1. Year	1999	2010	2003	1997
2. Architecture	Client/Broker	Client/Server or Client/Broker	Client/Broker or Client/Server	Client/Server
3. Abstraction	Publish/Subscribe	Request/Response or Publish/Subscribe	Publish/Subscribe or Request/Response	Request/Response
4. Header Size	2 Byte	4 Byte	8 Byte	Undefined
5. Message Size	Small and Undefined (up to 256 MB maximum size)	Small and Undefined (normally small to fit in single IP datagram)	Negotiable and Undefined	Large and Undefined (depends on the web server or the programming technology)
6. Semantics/ Methods	Connect, Disconnect, Publish, Subscribe, Unsubscribe, Close	Get, Post, Put, Delete	Consume, Deliver, Publish, Get, Select, Ack, Delete, Nack, Recover, Reject, Open, Close	Get, Post, Head, Put, Patch, Options, Connect, Delete
7. Cache and Proxy Support	Partial	Yes	Yes	Yes
8. Quality of Service (QoS)/ Reliability	QoS 0 - At most once (Fire-and-Forget), QoS 1 - At least once, QoS 2 - Exactly once	Confirmable Message (similar to At most once) or Non-confirmable Message (similar to At least once)	Settle Format (similar to At most once) or Unsettle Format (similar to At least once)	Limited (via Transport Protocol - TCP)
9. Standards	OASIS, Eclipse Foundations	IETF, Eclipse Foundation	OASIS, ISO/IEC	IETF and W3C
10. Transport Protocol	TCP (MQTT-SN can use UDP)	UDP, SCTP	TCP, SCTP	TCP
11. Security	TLS/SSL	DTLS, IPSec	TLS/SSL, IPSec, SASL	TLS/SSL
12. Default Port	1883/ 8883 (TLS/SSL)	5683 (UDP Port)/ 5684 (DLTS)	5671 (TLS/SSL), 5672	80/ 443 (TLS/SSL)
13. Encoding Format	Binary	Binary	Binary	Text
14. Licensing Model	Open Source	Open Source	Open Source	Free
15. Organisational Support	IBM, Facebook, Eurotech, Cisco, Red Hat, Software AG, Tibco, ITSO, M2Mi, Amazon Web Services (AWS), InduSoft, Fiorano	Large Web Community Support, Cisco, Contiki, Erika, IoTivity	Microsoft , JP Morgan, Bank of America, Barclays, Goldman Sachs, Credit Suisse	Global Web Protocol Standard

1.7 IoT Application Domains

Presently, IoT might not have widespread visible effects on the society, but its impact is already noticeable in many industrial sectors. Considering a few among many of the IoT application domains that have emerged in recent years. It focuses mainly on some of the domains that have great potential for exponential growth in the fourth industrial revolution, namely home automation, energy, developed urban areas, transportation, healthcare, manufacturing, supply chain, wearables, and agriculture, as depicted in Figure1.4.[3]

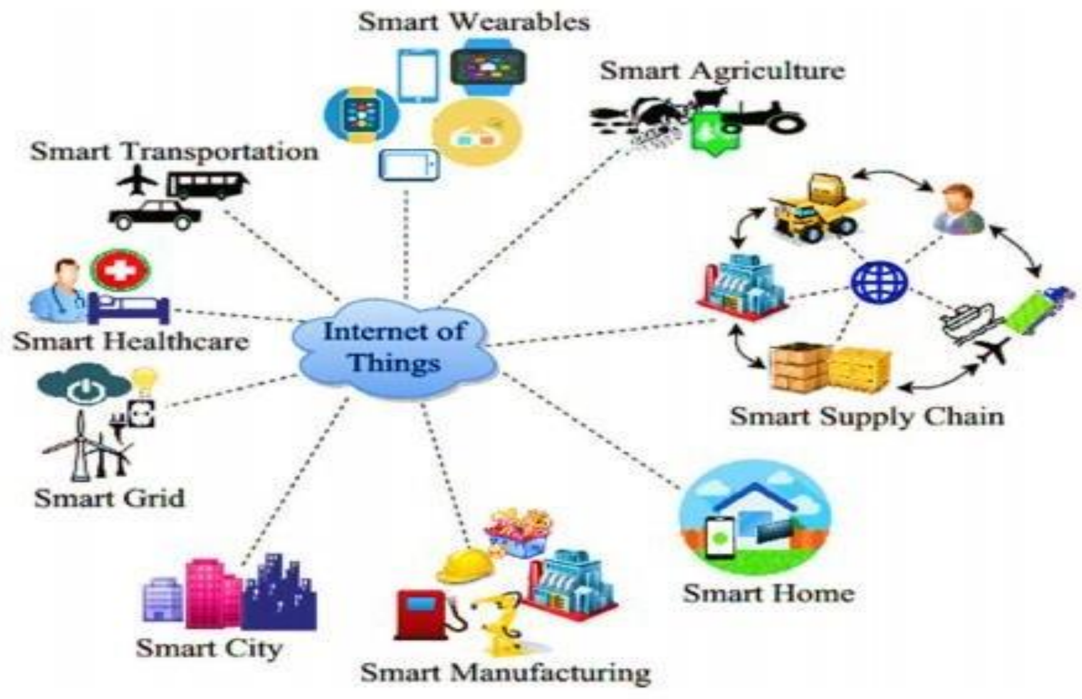


Figure 1.4 IoT application domains

1.7.1 Smart home system model

Smart home is a house or living environment where household appliances like toasters, washing machines and other everyday devices can be remotely monitored and controlled using smartphones, tablets, or laptop computers from anywhere in the World via the Internet or private network. This might enhance home care and monitoring, access control, energy efficiency, convenience and quality of everyday life.Smart TVs, security cameras, refrigerators, door locks, garage door openers, and smart hubs are typical cyber assets in the smart home domain.

The system model in Figure 1.5 comprised of a controller, which can be any IoT home automation hub, a wireless router, and 8 Wi-Fi enabled smart home appliances that can interface with a home automation software platform such as OpenHAB and Home Assistant. The software platform allows a user to wirelessly control devices from a smartphone or any computer on the home network. The controller is connected to the home network via the router Ethernet interface. A Raspberry Pi Single-Board Computer (SBC) can also be configured as the controller. The messaging protocol often employed for communication between the home automation software platform server and the smart devices is the MQTT. A typical lightweight server that implements the MQTT protocol is the Eclipse mosquito.

Figure 1.5 smart home

1.7.2 Smart transportation system model

Smart transportation provides a safer, cleaner and more efficient transport system by using real-time traffic information and interconnecting vehicles and roadside infrastructures for more efficient data acquisition. Additionally, it will improve quality of life by decreasing congestion, and hence shortening travel time, as well as reducing fuel/electricity usage.

The system model (Figure 1.6) shows a robust transportation system that exploits seamless information coordination and exchange across the different components of the network to provide an efficient and safe transportation. Thesystem components include smart traffic lights, cameras, radar, GPS, smart vehicle, smart trains, and smart road infrastructure. The system utilizes real-time GPS location data, traffic flow prediction mechanism, location-based services, smart traffic light scheduling management, and enormous amount of sensor datacollected from different areas in the network for making final decisions.

Figure 1.6 Smart transportation system model

1.7.3Smart supply chain system model

Smart supply chain Refers to a proactive and customer-centric monitoring, tracking, and remote asset management system that integrates different technologies (IoT, WSNs, RFID, big data, cloud computing...) to provide information on location, status, environment, and functionality of products andservices. to reduce operational costs, and allows for timely responses to unexpected events.

Figure 1.7 represents a system model for smart supply chain based on Blockchain comprising production and distribution systems of geographically dispersed enterprises that collaborate in a secure manner to jointly and efficientlyproduce and deliver end products to customers. In the context of supply chain, IoT security issues will typically revolve around authentication, connection and transaction. However, using the distributed ledger in Blockchain, the enterprises, namely, supplier of raw materials, factory, and distributor can conduct business transactions in a trusted and secure environment. The immutable record of Blockchain enables reliable creation of networks histories, allowing for tracking the actions of network devices.

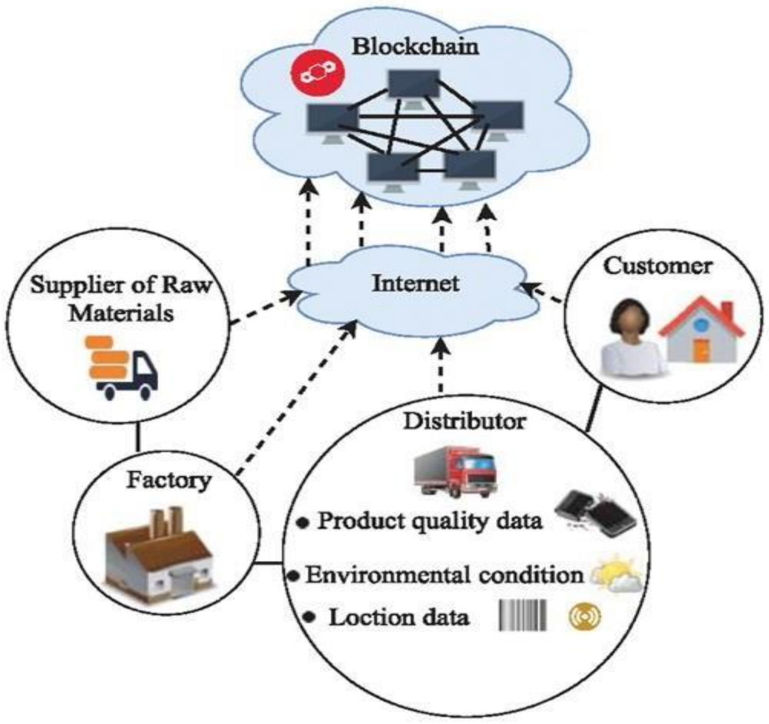


Figure 1.7 Smart supply chain system mode

1.7.4Smart agriculture system model

Smart agriculture is a sustainable farming practice that employs IT and other relevant technologies to increase the per unit yield of farming land by optimizingwater use and preserving other natural resources in order to increase crop yieldsand financial returns.

The Figure 1.8 shows a diagram representing the proposed smart agriculture system model. Plants normally require specific environmental conditions for optimal growth, health and overall crop yield. Thus, the system model consists of different sensor networks for measuring soil moisture, temperature, sun light, humidity, as well as sensor networks for monitoring the location of farm animals, and for prompt disease detection. Onboard the UAV are a SBC, a flight control system (autopilot) like Pixhawk, sensor and camera. The sensor is used to collect real-time data from the flight control system every second and sent to a server via4G LTE dongle attached to the SBC.

The autopilot can also receive commands through the server, hence by connecting to the server via the Internet, the farmer can control the drone by viewing the real-time data and issuing control commands.

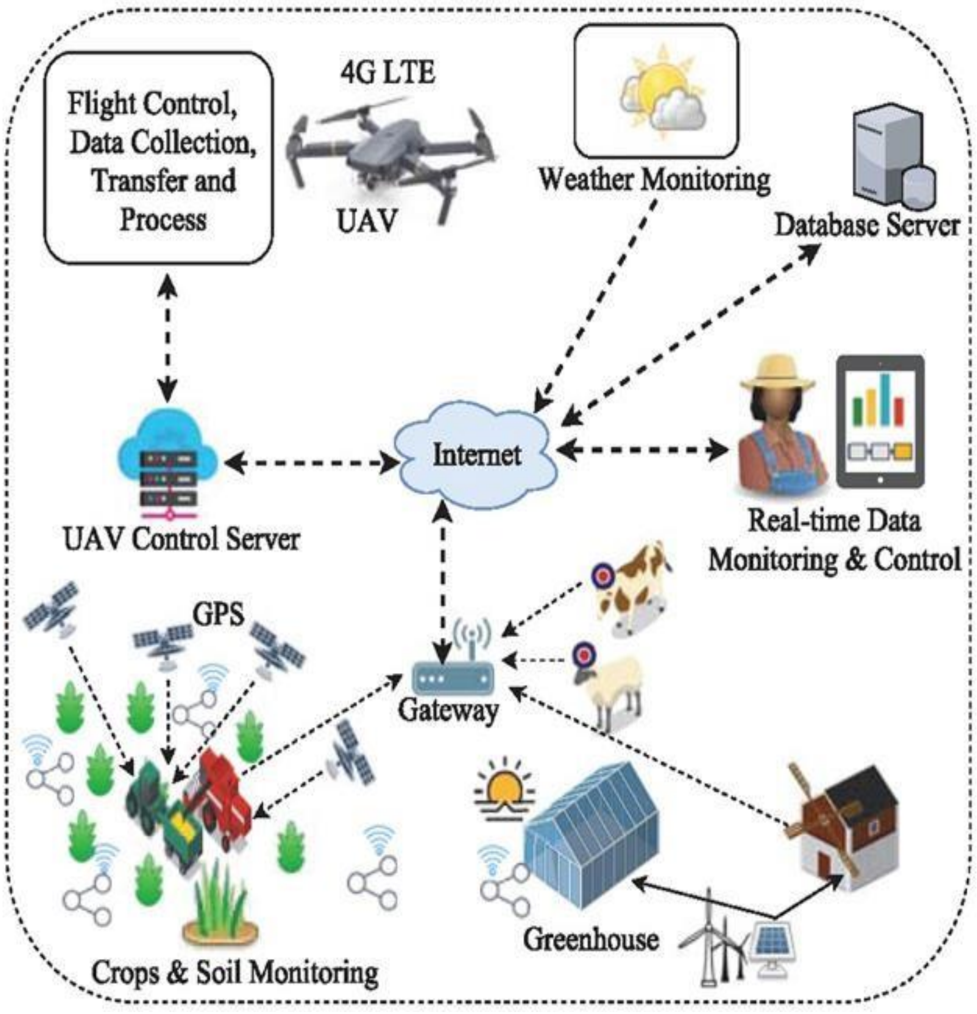


Figure 1.8 Smart agriculture system model

1.7.5 Smart wearables system model

Smart wearables are end-to-end integrated gadgets embedded with smart sensors and actuators that connect wirelessly to the smartphone or tablet of theuser, often using Bluetooth Low Energy (BLE) technology. They are usually worn on the wrist, clipped to the body, or hung around the neck for the purpose of staying fit, being more organized, losing weight, staying active, or for tele- medicine purposes.

The model shown in Figure 1.9 is made up of different components, including an athlete, an elderly outpatient, a coach, an emergency team, a doctor, and a weather forecast server. The athlete is wearing ECG sensor and motion sensors, and the elderly patient is wearing ECG sensor, respiratory rate sensor, and motion sensors. These sensors collect and send data to the smartphones via Bluetooth onregular basis, and the smartphones upload the data to a cloud server via cell phone network. The coach can only access information pertaining to the progressof the athlete; similarly, the doctor can only access information pertaining to the health of his patient. However, in case of emergency condition, the emergency team and the doctor will receive an urgent message from the ECG sensor on the athlete, or from the ECG sensor and/or respiratory rate sensor on the patient.

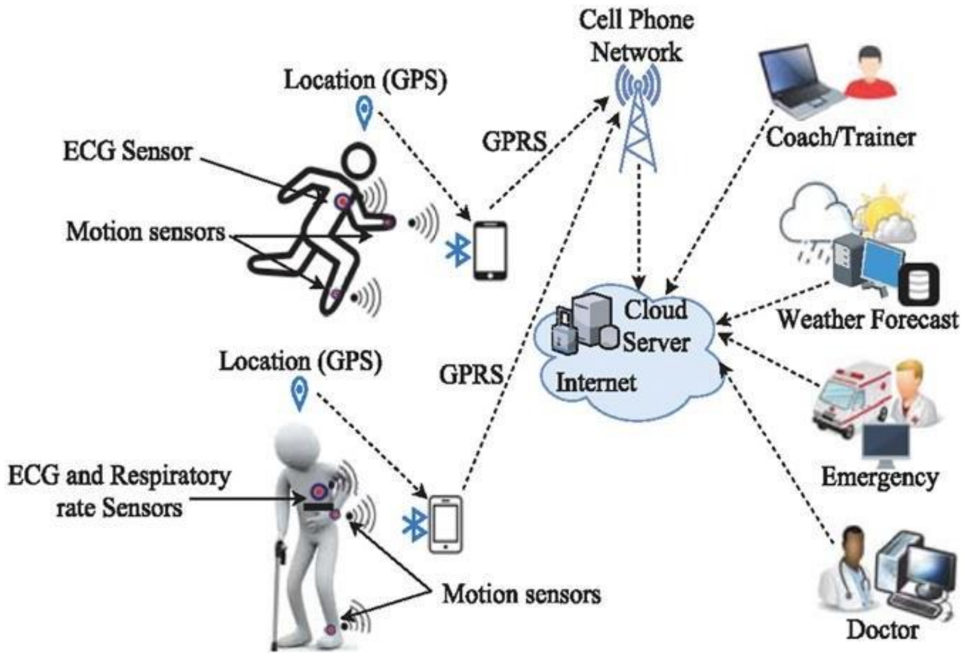


Figure 1.9 Smart wearables system model

1.6.6 Smart healthcare monitoring system model

Smart healthcare refers to a health care paradigm that allows for remote healthcare monitoring and Telehealth, where doctors and other medical practitioners can examine, diagnose and treat patients remotely. Smart healthcare services are becoming commonplace, especially in countries like India. Typical cyber assets in this domain include: glucose monitoring systems, infusion pumps, implantable, pacemakers, insulin pumps, electrocardiograms, medical databases and mobile devices like smartphones.

Figure 1.10 shows a typical smart healthcare monitoring system which consistsof 2 outpatients, a smart hospital, and an emergency team. The 2 outpatients are:a typical patient wearing Electroencephalography (EEG) sensor, Blood Pressure (BP) sensor and Blood Glucose (BG) sensor, and an elderly patient that needs constant monitoring, wearing Electrocardiogram (ECG) sensor and BP sensor. Thesensors on the patient's bodies continuously collect and send data to their smartphones via Bluetooth, and the smartphones in turn upload the data to the medical server via the Internet. In case patients are in critical condition, these sensors can immediately report the physical condition of the patient to the emergency team and to their doctors for appropriate actions to be taken.

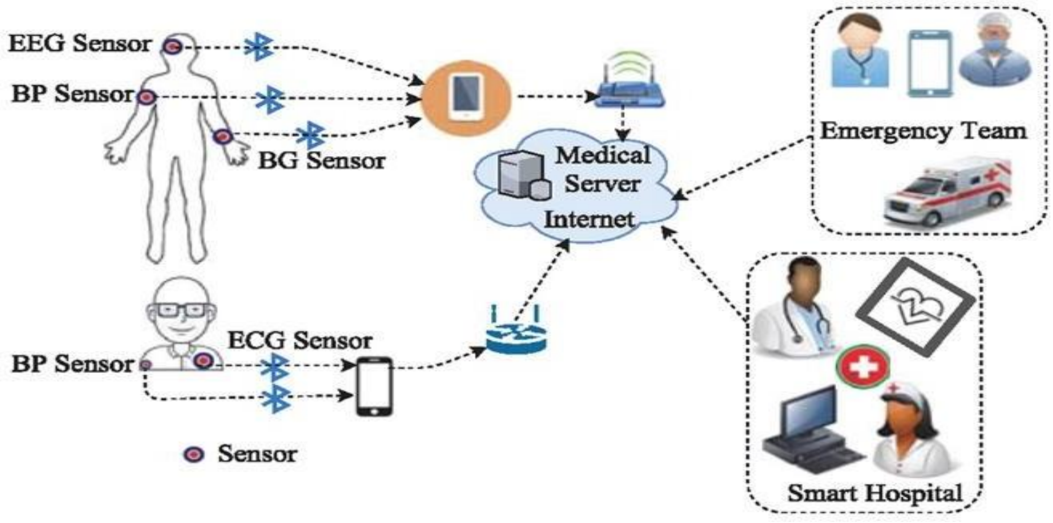


Figure 1.10 Smart healthcare monitoring system

1.8 Challenges

Energy efficiency is the crucial aspect for the IoT. Majority of the IoT devices a resource constrained in nature. Therefore, battery or other constrained energy sources are used to operate it. IoT deployment scenarios are diverse, challenging and sometimes in very remote areas. Humongous IoT network size demands IoT OS to be energy efficient to run the Io devices for many years. Io employs RDC to achieve the energy efficiency. Efficient techniques are required to achieve the accurate motes synchronization along with the RDC. Real time capabilities of the IoT motes is crucial for timely execution of critical tasks. Internet of Body (IoB) requires to meet hard deadlines to achieve certain task. Real time operating system (RTOS) is specifically designed to guarantee completion of these tasks within the certain time frame.

Therefore, IoT OS should have the capability to act as RTOS as well.[6]

1.8.1Network connectivity

is essential for upward and download traffic. Multi-interface may be used to provide multi-homing or to communicate on different spectrum frequencies. Continuous evolution and availability of heterogeneous industry standard protocols at different layers is desirable to provide seamless integration and connectivity of motes to form networks.

1.8.2 Security and safety

of critical systems such as health care, smart home, smart city etc.are highly desirable. In general, IoT OS should support security and privacyof overall IoT network. Open challenges include data integrity, authentication and access mechanisms. Blockchain based optimized solution is one of the promising viable technique to address the privacy and security in the IoT. Further, the deployed network solutions should be continuously reviewed to fix the bugs. Quick development, deployment, testing and adaptation to recent proposed security standards are essential to provide the ultimate network security.

1.8.3Small memory footprint

of IoT OS with the availability of complete TCP/IP stack to run on highly constrained devices is crucial to integrate seamlessly with the global internet. The optimization of the modules in memory efficient manner without losing any functionality is a trivial task. To achieve this, designers and developershave to follow the coding conventions with high degree of ease of configurability and modularity is desirable. Heterogeneous devices support is necessary for the IoTOS. The rapid growth of IoT with diverse use cases leads the way to develop massive heterogeneous devices. Hence, IoT OS needs to constantly incorporate newly developed hardware platform. Explosion of numerous IoT OSs and heterogeneous devices pose another challenge called interoperability. Thus, there should be standard set of rules for development of multiple layers of protocol by the IoT OSs. Consequently, deployment scenarios with multiple use cases that contains heterogeneous devices, running different IoT OSs can seamlessly integrate without an issue.

1.8.4 Intelligent IoT (I-IoT)

are the future key contributor for massive adaptation of IoT in daily life. In recent years, researchers start exploring and applying the artificial intelligence (AI)to IoT use cases. Machine learning (ML) is well explored and investigated in the area of neural networks and image processing. However, its potential and application is not yet fully explored for IoT. ML techniques needs to be optimized to run on IoT constrained devices.

1.8.5 IoT and big data

is closely linked together as IoT devices generate humongous amount of unstructured data. Therefore storage, processing and analyzing big data is essential to generate the meaningful reports to make decisions. This can lead todata-driven research instead of hypothesis driven. New efficient and accurate techniques of data analytics is crucial for development of future innovative solutions.

chapterTwo. **publish subscribe communications in the IoT: the case of e-healthcare application**

2.1 Introduction

Publish/Subscribe is well studied and widely adopted in distributed systems, and has traditionally focused on performance and reliability. Typically, clients are assumed to trust the pub/sub agents (called brokers)to be functioning correctly. In practice, this assumption cannot hold true in multi-federated environments, with limited trust across domains.

Since blockchains are ideal for regulating interactions in a low trust environment, we argue that there is a need to support publish/subscribe messaging in such blockchain platforms. To this end, we propose HyperPubSub, a decentralized pub/sub system which provides secure and privacy-preserving messaging. Pub/sub clients can audit the messaging service by querying the blockchain, validate past operations, and even monetize pub/sub operations. We demonstrate our conceptusing an implementation built with Kafka and Hyperledger. [7]

2.2Publish subscribe

2.2.1Definition

Publish/subscribe (pub/sub) is a many-to-many communication model that directs the flow of messages from senders to receivers based on receivers' data interests. In this model, publishers (i.e., senders) generate messages without knowing their receivers; subscribers (who are potential receivers) express their data interests, and are subsequently notified of the messages from a variety of publishers that match their interests. [8]

2.2.2The Basic Interaction schemes

The publish/subscribe interaction paradigm provides subscribers with the ability to express their interest in an event or a pattern of events,in order to be notified subsequently of any event, generated by a publisher, that matches their registered interest. In other terms, producers publish information on a software bus (an event manager) and consumers subscribe to the information they want to receive from that bus. This information is typically denoted by the term event and the act of delivering it by the term notification. [3]

The basic system model for publish/subscribe interaction (Figure 2.1) relies on an event notification service providing storage and management for subscriptions and efficient delivery of events. Such an event service represents a neutral mediator between publishers, acting as producers of events, and subscribers, acting as consumers of events. Subscribers register their interest in events by typically callinga subscribe () function on the event service, without knowing the effective sources of these events. This subscription information remains stored in the event service and is not forwarded to publishers. The symmetric function unsubscribe () terminates a subscription.

To generate an event, a publisher typically calls a publish () function. The event service propagates the event to all relevant subscribers; itcan thus be viewed as a proxy for the subscribers. Note that every subscriber will be notified of every event conforming to its interest (obviously, failures might prevent subscribers from receiving some events).

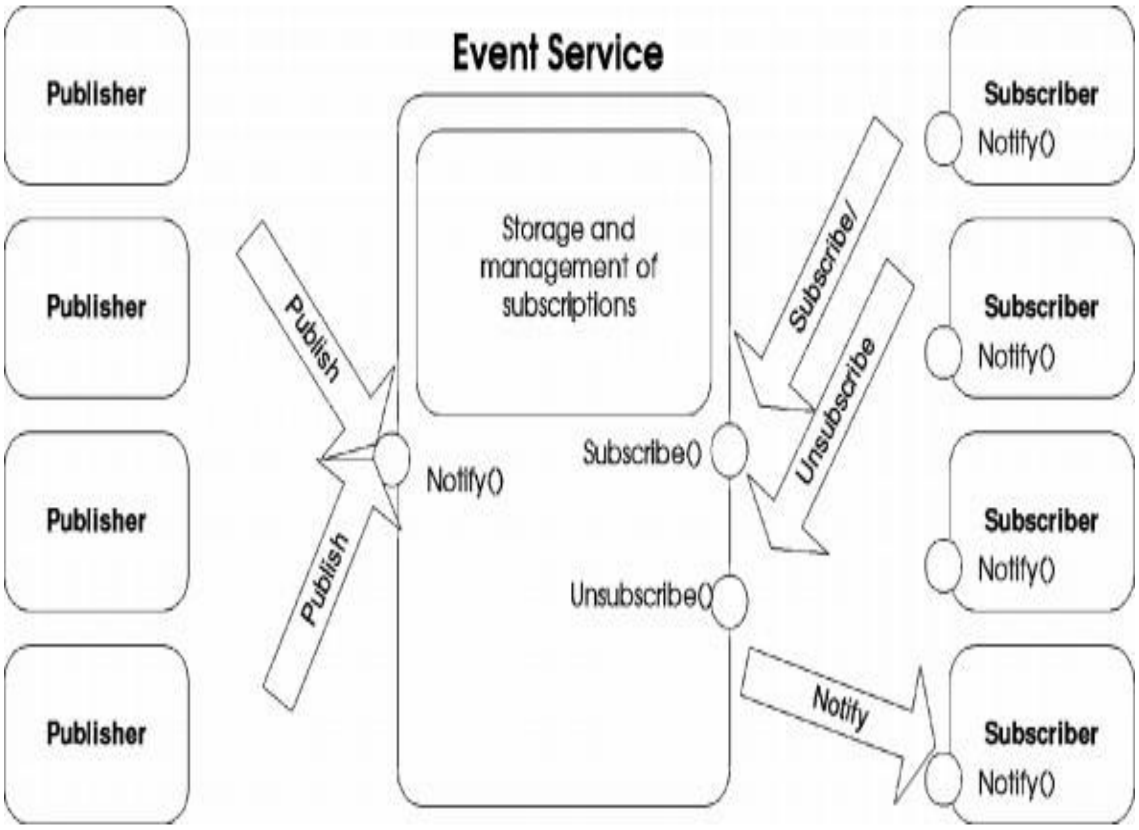


Figure 2.1 A simple object based publish/ subscribe system

The decoupling that the event service provides between publishers and subscribers can be decomposed along the following three dimensions:

2.2.2.1 Space decoupling

the interacting parties do not need to know each other. The publishers publish events through an event service and the subscribers get these events indirectly through theevent service. The publishers do not usually hold references to the subscribers, neither do they know how many of these subscribers are participating in the interaction. Similarly, subscribers do not usually hold references to the publishers, neither do they know how many of these publishers are participating in the interaction.

2.2.2.2 Time decoupling

the interacting parties do not need to be actively participating in the interaction at the same time. In particular, the publisher might publish some events while the subscriber is disconnected, and conversely, the subscriber might get notified about the occurrence of some event while the original publisher of the eventis disconnected. [9]

2.2.2.3 Synchronization decoupling

publishers are not blocked while producing events, and subscribers can get asynchronously notified (through a callback) of the occurrence of an event while performing some concurrent activity. The production and consumption of events do not happen in the main flow of control of the publishers and subscribers, and do not therefore happen in a synchronous manner.

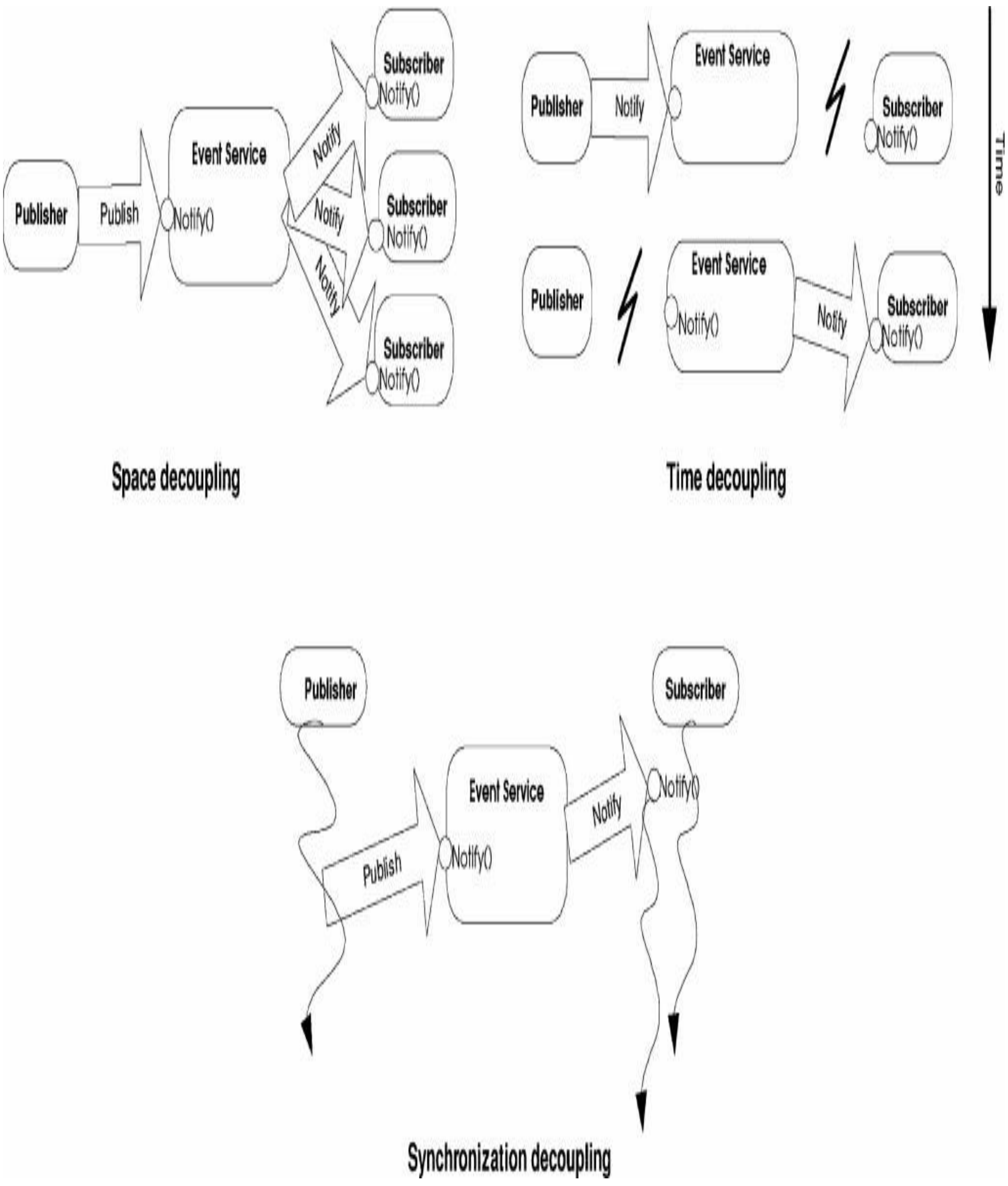


Figure 2.2 Space, time, and synchronization decoupling with the pub/sub paradigm

2.2.3Classification of Pub/Sub Architectures

There are many architectures mentioned as follow:

2.2.3.1Centralized Broker

Consists of multiple publishers and subscribers with centralized broker.

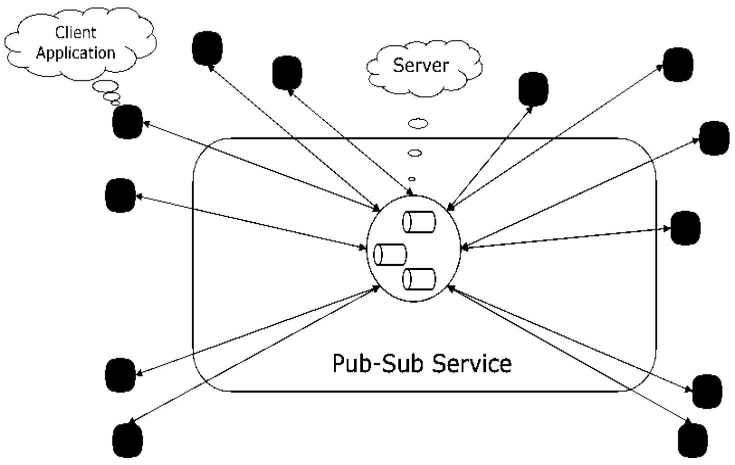


Figure 2.3 Centralized Broker Architecture

2.2.3.2Centralized multi-Broker

Many brokers/servers existed so that each topic can be placed on a different one.

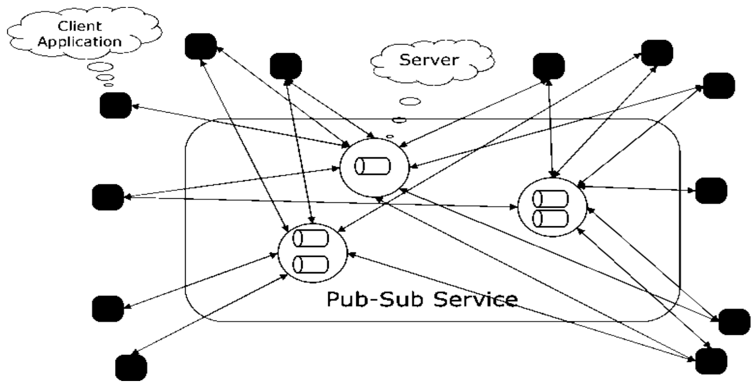


Figure 2.4 Centralized Multi-Broker Architecture

2.2.3.3De-centralized Brokered

Application use messaging to interact with Service Access points. Pub/Sub Service distributes messages internally between servers andcan be peer-to-peer, multicast, etc.

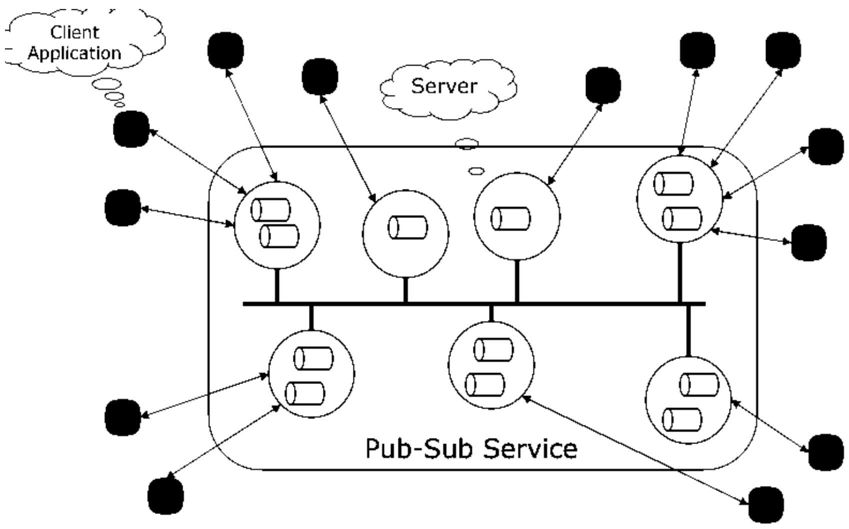


Figure 2.5 De-centralized Brokered Architecture

2.2.3.4De-Centralized Unbrokered (Peer-to-peer)

Each node can be publisher, subscriber or broker. Subscribers subscribe to publishers directly and publishers notify subscribers directly. Therefore, they must maintain knowledge of each other. [9]

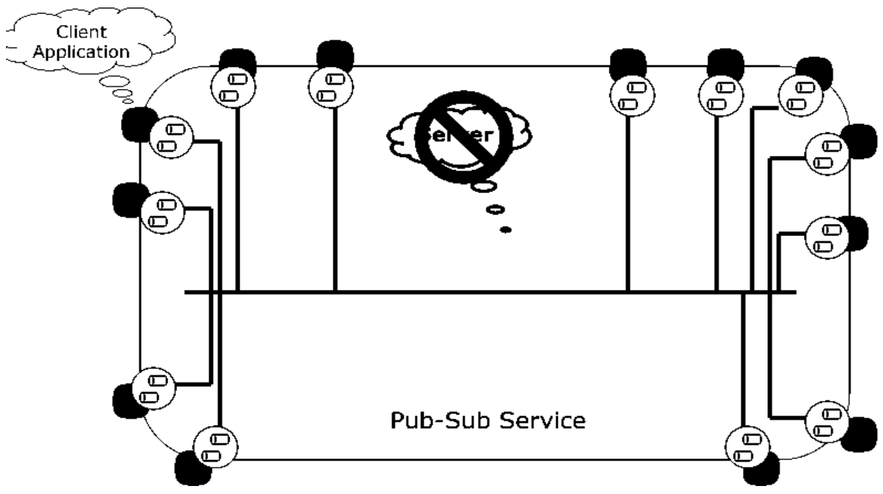


Figure 2.6 Peer to Peer Architecture

2.2.4Pub/Sub Protocols

Many standardized messaging protocols that implement Publish/Subscribe pattern exist. In the area of application-level protocols, the most interesting ones are:

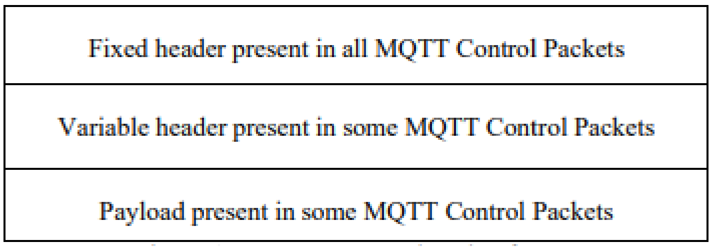
MQTT, MQ Telemetry Transport

COAB

Since our solution is based on the implementation of the MQTT will be explain in the next chapters, the next section will focus on that protocolin details.

2.2.4.1MQTT

MQTT is a standardized publish/subscribe Push protocol that was released by IBM in 1999. MQTT was planned to send a data accurately under the long network delay and low bandwidth network condition [7]. In MQTT for communication, it exchanges a range of control packets in a specify manner. There are fourteen control packets. Eachof contains three parts as illustrated in Figure 2.7.[11]



Fixed header present in all MQTT Control Packets
Variable header present in some MQTT Control Packets
Payload present in some MQTT Control Packets

Figure 2.7 common control packet format

A. Basic Concepts of MQTT

1. Publish/subscribe

In MQTT protocol, publisher publishing messages and users subscribing to topics that are commonly considered as a Publish/Subscribe model. Subscriber subscribes to particular topics which are relate to them and by that receive all messages are published to those topics. On the other hand, clients can publish messages to topics, in such a way that allow all subscribers to access messages of those topics.

2.Topics and subscriptions

In MQTT, publisher publish messages to topics that can be consideredas message subject. Subscriber, thus, subscribe to topics to get specific messages. The Subscriptions of topics can be express, that restricts the data which are collect to the particular topic [14]. Topics contain two wildcard level, to get data for a range of related topics.

3.Quality of service levels

This protocol describes the Quality of Service (QoS) levels that are adeal within two parties of a message with respect to the assurance of distribution of data [8]. It supports three level of Quality of Services which are described below

a) **QoS0** (At most once)

In these Quality levels of service, the messageis sent at most once and it does not provide guarantee delivery of a message.

b) **QoS1** (At least once)

In these Quality levels of service, the data is sent at least once and it is possible to deliver a message more than once by setting the value of duplicate flag by 1.

c) **QoS2** (Exactly once)

In these Quality levels of service, the message is sent exactly once by using 4-way handshaking. The selection of theQoS level depends on the system like if a system needs constant data delivery, adapts QoS2 for transmission of data even if there is a time delay.

4.Retained messages

In MQTT, the messages are retaining in the broker after distributing it to all present clients. At the point when another membership is gottenfor the identical subject, then retained messages of those topics are transmitted to the new customer.

5.Clean sessions and reliable connections

At the point when a subscriber associates with the broker, clean session association is considered as permanent, if its value is false. Inthis task, consecutive messages which come out conveying a highest QoS assignment are reserved for delivery when the association is resumed Use of these flag is optional.

6.Wills

A client can inform the broker that it contains a will (message) which should be distributed to a particular topic or topics in case of an unenticing-pated detach. These will be especially valuable in the system such as security or alarm settings where managers instantly notified just as a sensor has extinct connection with the system.

B. MQTT ARCHITECTURE

The typical MQTT architecture can be divided into two main components as shown in figure 2.8. Each component briefly described below.

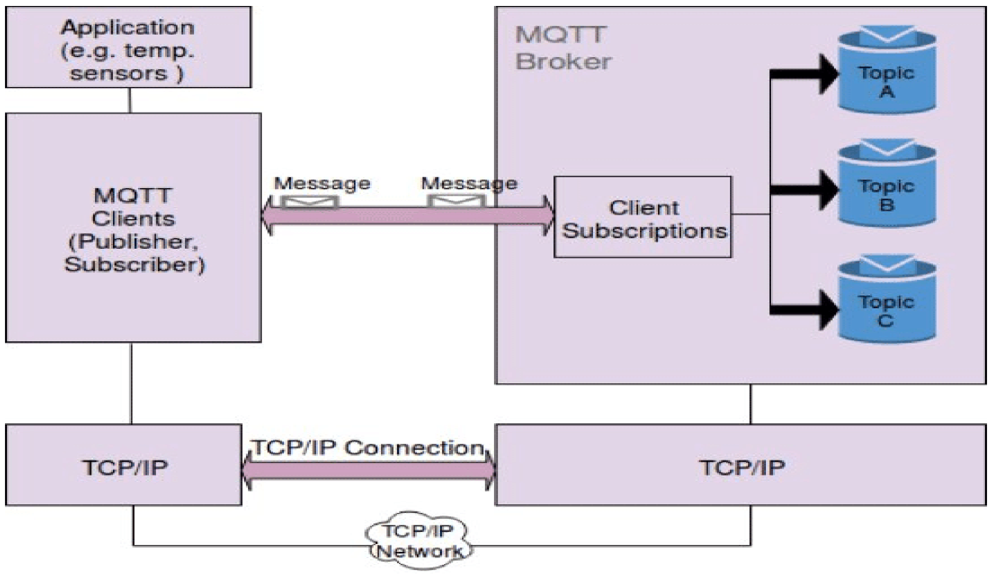


figure 2.8 MQTT Architecture

Client: Client could be a Publisher or Subscriber and it alwaysestablishes the network connection to the Server (Broker). It can do the following things:

- Publish messages for the interested users.
- Subscribe in interested subject for receiving messages.
- Unsubscribe to extract from the subscribed subjects.
- Detach from the Broker.

Broker: Broker controls the distribution of information and mainlyresponsible for receiving all messages from publisher, filtering them,decide who is interested in it and then sending the messages to all subscribed clients. It can do the following things:

- Accept Client requests. Receives Published messages by Users.
- Processes different requests like Subscribe and Unsubscribe fromUsers.
- After receiving messages from publisher sends it to the interestedUsers.

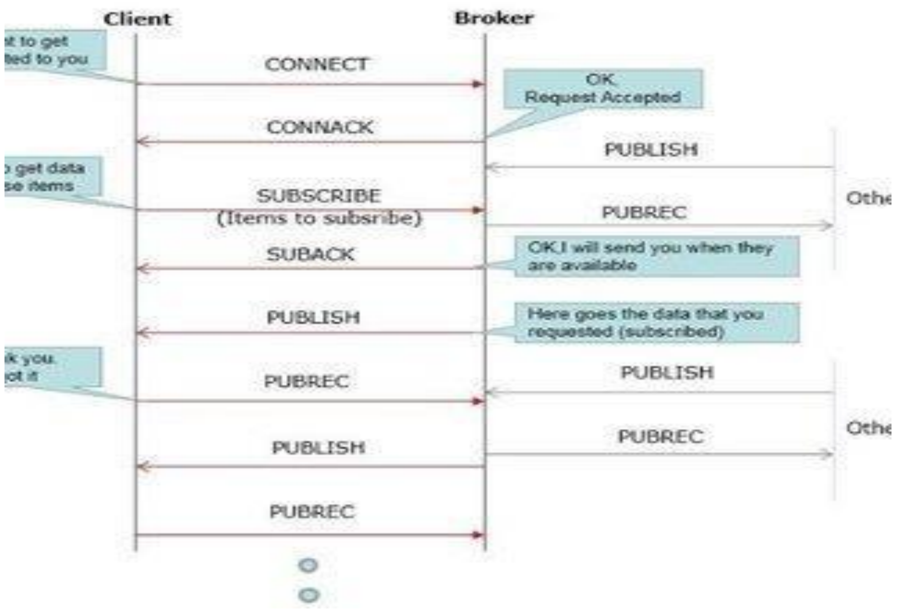


Figure 2.9 Working of MQTT

2.2.4.2CoAP

A. Definition

The Internet of Things Application Protocol (CoAP) was recently created by the IETF's Constrained RESTful Environment Core group. This software protocol built on top of UDP is intended for use in devices with limited and unreliable capabilities such as nodes in wireless sensor networks. CoAP is based on a style of REST (Representational State Transfer) architecture using original HTTP protocol specifications such as 1 'URI and HTTP responses as a representation of resources in various formats (HTML, XML, JSON, etc.) The CoAP architecture consists of two layers:

- Message layer: ensures the sequencing of exchanges from end to end.
- Request Response layer: this layer uses methods and response codes for the Interactions (see Figure2.10 below). [10]

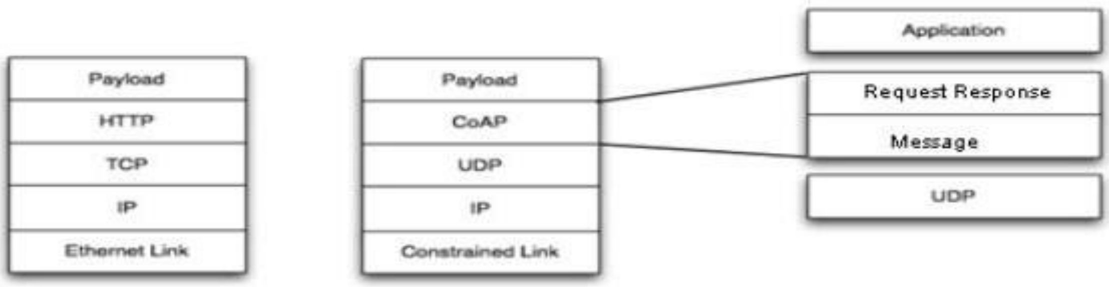


Figure 2.10 Architecture de CoAP

To implement the REST architecture, CoAP takes several concepts from HTTP. Indeed, there is a direct correspondence between the methods of the two protocols, which simplifies the design of a proxyto connect a CoAP network to the "usual" Internet. However, several optimizations have been made to make it more suitable for embedded systems. Firstly, exchanges are made by asynchronous messages, with optional reliability mechanisms; datagram transport protocols (notably UDP, which is used by default) are therefore preferred. In addition, the headers are fairly compressed, to reduce decoding complexity and bandwidth requirements. Finally, several additional features, including simplified proxy and caching capabilities, resource discovery and discovery and observation of resources, increases the interest of the protocol for constrained networks).

B. How the CoAP protocol works

The CoAP protocol uses the UDP protocol at the transport layer, it supports IPv6 for the network layer and uses the 802.15.4 communication protocol at the link layer, if we consider a sensor network. Regarding the message model, a CoAP message has a unique 16-bit Message ID. This allows to identify it and to retransmit the message in case of errors. The message can be of 4 types: The Confirmable CON message: The client waits for an acknowledgement response from the destination server. If at the end of a time T, the client did not have a response, it considers that the packet was lost, and retransmits the message in a random time interval, to control the congestion of the network. TheNon-Confirmable NON message: The client sends the message without waiting for an acknowledgement.

The Acknowledgement ACK message: allows the client to be acknowledged, and therefore to ensure that the message is received correctly.

The Reset RST message: allows the server to reject the request sentby the client.

2.3Overview e-health

2.3.1 Introduction

From the users' perspective, it is evident that lot of benefits exist for the users for being part of e-health systems. It is practical to consider that any person will probably seek the help of different care services and care providers during his or her lifetime. Given the fact that a patient's health record is available electronically and can be shared between different care providers, it would definitely help the patients to convey information regarding previous consultations and diagnosis while also helping the healthcare professionals to make better judgments with the help of comprehensive nature of supportiveelectronic documents Among various messaging protocols for IoT, MQTT is widely used protocol being simple, easy to implement and having small code footprint, but it lacks default security features.[12]

In this chapter, we will target privacy issues and focus on presenting existing solutions based on e-healthcare and on MQTT protocol. Finally, we will compare the studied research work.

2.3.2 E-healthcare

Public healthcare has to deal with a high volume of data which is heterogeneous and distributed widely. A sudden shift to a common paradigm is not possible since too many existing healthcare units are involved which have different systems in place. Privacy and security of the information are also the primary concerns. Apart from the technical aspects, factors like incentives involved in the exchange of information play a crucial role. In this section we discuss these technical challenges in detail.

The distributed nature of healthcare systems results into wide scale heterogeneity. Many of the legacy systems did not incorporate any fixed standard nor do they have the flexibility to modify existing architecture to support them. Hence, interoperability is the primary challenge for healthcare information exchange. In healthcare, interoperability is defined as the ability of different information technology systems and software applications to communicate, exchange data, and use the information that has been exchanged.

1) Another important challenge in health information exchange is the need of a highly scalable architecture. The architecture should be ableto handle high volume of data while respecting the federated natureof the health data. Centralization of data can pose threat to security and privacy of the data, as it creates a large repository of sensitiveinformation. Distributed architecture forms an ideal approach, as it aligns with the federated nature of the health data, but at the same time we may need certain authority that can regulate the information exchange if needed.

2) Security and privacy of the exchanged information are other concerns that cannot be neglected. The information to be exchangedin health care can be sensitive in nature. Some studies suggest thatclinical information should be exchanged on a need-to-know basis. Public health information, though not as sensitive as clinical information, is crucial in the sense that it provides a holistic view ofthe health status. Hence validity of the reported data and authorization of the reporting source are essential aspects which cannot be neglected. [14]

3) The work of gives a generic survey on WSN security. Many of the presented mechanisms may also be used in PEMS but need to be analyzed considering the specific requirements. Although looking at the sensor network alone is not enough, as the backend systems needto be taken into consideration, too. An initial discussion of PEMS security is found in the following figure illustrates some of the current thinking on how to secure wireless sensor networks. The issue is especially relevant in the case of healthcare data. Researchers from Berkeley believe that security for motes must be integrated into every component as components designed without security are the vulnerable points where attacks start. The accuracy and security of vital medical information received from motes must be maintained and is an area for further exploration. If motes move into critical healthcare applications, they may be subject to regulation as medical devices. Of course, performance issues also need to be taken into account as the application should exhibit real-time characteristics.

Another issue that needs to be addressed is to ensure that the huge data stream of information regarding the patients does not overwhelm the medical personnel and does not clutter the information system with unnecessary detail. However, sensors are capable of sending vast streams of personal, private, medical information wirelessly where it could be intercepted by unauthorized people or perhaps mishandled by medical/office personnel who could be unused to handling such large streams of information and who may be unable to work out which data should be kept and which discarded in a safe fashion. The huge data stream of information regarding the patients must not overwhelm the medical personnel and clutter the information system with unnecessary detail. Healthcare providers or others who have access to health information and do not act on it may incur substantial liability. For using PEMs in a limited scenario like Home Monitoring, it might be sufficient to ensure a trusted 1: n connection between a private monitoring system (e.g., a PDA) and a set of sensors. suggests a process called imprinting for this and the European projects MAGNET and MAGNET-Beyond propose to establish trusted "personal networks" composed of trusted personal devices. [13]

chapterThree. **Overview on our solution**

Chapter three Overview on our solution

3.1Introduction

E-healthcare is such an important application domain in IoT. The common objective is to ensure efficient medical monitoring for remote patients. While technology cannot stop the population from ageing or getting ill, it can at least make e-healthcare easier and widely available.

The present chapter is devoted to the presentation of the realized prototype that carries out publish/subscribe communications in e-health.

3.2General Architecture

The general conceptual model of the system we have been working on is shown in Figure 3.1.

This proposed diagram It's made up of a three-section.

The first part consists of a set of medical sensors (pulse heart sensor, blood pressure sensor, blood sugar sensor, temperature body sensor) that we place in the human body, where each sensor picks up values and sends them to Raspberry pi, where each patient has a set of medical sensors attached to his body.

Each type of sensor publishes in a particular type of Topic, for example, we take blue temperature sensor that belongs to topic *temp*.

The raspberry is in contact with the broker at the other end via local network, which represents the Publisher .

The second part consists of a broker, where it receives the values sent from the publisher, and promotes a patient database that contains information for each patient, stores all the sent values, and the computer contains a local server.

The third party is a subscriber, where every doctor can be a subscriber for one or more topics, where he can access the computer's database to consult the readings of the patient's sensors keep being informed of the patient's condition.

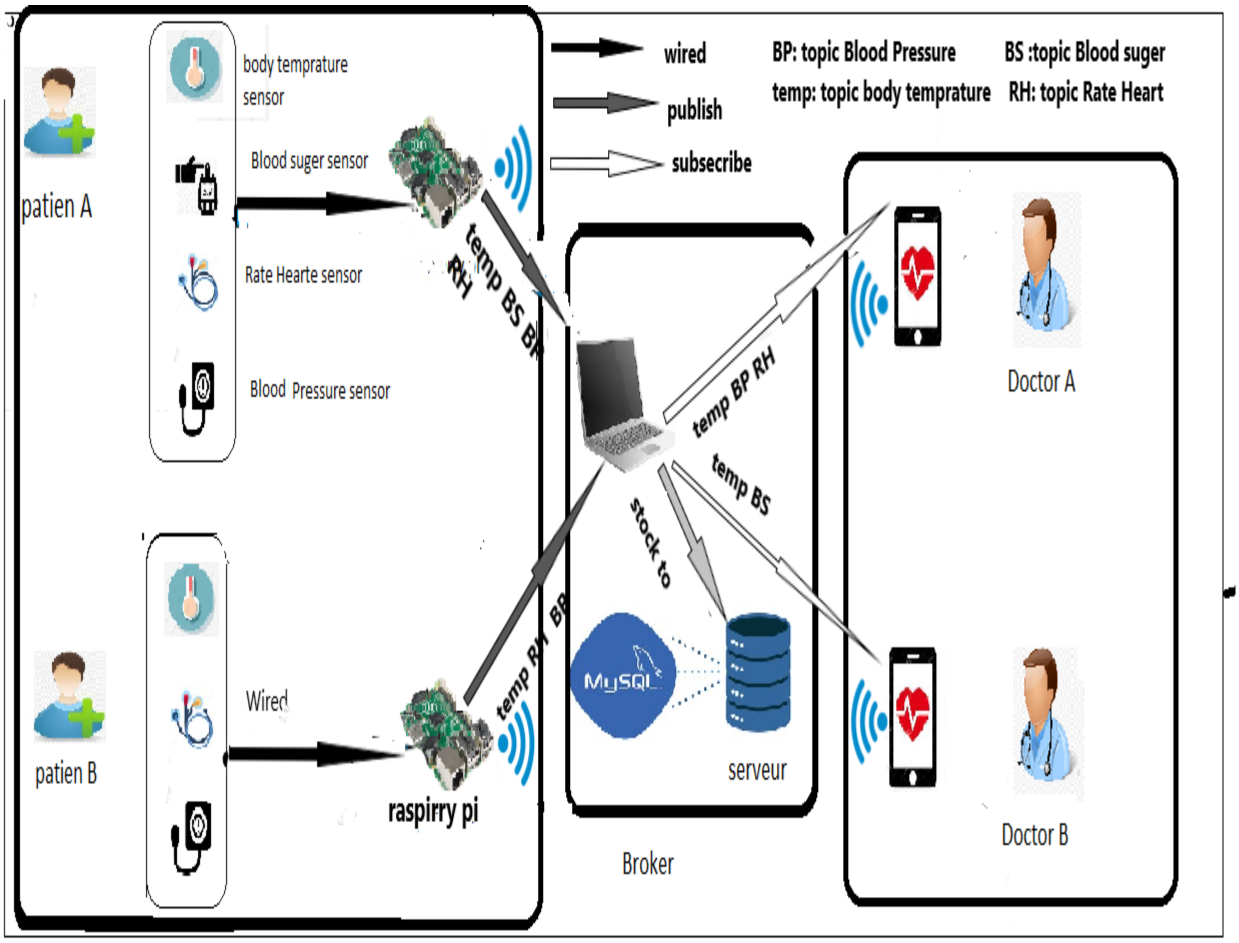


Figure 3-1 *General* Architecture of the considered E –health system

3.3Steps of the realization of the prototype

In this section, we explain the followed steps during the realization of the prototype that are divided into three parts:

3.3.1 Part1: publisher

The publisher part contains raspberry pi as a sensor node controller tied to heart-rate sensor, this sensor is primarily responsible for sensing, collection and acquisition of information or sensor data. pulse heart sensor responsible to measure rate heart. After that, these data will send it to broker with the topic "rate-heart " using port 1883. We chose the broker (mosquitto) because is an open source, free to use, lightweight, support for TLS with client certificate-based authentication, support for authorization using a database and scalable horizontally and vertically (clustering, multithreaded...), the broker mosquitto send the data to subscriber in order to save these data in database using MySQL server database.

3.3.1.1Implementation

In this section, we explain the Implementation of the publisher side. First, we need :

1. Raspberry Pi
2. Pulse Sensor
3. ADS1115 ADC module
4. Jumper wire
5. Power supply or power bank

Before we get into the actual project let us take a look at how the Pulse Rate sensor works. The working of the **Pulse/Heart beat sensor** is very simple. The sensor has two sides, on one side the LED is placed along with an ambient light sensor and on the other side we have some circuitry. This circuitry is responsible for the amplification and noise cancellation work. The LED on the front side of the sensor is placed over a vein in our human body. This can either be your Finger tip or you ear tips, but it should be placed directly on top of a vein.

Now the LED emits light which will fall on the vein directly. The veins will have blood flow inside them only when the heart is pumping, so if we monitor the flow of blood, we can monitor the heart beats as well. If the flow of blood is detected then the ambient light sensor will pick up lighter since they will be reflecting ted by the blood, this minor change in received light is analyzed over time to determine our heart beats. In this project the pulse sensor works as a **heartbeat sensor for Raspberry Pi**.

A. Circuit Diagram

The complete circuit diagram for IoT patient monitoring system using Raspberry Pi is shown in the figure below.

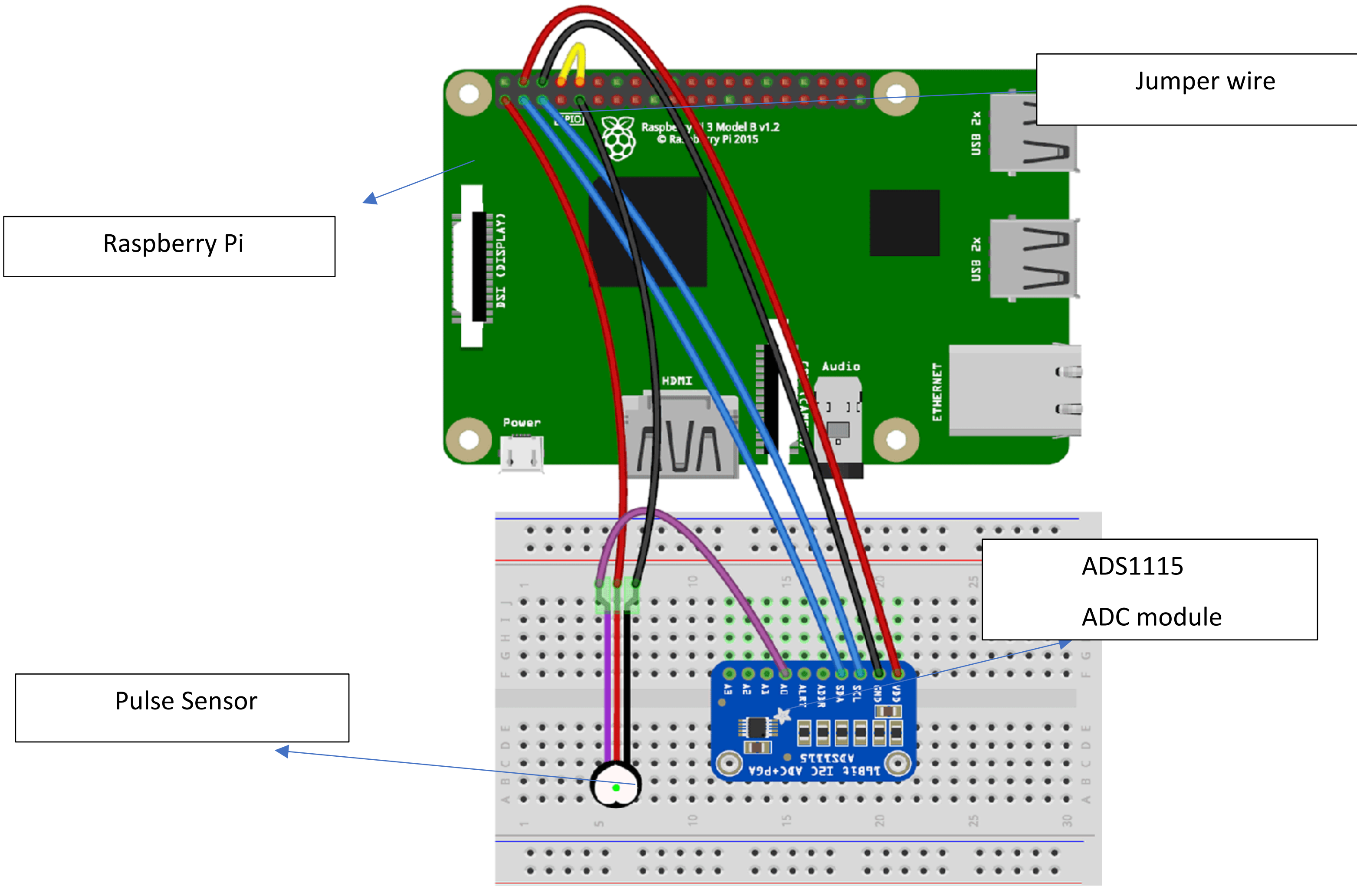


Figure 3.2 the Circuit Diagram of prototype

After that we will create un connection between raspberry pi and a computer (broker) by local network (Wi-Fi), we develop un python code and install paho mqtt library to use it as mqtt client to publish a topic(heart-rate) and send the value of the heartbeat captured in an orderly manner, where every quarter of an hour we send the value.

3.3.1.2 Component of publisher

A. Pulse heart sensor

Figure3-3 Pulse heart sensor

The heartbeat sensor is based on the principle of photoplethysmography. It measures the change in volume of blood through any organ of the body which causes a change in the light intensity through that organ (avascular region). In the case of applications where the heart pulse rate is to be monitored, the timing of the pulses is more important. The flow of blood volume is decided by the rate of heart pulses and since light is absorbed by the blood, the signal pulses are equivalent to the heartbeat pulses.

B. Raspberry Pi

This device works well as a multi-processor. It has a graphics card, a volatile memory, RAM, device interfaces and other external wireless device interfaces. This raspberry Pi is consuming very less power, but it is still cheap and powerful. It requires a keyboard to provide commands, display unit and power supplies as a standard PC. Here, Raspberry Pi used the SD card as a hard disk. Raspberry Pi able to connect via a LAN / Ethernet or via a USB modem or via wireless. Raspberry Pi is supposed to support for various home and business applications. Raspberry Pi runs on a Linux-based OS and which operated by the Raspbian OS. Python is a programming language used to implement the Raspberry-Pi. It is capable of communicating with other external devices using wireless communication technologies, cellular networks, NFC, ZigBee, Bluetooth etc.

FIGURE 3.4 Raspberry Pi model 4

C. ADS1115 Module

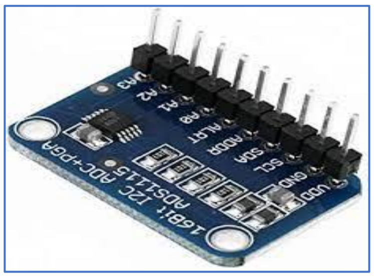


Figure3.5 ADS1115 Module

ADS1115 is an analog-to-digital converter module with a 16-bit resolution. It is a low-power device and operates at a voltage range of 2.0-5.5V. The ADS1115 IC oscillator and communicated to a microcontroller using the I2C communication protocol. It also includes a programmable gain amplifier up to x16, which helps to scale up small/differential signals to the full range.

3.3.2Part 2: broker

In this section, we explain how we dealt with the broker side, firstly, we installed mosquitto broker on a personal computer that is connected to the same local wireless network (wi-fi) and create a local server MySQL for management database. The latter, contains patient-specific information in addition to the stored readings of heart pulses sent from Raspberry Pi.

In this part, we developed a Python code to communicate with the local server MySQL to control and run the database as well as to receive the values sent by the Publisher (Raspberry Pi).

3.3.3Part 3: subscriber

In our case, the subscriber is a doctor that uses its computer or mobile device to check the health status of the patient. For this aim, we have developed a mobile application using Android Studio tool. The application shows through its interface a list of patients, after the doctor select the patient who wants to see the related information, showing him an interface with his information and the topics values that he is interested in (in our Prototype, we have one patient, one Topic and is heartbeat).

The Subscriber can access the database in the Pc with the SQL query and PHP by the local server MySQL, the mobile must be connected to the same local network that the Raspberry and computer connect to.

3.4Modelling of our solution

Figure 3.5 shows the general interaction model of our approach.

Firstly, the patient (publisher) connects to broker. After the reception of the broker’s Acknowledgement message, he registers to topics. In the other side,doctor (subscriber) opens a session too to the broker with connect, than subscribe to topic. The publisher publishes in each topic of interest with its message of data. The broker stores then the received values topics to send them to the doctor.

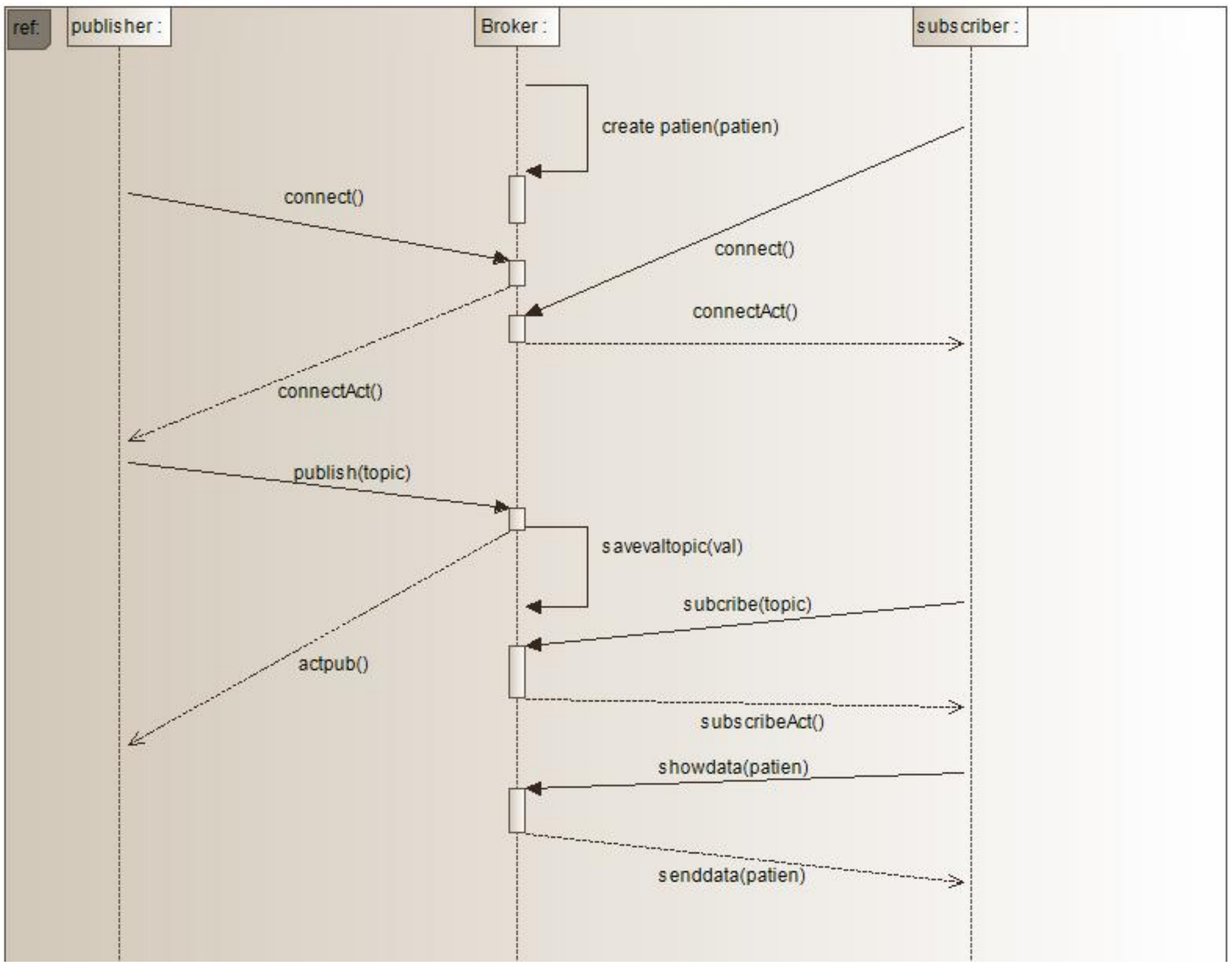


Figure 3.6 Sequence diagram of our approach

3.4.1 Software Description

For the implementation of our project, we need two different kinds of software. First, we have utilized certain open-source programming languages. Secondly, we developed some applications by software IDE, in laptop hp ProBook, RAM 4gb, Hard disk 500gb, processor i5.

3.4.1.1Programing Language

A. Python

“*Python is a high-level programming language designed to be easy to read and simple to implement. Raspberry Pi4 Model B and the Web server code are written in Python language***. “[16]**

B. Mosquitto

“Eclipse Mosquitto is an open source (EPL/EDL licensed) message broker that implements the MQTT protocol versions 5.0, 3.1.1 and 3.1. Mosquitto is lightweight and is suitable for use on all devices from low power single board computers to full servers.

The MQTT protocol provides a lightweight method of carrying out messaging using a publish/subscribe model. This makes it suitable for Internet of Things messaging such as with low power sensors or mobile devices such as phones, embedded computers or microcontrollers.

The Mosquitto project also provides a C library for implementing MQTT clients, and the very popular mosquitto_pub and mosquitto_sub command line MQTT clients.”[17]

C. MySQL

“Is an open-source relational database server.

A database server stores data in separate tables rather than collecting everything in a single table. This improves the speed and flexibility of the whole. The tables are linked by defined relationships, which make possible the combination of data between multiple tables during a query. SQL in "MySQL" means "Structured Query Language": the standard language for database processing.”[18]

D. PHP

“PHP (officially, this acronym is a recursive acronym for PHP Hypertext Preprocessor) is a generalist and open-source scripting language, specially designed for web application development. It can be easily integrated into HTML. The web server is implemented using PHP and MySQL."[19]

F. Java

"Java is a high-level programming language developed by Sun Microsystems. It was originally designed for developing programs for set-top boxes and handheld devices, but later became a popular choice for creating web applications. The mobile application requires both Java and XML languages." [20]

F. XML

"Stands for "Extensible Markup Language". XML is used to define documents with a standard format that can be read by any XML-compatible application [32]. We use the XML language to describe items that may be accessed when a Web page loads. It allows as to create a database of information without having an actual database." [21]

3.4.1.2 Software IDE

A. PyCharm

"The PyCharm development platform is easy-to-use and helps you quickly create embedded programs that work. The PyCharm editor and debugging are integrated in a single application that provides a seamless embedded project development environment. We used it for the implementation of our web server for different reasons such as it makes facilities, source code editing, program debugging, and complete simulation in one powerful environment. "[22]

B. Android Studio

"Android Studio is the official Integrated Development Environment (IDE) for Android app development, based on IntelliJ IDEA_. On top of IntelliJ's powerful code editor and developer tools, Android Studio offers even more features that enhance dfzyour productivity when building Android apps." [23]

3.5 Mobile Application Interfaces

3.5.1 Home Interface

This interface is the main one, it contains a list of names of patient, and button to insert a new patient.

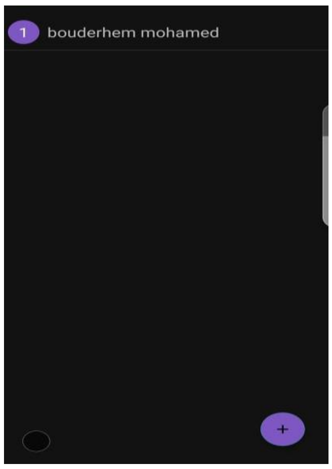


Figure3.7 Home Interface

When doctor selects a patient, it shows a list of options

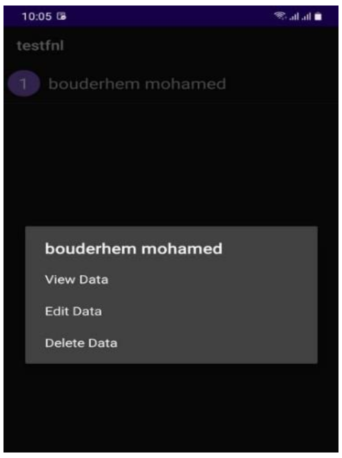


Figure3.8 list option

3.5.2View data option

It shows the data of patient selected.

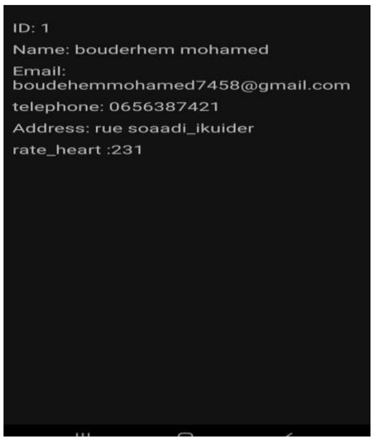

ID: 1
Name: bouderhem mohamed
Email: boudehemmohamed7458@gmail.com
telephone: 0656387421
Address: rue soaadi_ikuider
rate_heart :231

Figure3.9 View data option

3.5.3 Update data option

It can to change information of patient

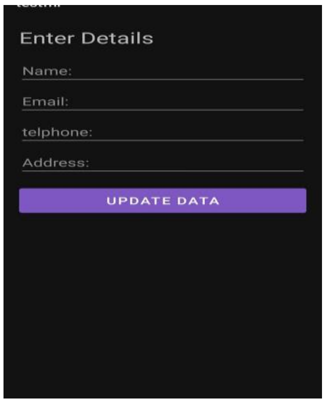


Figure3.10 Update data option

3.5.4Delete data option

It can to delete the patient from data base.

3.5.5Insert patient

This interface can access to data base to insert a new patient by the doctor, just fill out the patient's information.

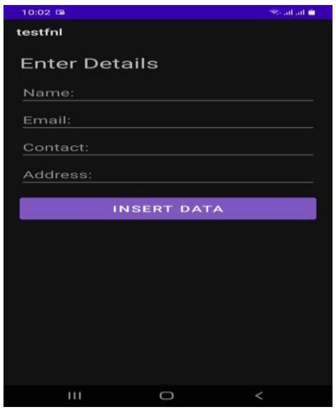


Figure3. 11 Insert patient

Bibliography

[1] S Sahraoui, *Mécanismes de sécurité pour l'intégration des RCSFs à l'IoT (Internet of Things)*, thèse de doctorat, Université de Batna, 2016

[2] Things (IoT) history, technology and fields of deployment." International conference on science engineering and management research (ICSEMR). IEEE, 2014

[3] ZIRARA, Imane. "Privacy preservation for publish/subscribe communications in the context of IoT-connected e-healthcare." (2020).

[4] Khan, R., Khan, S. U., Zaheer, R., & Khan, S. (2012, December). Future internet: the internet of things architecture, possible applications and keycha In 2012 10th international conference on frontiers of information technolc 257-260). IEEE.

[5] Naik, N. (2017, October). Choice of effective messaging protocols for IoT systems: MQTT, CoAP, AMQP and HTTP. In 2017 IEEE international system: engineering symposium (ISSE) (pp. 1-7). IEEE.

[6] Zikria, Y. B., Kim, S. W., Hahm, O., Afzal, M. K., & Aalsalem, M. Y. (2019). Internet of Things (IoT) operating systems management: Opportunities challenges, and solution.

[7] Zupan, Nejc, Kaiwen Zhang, and Hans-Arno Jacobsen. "Hyperpubsub: a decentralized, permissioned, publish/subscribe service using blockchains." Proceedings of the 18th ACM/IFIP/USENIX Middleware Conference: Posters and Demos. 2017.

[8] Diao, Yanlei, and Michael J. Franklin. "Publish/Subscribe Over

Streams." (2009): 2211-2216

[9] Marco Zennaro, Intro to Internet of Things, ITU ASP COE TRAINING ON "Developing the ICT ecosystem to harness IoTs, 13-15 December 2016, pp9.

[10] Hammi, Messouda. "Abstraction de la couche Hardware par un serveur web embarqué pour le web des objets." (2017).

[11] Soni, Dipa, and Ashwin Makwana. "A survey on mqtt: a protocol of internet of things (iot)." International Conference, Power on Telecommunication Analysis and Computing Techniques (ICTPACT-2017). Vol. 20. 2017.

[12] O'Kane, Mentis, & Thereska, Non-static nature of patient consent: Shifting privacy perspectives in health information sharing, Group and Team Issues in the Health Domain, San Antonio, TX, US, 2013, pp23–27

[13] Kargl, Frank, et al. "Security, privacy and legal issues in pervasive ehealth monitoring systems." 2008 7th International Conference on Mobile Business. IEEE, 2008.

[14] Wadhwa, Rakshit, et al. "A pub/sub-based architecture to support public healthcare data exchange." 2015 7th International Conference on Communication Systems and Networks (COMSNETS). IEEE, 2015.

[15] Jie Lin, Wei Yu, Nan Zhang, Xinyu Yang, Hanlin Zhang, and Wei Zhao, A Survey on Internet of Things: Architecture, Enabling Technologies, Security and Privacy, and Applications, IEEE Internet of Things Journal,2017

[16] **https://www.python.org/ viable in 27-06-2021**

[17] **https://mosquitto.org/ viable in 27-06-2021**

[18] **https://www.futura-sciences.com/tech/definitions/internet-mysql-4640/ viable in 27-06-2021**

[19] **https://www.php.net/ viable in 28-06-2021**

[20] **https://www.java.com/fr/download/ viable in 28-06-2021**

[21] **https://www.w3.org/XML/ viable in 28-06-2021**

[22] https://www.jetbrains.com/pycharm/ viable in 28-06-2021

[23] https://developer.android.com/studio viable in 28-06-2021