Bluetooth Low Energy Based CoAP Communication in IoT

----- CoAPNonIP: An Architecture Grants CoAP in Wireless Personal Area Network

A Thesis Submitted to the
College of Graduate Studies and Research
in Partial Fulfillment of the Requirements
for the degree of Master of Science
in the Department of Computer Science
University of Saskatchewan
Saskatoon

by
NAN CHEN

PERMISSION TO USE

ABSTRACT

In recent years, the development of smart devices has brought the Internet of things (IoT) become a popular topic. In IoT, the Constrained Application Protocol (CoAP) is a well-known protocol used in constrained nodes and constrained networks. CoAP aims to work in an IP-based network where a socket is available. However, there are many constrained devices using different scenario to transfer data. For example, in Bluetooth Low Energy (BLE) device use MAC address as an identifier and use Generic Attribute Profile (GATT) to transfer data. Therefore, how to overcome those barriers of communication technologies to support CoAP is a meaningful research field.

There are several approaches to overcome those barriers. For example, a new hardware component can be added to make those device support TCP/IP protocol stacks. Then those devices can easily implement CoAP. On the other hand, an application layer architecture can be added upon existing communication technologies to support CoAP. Considering to minimize the changes of underlying communication infrastructure, the second approach can achieve the goal with less cost.

The objective of this research is to propose an architecture that grants CoAP in different Non-IP based communication technologies. Meanwhile, the research will take Bluetooth Low Energy as an example to explore how to overcome limitations of underline communication technology.

ACKNOWLEDGEMENTS

I would like to express my very great appreciation to my supervisor Professor Ralph Deters. Under his supervision, I have gained great skills in the past two years. The past two years was a wonderful and memorable journey of my life. Under Professor Ralph Deters's guide, I not only achieved marks in academic but also increase skills in programming.

I would also like to extend my thanks to all colleagues in MADMAC lab. Their support helped me to work through the tough time at the first year and granted me confidence to overcome problems.

At last, I want to thank the support of my parents and family members. Without their support, I can not study and work smoothly in the past two years.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATION

IoT	Internet of Things
CoAP	Constrained Application Protocol
WPAN	Wireless Personal Area Network
MQTT	MQ Telemetry Transport
REST	Representational State Transfer
SOAP	Simple Object Access Protocol
BLE	Bluetooth Low Energy
SIG	Special Interest Group
NFC	Near Field Communication
IPSP	Internet Protocol Support Profile
PA	Partition Tolerance and Availability
PC	Partition Tolerance and Consistency
IP	Internet Protocol
IPV6	Internet Protocol Version 6
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
MAC	Media Access Control
GATT	Generic Attribute Profile
MTU	Maximum Transmission Unit
L2CAP	Logical Link Control and Adaptation

CHAPTER 1 INTRODUCTION

The boom of the smartphone market and the development of wearable devices are introducing low energy sensors and personal hubs (phone, tablet or other portable devices) with increasing rate. In this trend, we find that the role of the smartphone has shifted from a single function device to an integrated personal data hub. On the other hand, with the development of tiny sensors, more and more small devices are installed to collect data as the new edge of Internet. Since more and more people have multiple smart devices to interact with each other, the communication between personal hubs and edge devices becomes increasingly important.

There are multiple technologies to support short range wireless data communication like BLE and NFC. However, different technologies have the different restrictions on data transfer. Although, technologies like MQTT and COAP are available to talk with different machines running in IP cluster, IP address is an essential requirement in the current implementation. Therefore, there is always a protocol transfer to consume resources through WPAN technologies like BLE and NFC. There is still a great room to improve in merge edge nodes into IoT. For example, Johanna proposed a hardware solution to install IPv6 in BLE devices by adopting 6lowpan (Isomaki, M., Nieminen, J., Gomez, C., Shelby, Z., Savolainen, T., & Patil, B., 2011) . However, there are no efforts being made on implementing a software layer upon existing architecture of BLE.

In this research, I will take BLE as an example to show how to merge lightweight Non-IP based wireless communication technologies into IoT context by using the proposed architecture. The remaining sections of the thesis are constructed as follows:

- Chapter 2: Discuss what problems need to be solved in the research.
- Chapter 3: Review related work.
- Chapter 4: Discuss proposed architecture in details.
- Chapter 5: Explain key implementations.

- Chapter 6: Discuss experiments for performance test.

CHAPTER 2 PROBLEM DEFINITION

As mentioned above, the research aim is to propose an architecture to accelerate the process of merging NonIP based technologies into IoT. To achieve this goal, we need to develop a lightweight protocol to carry information as well as a set of mechanisms to guarantee data consistency. Since one or more apps in one device need to communicate, the architecture also needs to run as a background service.

In short, we need to answer the following key questions:

1. How to unify communication language between Non-IP and IP-based technologies?

2. How to overcome limitations of standard BLE communication?

3. How to serve multiple applications as a background service?

4. How to provide a common interface to support different technologies?

2.1 How to unify communication language between Non-IP and IP based technologies?

The concept of IoT is firmly associated with IPv6. It tags things by assigning a unique address (usually IPv6) to each of them. However, in the real world, not all things tagged following the same rule (IPv6). For Example, BLE uses MAC address to identify devices. To unify communication language between Non-IP and IP-based devices, the following two things are necessary: 1. A software layer identifier. 2. A protocol which is independent of IP address and MAC address

2.2 How to overcome limitations of standard BLE communication?

Since BLE is designed for low bandwidth data transfer, some limitations need to overcome, in order to merging it with existing IP-based IoT networks.

2.2.1 Size limitation of a BLE request

In a standard BLE communication, GATT profile is a must, in which the payload of a characteristic is limited. According to Bluetooth4.0 specification (SIG, 2010). "All L2CAP implementations shall support a minimum MTU (maximum transmission unit) of 48 octets over the ACL-U logical link and 23 octets over the LE-U logical link". According to my experiment, the available payload size of a characteristic is 20 bytes. However, in IoT, one packet can easily extend 20 bytes. Therefore, the architecture should be able to cut long byte array messages into short sub packages and assemble them at remote side. We need a simple protocol, as well as relevant pack and unpack mechanism to solve this problem,

2.2.2 Chaos server-side callback

In BLE, a server may serve multiple clients. Unlike classic Bluetooth where each connected client can create its own socket to maintain communication with the server, BLE server handles all requests for a certain operation in one callback function. In brief, one callback function may be triggered by different clients. Therefore, the proposed architecture must provide a mechanism to handle messages from different clients in one function.

2.3 How to serve multiple applications as a background service?

The consumer of a specific request may come from different applications in one device, which means the communication between provider and consumer are not at device level but application level. Thus, the architecture must be able to identify a certain piece of message come from which application at which device.

2.4 How to provide common interface to support different technologies?

Beside BLE, other Non-IP based wireless communication technologies have the same requirement to take advantages CoAP. We can support those technologies by proposing an abstract interface.

2.5 Research Goal

The goal of the research is to propose architecture to accelerate the merging of WPAN technologies into IoT.

It is composed of the following three sub-goals.

Goal 1. Propose a solution for identifying Non-IP based devices.

Goal 2. Put forward a protocol and architecture to support CoAP communication in BLE.

Goal 3. Design the architecture to support multiple apps as a background service.

Goal 4. Provide abstract interface to support other WPAN technologies.

CHAPTER 3 LITERATURE REVIEW

3.1 IoT

In recent years a new concept named "IoT" is popular. IoT is a concept of making real world things become available through The Internet. It is happening and will fundamentally change our world. A background of this technological trend is the increasing number of smart devices. According to an analysis of IoT from Cisco in 2011 (Evans, 2011), by 2010, the number of connected devices has exceeded the population of human beings. In 2020, 50 billion devices will be connected by IoT.

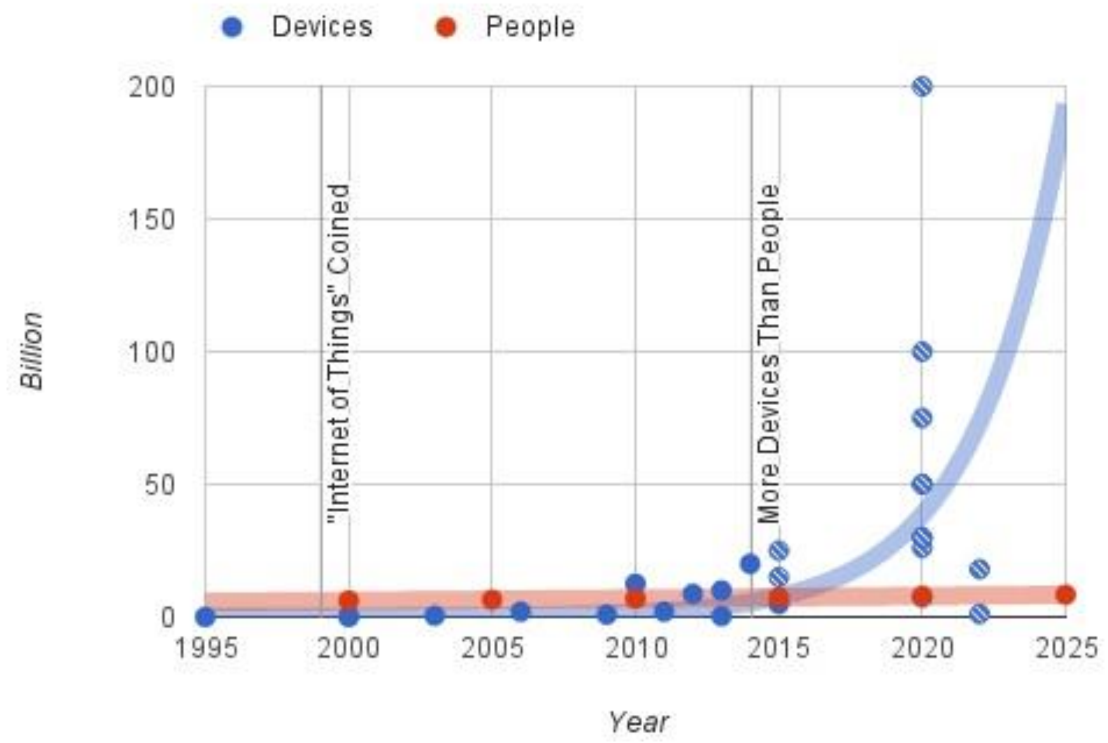


Figure 3.1 Trend of devices vs people (Barrett, 2012)

As shown above, recent research from Philip N. Howard has demonstrated this trend. Between 2011 and 2015 the connected device has experienced exponential growth.

The concept of IoT is not only a simple extension of Internet. It introduces a way to make physical devices become available in the virtual world. In order to make a real object become a "thing" on the internet, according to Dr. John Barrett (Barrett, 2012), there are several essential steps to convert an ordinary object in daily life into a smart object in IoT.

First, a unique identifier to tag a thing. Every asset in the IoT has a tag to uniquely identify itself. In this context. IPv6 is the solution for the first step because of its huge volume. Besides IPv6, IPv4 and MAC are also widely being used. However, IPv4 has a fatal limitation of its capacity. With 32-bits space, IPv4 has only 4,294,967,296 addresses which will be used up soon. In contrast, the IPv6 has 128-bits space available. Similarly, the MAC address is designed to identify a network interface on the physical network level. It adopts 48-bits address space which is better than IPv4. However, the capacity of MAC is still not enough to tag all objects on earth. Therefore, IPv6 is the best solution we have.

Second, the ability to communicate. In order to communicate with the internet, smart objects or called things in IoT must implement a way to communicate. Therefore, different transmission media need different communication mechanisms. Today, we mainly use microwave and twisted-pair as the front end media.

Third, give object senses. In the real world, an object can be identified by smell, feeling, color and so on. It is also true in the digital world. By collecting different data of an object, people can know changes of an object or its surrounding environment. Therefore, people need to implement different kinds of sensors on an object to make it become "smart". By implementing sensors, we acquire desired data of an object to share on the internet.

Fourth, a controller at remote side. Since we can get data from sensors and deliver them to a remote object through the internet, the remote side may have requirements to modify some values of a smart object through Internet. Therefore, people need a remote controller to achieve remote control.

After above four procedures, we can make sure that an object becomes smart. As a smart object, it can take part in data communication of IoT.

The highlight of IoT is the concept of gathering data from the machine and transmitting data between with pre-programmed procedures, which means less manual operations are required in IoT. In this way, it liberates more productive forces from input data to The Internet to more valuable work.
As a basic concept of ¨smart homes¨ and ¨wearable devices¨, the IoT is a promised coming future. It will make a great influence on our daily life.

In the section, we will discuss the concept of personal cloud which helps us define the edge of proposed architecture.

3.2 Personal Cloud

The personal cloud is a combination of the private cloud and the public cloud. According to Sang-Ho Na, Jun-Young Park and Eui-Nam Huh (Na,S.,Park,J.&Huh,E., 2010), “The Personal Cloud describes a user-centric model of Cloud computing where an individual's personal content and services are available anytime and anywhere, from whatever device they choose to access it.”.

In fact, there are business models available for years. For example, Seagate proposed a personal cloud solution (Seagate, 2016) which aims to provide a centralized media library where a user can access to their data from anywhere. The model consists of software on different platforms and a central server. People can access their digital assets on different platforms through different interfaces but their data are backed up in one physical device.

In recent years, with the increasing number of personal mobile devices (like smartphones, pads or wearable devices), more and more digital assets are distributed on different devices. For most of the time, those devices are not all in the radiation range of a Wi-Fi network. In this case, those scattered personal devices need to connect to a center node or a bridge device to reach the

Internet. Therefore, there are two important facts of a short-range wireless communication technology.

First, the data communication between mobile devices is relatively less intensive than PC to PC communications. It depends on the computing power of mobile devices.

Second, the network state is constantly changing from time to time. It is determined by the nature of short-range wireless communication.

After examining the context of IoT, we found that more and more smart devices are available for a single person. Those smart devices may be placed in a particular place or taken by their owners as portable devices. In order to let those devices working together to serve us, I would like to explore.

In the next section, we will discuss two popular design styles for the machine to machine communication: SOAP and REST.

3.3 SOAP and REST

In general, SOAP is a protocol which was popular in the late 1990s. REST is a design style which has been proposed with HTTP but is not popular until recent years. Technically, SOAP and REST are two different kinds of concept and not directly comparable. Here we put them together just because each of them stands for a style of design.

3.3.1 SOAP

SOAP is the abbreviation for Simple Object Access Protocol. It is a widely adopted protocol for data exchanging among web applications. Both SMTP and HTTP are notable transport protocols which support SOAP well. The format of SOAP message is based on Extensible Markup Language (XML).

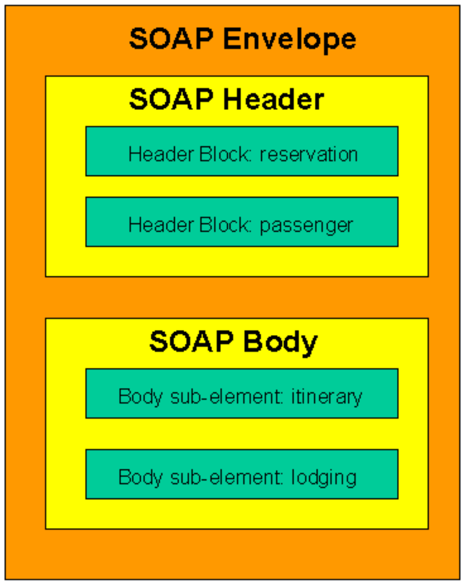

SOAP Envelope
SOAP Header
Header Block: reservation
Header Block: passenger
SOAP Body
Body sub-element: itinerary
Body sub-element: lodging

Figure 3.2 SOAP message structure example (Mitra & Lafon, 2007)

As shown above, a SOAP envelope consists of two sub-elements: Header and Body. The Header contains metadata which are optional. The Body contains the payload.

The benefit of SOAP is obvious. It allows internet communication between programs with the support of different transport protocols. Meanwhile, because it follows post/response mechanism in HTTP, it can easily pass through firewalls and proxies.

3.3.2 REST

REST stands for representational state transfer. It is proposed by Fielding, Roy Thomas in 2000. In REST, each available resource at server side can be visited through a consistent path (URI). The client can request different operations (Create, Read, Update, and Delete) through standard HTTP verbs (POST, GET, PUT, and DELETE). Although operations in REST are defined by HTTP verbs, REST is not bound with web service.

According to the author (Fielding, 2000), the advantage of REST is obvious because it "emphasizes scalability of component interactions, the generality of interfaces, independent deployment of components, and intermediary components to reduce interaction latency, enforce security, and encapsulate legacy systems”. However, those features not just bring benefits but also limitations as well. According to whatisrest.com (Arcitura, 2016), there are six constraints for a REST architecture.

- Stateless: Stateless is the most important feature of the RESTful design. It indicates that each request from client contains all information a server needs to know, and all session state data should be sent to a client after each request.
- Cache: A RESTful design should support cacheable mode by which people can save the latest response for further usage.
- Uniform Interface: It is the fundamental characteristic of a REST service which guarantees each request can independently get an individual resource.
- Layered system: An REST-based solution may consist of multiple layers. Communications between service providers and consumers are independent events that can not be affected by changes in other layers.
- Code-On-Demand: It is an optional constraint. A client can update its codes independently from a server.

The RESTful design makes web application back to the original purpose of HTTP. Since many existing systems adopted SOAP design style, martin Fowler (Fowler, 2010) proposed three ways towards REST (from a traditional HTTP web service to a RESTful service).

- Level 1. Identify every resource: Developers need to design URI to guarantee one to one mapping between resources and URIs.
- Level 2. HTTP Verbs: Use appropriate HTTP verbs under different situations instead of using POST only.
- Level 3. Hypermedia Controls: Grant discoverability to response. By adopting hyperlinks in response, users can always get links for next possible operation.

3.3.3 REST VS SOAP

It is common that REST and SOAP are put together to make a comparison. Theoretically, SOAP is a protocol. It should not be compared with REST. However, those two terms define two

popular ways to design services. According to Cesare (Pautasso, C., Zimmermann, O., & Leymann, F., 2008.April), “REST is well suited for basic, ad hoc integration scenarios, WS-* is more flexible and addresses advanced quality of service requirements commonly occurring in enterprise computing” (Ws-* refers to Web service specifications).
In SOAP, all requests and responses must follow XML standard as well as use the verb: POST. In REST, all four verbs are available and each URLs map to resources. The following section will discuss those two technologies in detail.

- Complexity: SOAP adopts XML as transfer format, which means the complexity of serializing and de-serializing is certain. In contrast, REST can adopt different protocols to transfer data. For example, comparing with XML, JSON can store more information with fewer words. On the other hand, in SOAP, operations are all encapsulated in a POST request, which makes it become a black box for users. In contrast, operations in REST are defined by four HTTP verbs. Therefore, the user can easily tell the target resource and action of a request.

- Scalability: SOAP was popular when IT services were only available for giant companies. It is designed to fulfill a particular task, which makes codes are hard to scale. On the other hand, REST maps one URL to one resource. It divides complexity of a single page into multiple pages. This characteristic naturally increases the scalability of a system.

- Cache: REST is a natural suit for caching. As mentioned above, at the last step towards REST, possible operations of next request are included in the return. In this way, the client-side can easily load and store result before the user makes a request. In SOAP, resources are always wrapped together which makes the next operation of a user is hard to predict. This characteristic makes cache becomes much harder in SOAP.

- Security: Because of WS-Security, many people believe SOAP is more secure than REST. However, it is not true. According to Flanders’ research(Flanders.J, 2009),” the WS-* arena certainly has more standards than the RESTful arena (and this will probably

> always continue to be the case), but there are efforts to support federated security in the world of REST. OpenID is one such effort” (Ws-* refers to Web service specifications).

In conclusion, we can not simply state SOAP is better than REST or REST is better than SOAP. In order to make a right architectural decision for the proposed architecture, developers need to take more factors into consideration.

3.3.4 REST VS SOAP in mobile app

For a mobile app, where the internet connection is fragile (nature of wireless connection), one single communication between client and server is expected to be short and light. As mentioned above, compared with SOAP, REST is more lightweight, flexible and controllable. Those characteristics meet the requirement of the wireless connection between mobile devices and the internet. On the other hand, SOAP is not a bad option when mobile apps are used to serve web applications for complex business logic.

3.3.5 Conclusion

From the analysis above, it is obvious that REST is more simple and clear. In REST, each URI has clear meaning and operations are limited within POST, DELETE, UPDATE and GET. On the other hand, in general, there are more logics in a single page of a SOAP architecture. The XML makes the SOAP can handle complex logics but also increases complexity.

In summary, REST prevails over SOAP in lightweight communication. It is especially true at the edge of the network where lightweight payload is required.

After the comparison, I chose REST as the design style of proposed architecture. Since REST is chosen, I turn to explore popular protocols based on REST.
In the next chapter, we will discuss advantages and disadvantages of CoAP by comparing it with MQTT.

3.4 MQTT and CoAP

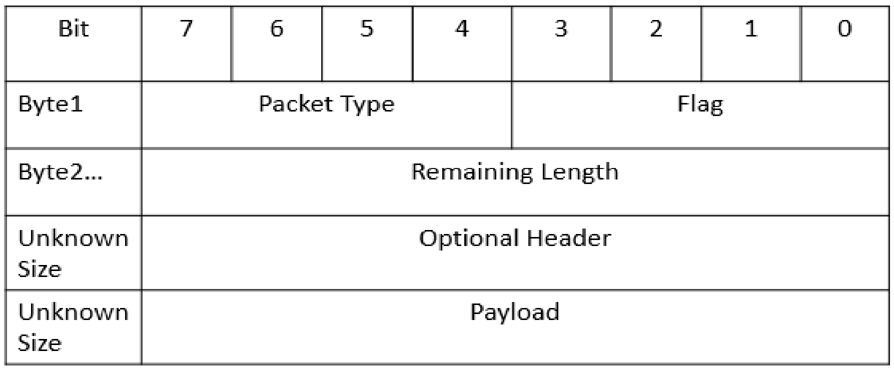
MQTT and CoAP are two popular protocols in IoT. In this section, we put MQTT and CoAP together to discuss features of CoAP.

3.4.1 MQTT

MQTT stands for MQ Telemetry Transport. It is a lightweight publish-subscribe protocol running on TCP/IP protocol. According to its official website, (OASIS Technical Committee, 2014) “MQTT is a Client Server publish/subscribe messaging transport protocol. It is light weight, open, simple, and designed so as to be easy to implement.”

3.4.1.1 Packet Format

According to 3.1.1 specification (OASIS Technical Committee, 2014), MQTT packet has two bytes fixed header. Therefore, the minimum size of an MQTT packet is two bytes. The following chart shows how an MQTT packet looks like according to the description of the official document.



Bit	7	6	5	4	3	2	1	0
Byte1	Packet Type				Flag			
Byte2…	Remaining Length							
Unknown Size	Optional Header							
Unknown Size	Payload							

Figure 3.3 MQTT format

As shown above, the first byte of MQTT consists of two parts. Bits from 7 to 4 indicate the packet type. Bits from 3 to 0 set flags. Bytes since the second indicate the remaining length of the message. The minimum size of this section is one byte and the maximum is four bytes. After specifying remaining size of a packet, the developer can add optional headers to a packet. After the optional headers, the remaining section of a packet is payload.

3.4.1.2 Feature

Publisher and subscriber (pub/sub): In this model, the publisher and the subscriber do not know about the existence of each other. Instead, a broker will gather all published messages from a publisher and accordingly send them to the subscribers after filtering. In other words, there is a central server between publishers and subscribers.

QoS: MQTT supports three levels of QoS. At level 0, a message will not be acknowledged or resent by a sender. At level 1, a message will be guaranteed to deliver at least once. At level 2, the receiver will guarantee the message to process. In this case, the sender stores a message and is ready to resend of it. Meanwhile, the receiver stores the reference of a received packet’s id to prevent from processing the same message twice.

Last Will and Testament(LWT): The client can register a LWT when a connection is initialized with a broker. If a client has been disconnected from a broker, the broker will send LWT to all clients subscribed the “lastWillTopic”.

In conclusion, the MQTT is a well-designed protocol for lightweight bandwidth. Its logic is simple where all communication between clients is based on message transition of a broker.

3.4.2 CoAP

CoAP is short for Constrained Protocol. It is based on REST model and designed for constrained networks. According to RFC 7252 (Shelby, Z., Hartke, K., & Bormann, C., 2014), UDP is the default transport protocol for it. Meanwhile, it could also be used over other transports such as TCP. CoAP offers an elegant solution for low bandwidth communication with HTTP-like mechanisms.

3.4.2.1 Packet Format

CoAP has a four-byte header which includes information of “version”, “token”, “length of variable-length token field”, “message ID” and code of “message type”.

```
  0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|Ver| T |  TKL  |      Code     |          Message ID           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   Token (if any, TKL bytes) ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   Options (if any) ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|1 1 1 1 1 1 1 1|    Payload (if any) ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 3.4 CoAP format (Shelby, Z., Hartke, K., & Bormann, C., 2014)

As shown above, the header of a CoAP message consists of five parts. The first 2 bits of a message shows CoAP version of the message. The following 2 bits indicates message type (4 options available). TKL stands for Token Length (4 bits), which indicates the length of the token field.

The second byte is “Code” which indicates operation of the message. There are two types of Code. One is called “method code” which consists of 4 verbs in HTTP (GET, POST, PUT and DELETE). The other is called “response code” which returns the result of a request. For example, CREATED, DELETED, VALID and etc.

The third byte and fourth byte store Message ID.

Besides the four-byte header, token and options are additional information which is neither part of header nor payload. The size of token ranges from to 0 to 8 bytes. It is used to match the request and the response. CoAP also supports some metadata in HTTP. In the Options section, a user can define parameters like in HTTP header, such as Etag, Max-Age, Content-Format, etc. Those options simplify the process of translating COAP messages into HTTP messages.
Unlike MQTT where the remaining length is defined to indicate the length of the optional header and payload size, in CoAP, a 1-byte flag is used before the payload. When a processor first time encounters a 0xFF, it indicates that the payload starts from the next byte.

3.4.2.2 Feature

CoAP has several features. I list the most interesting four as the following:

- URI support: CoAP follows a URI scheme which is similar as the one in HTTP.

- Similar features to HTTP: Operations in CoAP are based on four HTTP verbs (GET, POST, PUT, and DELETE). Metadata and URI are optional in CoAP. Those two characteristics make it easy to convert messages to HTTP messages.

- Resource discovery: There is a special URI in CoAP defined as "/.well-known/core". By sending request to this URI, a client can get all available resources of a server.

- Observation: CoAP supports data observation as an extra option besides traditional “send” and “receive”. At server side, a developer can add observable resources to the observer list. Then, a client can register resource observer to get data change notifications.

From the highlights introduced above, we know that CoAP is a compatible protocol designed for IoT. Although it is efficient, it does not aim to replace HTTP. It is designed to support constrained networks under the consideration of merging them into existing networks. As shown in the following chart, we expect the CoAP to handle medium or small size messages as a bridge between fat web services and low payload networks.

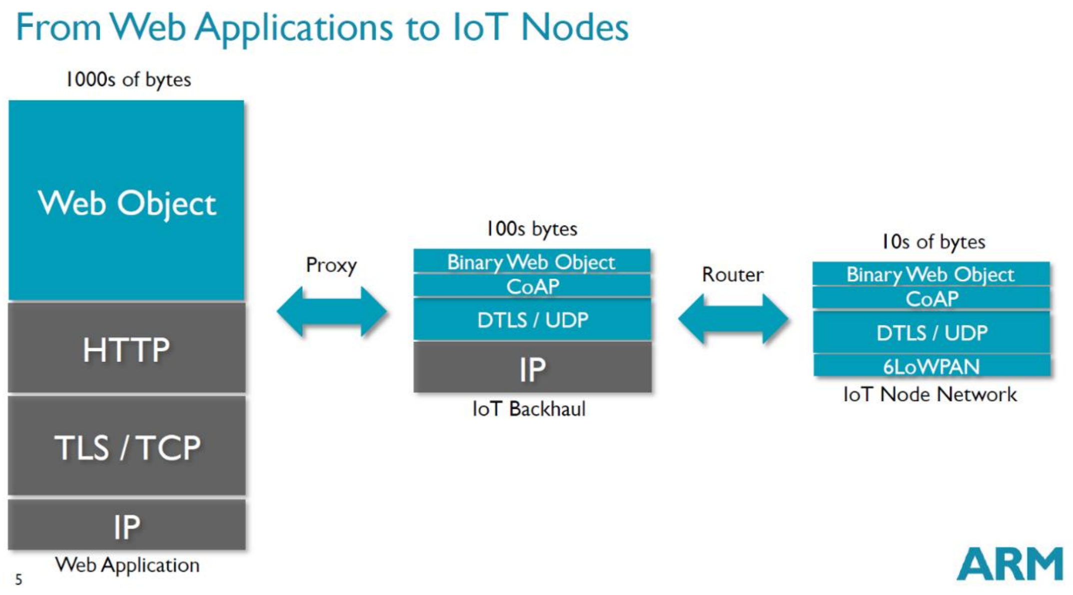


Figure 3.5 From Web Applications to IoT Nodes (Shelby, 2014)

The chart above also shows the current status of IoT network regarding payload size.
Web applications are the backbone of IoT network. The majority of web applications adopt HTTP. In general, the payload size should beyond 1000 bytes since the maximum size of TCP is 65,535bytes and 65,507bytes for UDP.

Between giant network and IoT nodes, there are routers which act as interpreters between CoAP services and HTTP services. As shown above, attributes in CoAP and HTTP are similar, which minimize works in a proxy. On the other hand, since CoAP is designed for lightweight communication, it can be transferred in channels with lower bandwidth (less than 1000 bytes). Further, at the far end nodes of IoT networks, there are many tiny devices with low bandwidth networks, for example, sensors in a room, iBeacon in a small store, and remote start components in cars.

In conclusion, CoAP provides a solution for merging low payload network into the existing HTTP-based network. Apparently, it accelerates the communication between personal networks and the cloud.

3.4.3 COAP VS MQTT

Although both CoAP and MQTT can be used as IoT protocols, they focus on different aspect of data transfer.

MQTT, it applies the concept of “client and broker” to replace the concept of “server and client”. Since the central hub (broker) is used, it is good for multiple communications between smart machines. Meanwhile, with smaller header, MQTT is more efficient than CoAP in transferring small payload. On the other hand, since it is based on subscription and push model, it expects little data in the payload. In addition, the implementation of “last will statement” makes the protocol with better performance when running in intermittent connectivity. Further, the support of three level QoS makes it a clear solution for reliable communication.

CoAP is designed for HTTP-like communication. A CoAP message can easily be transferred into a HTTP message. One the other hand, the minimum size of a CoAP message is 4-bytes which makes it suitable for low bandwidth environment. CoAP does not have a build-in standard to provide quality service, but gives options like “if-match,” “if-none-match” and “accept”, which can be used to implement developers’’ own strategies for quality service. Similarly, regarding data access mechanism, as mentioned above, the last will statement is implemented in MQTT. However, in CoAP, developers have to take advantages of supported “Etag” and “Max-Age” options to implement their own solutions.

In conclusion, the MQTT aims to support multiple clients’ communication within a small network. The CoAP focuses on REST communications and the communications between small networks and the Internet. The CoAP is more flexible and closer to HTTP protocol. In addition, it is more friendly to developers. At last, It requires less effort to transfer a HTTP service into a low bandwidth network.

3.4.4 Conclusion

In the above section, I compared two most popular technologies in IoT. According to the analysis, I draw the conclusion that CoAP is closer to the Internet than MQTT. If I plan to design an architecture where messages can easily be transferred into HTTP messages, I should adopt COAP. As mentioned in the problem definition, our goal is to accelerate the process of merging NonIP based devices into IoT. So, I decide to choose CoAP.

Since I decided to use BLE as an example to test the proposed architecture, I need to review associate works of BLE. In the next chapter, the paper will discuss strength and weakness of Bluetooth and Bluetooth Low Energy.

3.5 Bluetooth

Bluetooth is a standard for low bandwidth wireless communication. It is maintained by the Bluetooth Special Interest Group (SIG). The latest version of Bluetooth is 4.2. The BLE has widely been implemented in smart devices like tablet, smartphone, and wearable devices. It is designed for networks with low data payload but needs frequent small data transfer. It is invented by telecom vendor Ericsson in 1994. Originally, Bluetooth was designed as an alternative technology of RS-232 (a serial port standard).

So far, Bluetooth consists of four technologies: Bluetooth, Bluetooth EDR, Bluetooth HS and Bluetooth low energy. Since Wi-Fi takes great advantages in the field of high-speed data transfer, SIG focuses more on lightweight and low energy communications. In version 4.0, the SIG proposed "Bluetooth Low Energy" to take up the market of low energy wireless communication.

In the remaining section of the chapter, I will discuss the differences between class Bluetooth and Bluetooth low energy. Further, I will explore relevant works of BLE.

3.5.1 Classic Bluetooth

As mentioned above, Classic Bluetooth is a reference of Bluetooth 2.1 +EDR/4.0. It is designed for streaming data transfer. Its data rate can reach 3Mbps. It adopts Standard Bluetooth Profiles (SPP, DUN and PAN) where one master device can have up to seven slaves. Communication through Classic Bluetooth is based on data streaming. In this way, the Classic Bluetooth has better data rate but less energy efficiency.

As shown in the following figure (SIG, 2010), in a BR/EDR piconet, two or more devices occupy the same physical channel. Messages in a physical channel are synchronized by a common clock and hopping sequence. A Bluetooth can not be a master of more than one piconet, but it may belong to two or more piconets. In the following figure, the device A is a master of a piconet with device B, C, D and E as slaves. Meanwhile, the device D is a master of another piconet with J as the slave. The device E plays a role of slave in both the piconet of A and F. The K is an isolated advertising node.

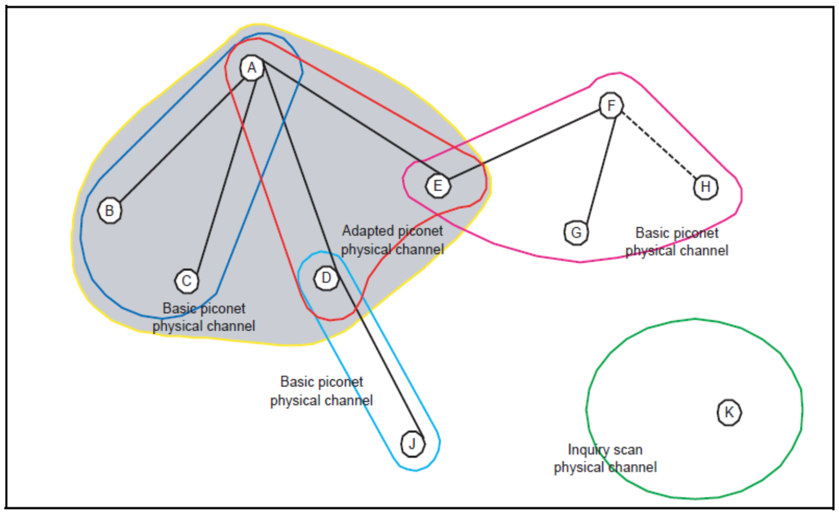

A
B
C
D
E
F
G
H
J
K
Adapted piconet physical channel
Basic piconet physical channel
Basic piconet physical channel
Basic piconet physical channel
Inquiry scan physical channel

Figure 3.6 Topology of Classic Bluetooth (SIG, 2010)

As mentioned above, the classic Bluetooth has widely been adopted in wireless communications. However, it has a natural disadvantage. Since wireless communications are built-in fragile, the streaming data can be interrupted at any time. Moreover, in a single piconet network, one master device can only be connected to up to seven slave devices. Those limitations have impeded the development of it. On the other hand, in 1997, Wi-Fi is introduced. And, the data rate of Wi-Fi reached 11Mbit/s quickly (in 1999). With the development of Wi-Fi, it gradually overlaps the use cases of Bluetooth. After the attempt of increasing data rate in version 3.0, the SIG turns to develop low-energy version of Bluetooth.

In conclusion, the class Bluetooth is a success product. However, it can not make a breakthrough in data transfer rate and facing the challenges of Wi-Fi. The future of Bluetooth has turned out to focus on support smart devices by adopting BLE. In the following section, we will discuss BLE.

3.5.2 Bluetooth Low Energy

The concept of Bluetooth Smart is introduced since Bluetooth 4.0.

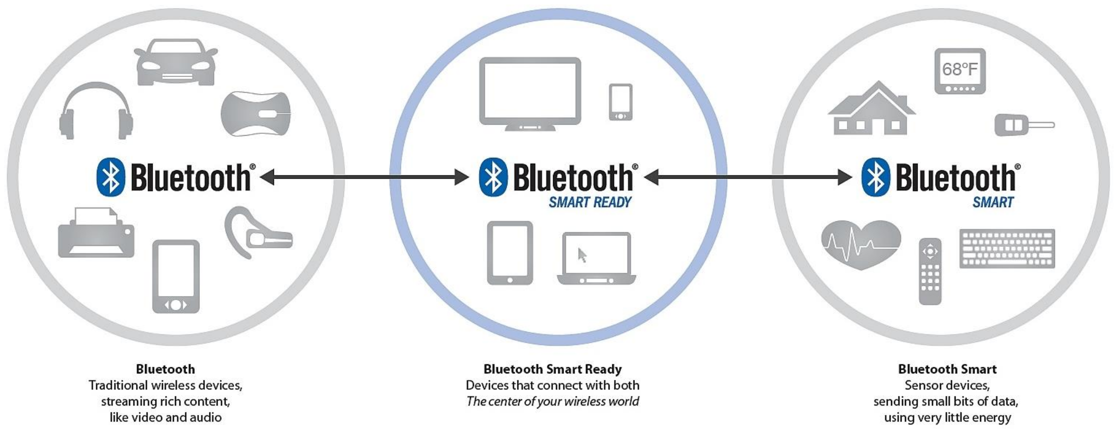

Bluetooth
Bluetooth SMART READY
Bluetooth SMART
68°F
Bluetooth
Traditional wireless devices, streaming rich content, like video and audio
Bluetooth Smart Ready
Devices that connect with both
The center of your wireless world
Bluetooth Smart
Sensor devices, sending small bits of data, using very little energy

Figure 3.7 Topology of Classic Bluetooth (Torvmark, 2013)

As shown above, so far, there are three kinds of Bluetooth devices ("Bluetooth", "Bluetooth Smart Ready" and "Bluetooth Smart"). Devices with "Bluetooth" logo only supports Classic Bluetooth connection. If a Bluetooth device can communicate with both Classic Bluetooth devices and BLE device, it is named as " Bluetooth Smart Ready ". At last, those devices that only support BLE are called "Bluetooth Smart".

According to the official website of Bluetooth, the new low energy technology of Bluetooth supports short data packages with the speed of 1Mbps. In addition, it supports fast transactions as short as 3ms. Moreover, its propagating range can extend as far as 100 meters.

As shown below, unlike "Classic Bluetooth, communication between BLE devices is based on GATT (Generic Attribute Profile) and ATT (Attribute Protocol)". According to Bluetooth4.0 specification (SIG, 2010), "The GATT server sends responses to requests and when configured, sends indication and notifications asynchronously to the GATT client when specified events occur on the GATT server.". As shown in the following figure, a GATT profile may contain one or more services. Each service acts as a folder to contain a set of characteristics to store data. For a characteristic, there are three types of attributes: “Properties”, “Value” and “Descriptor”. The BLE client can read, write or monitor the “Value”.

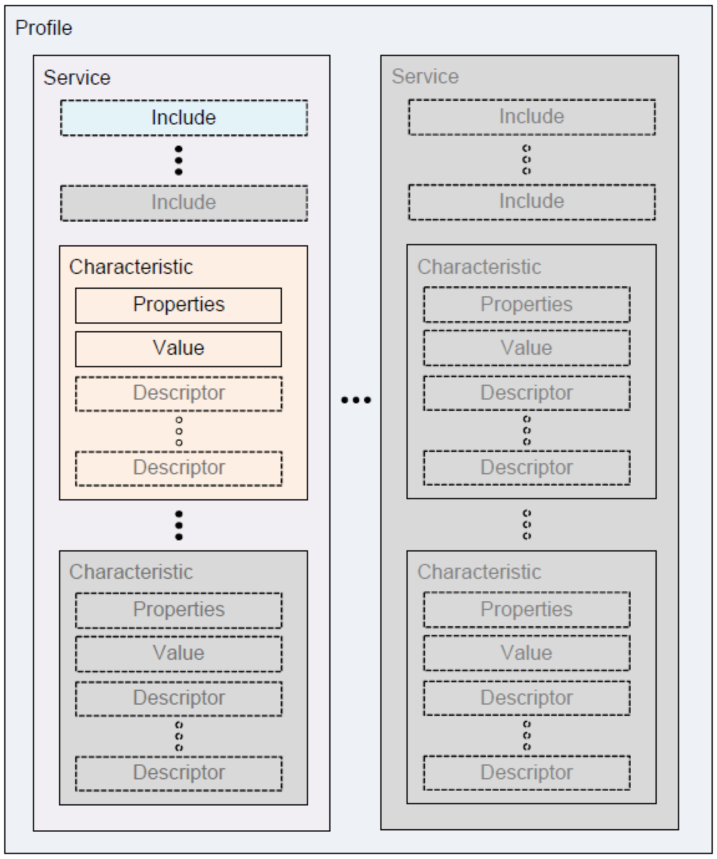


Figure 3.8 GATT format (SIG, 2010)

As shown below, there is a great difference between Classic Bluetooth and Bluetooth Low Energy in terms of topology. In BLE, one physical channel consists of two devices, which means each slave communicates with a master in a separate physical channel.

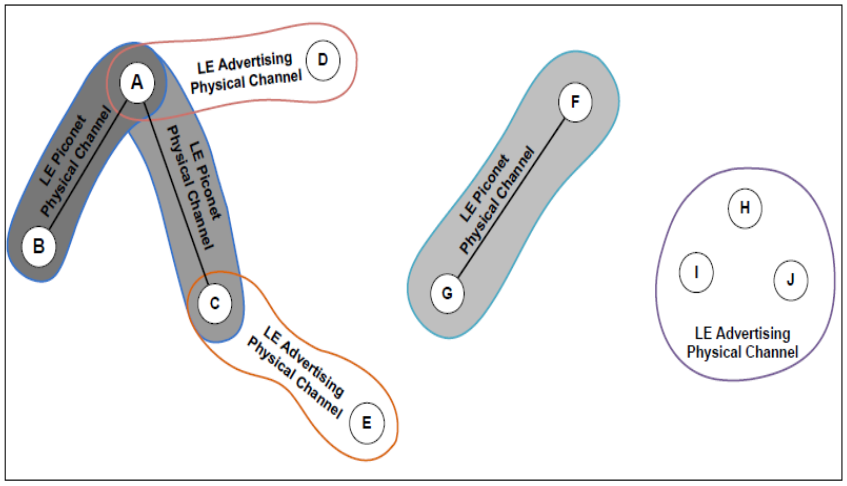

LE Advertising Physical Channel
D
A
LE Piconet Physical Channel
B
LE Piconet Physical Channel
C
LE Advertising Physical Channel
E
F
LE Piconet Physical Channel
G
H
I
J
LE Advertising Physical Channel

Figure 3.9 GATT format (SIG, 2010)

So far, there are two more versions available for BLE. The first update is available since 2013 (called Bluetooth 4.1). The second update is available since 2014 (called Bluetooth 4.2).

Bluetooth 4.1. There are four main changes in version 4.1 (SIG, 2013):

1. Solve interference: Bluetooth and LTE interfered with each other. In this version, it tries to prevent the interference between them.
2. Flexible Connections: Bluetooth 4.1 allows manufacturers to customize reconnection timeout intervals, which helps to reduce power consumption.
3. Multiple roles: Devices can act as hubs and end points at the same time.

According to Suke Jawanda who is the chief marketing officer of Bluetooth SIG (SIG, 2013), “We updated the Bluetooth specification to address this projected growth, making changes to give developers more control in assigning a role to their product, limiting interference with other wireless technologies, and allowing Bluetooth Smart products to exchange data faster and maintain connections with less manual intervention.”.

Bluetooth 4.2. The latest version of Bluetooth is 4.2. According to FAQ document of Bluetooth (SIG, 2014), the latest version has improved BLE in three aspects: IoT capability, security, and speed.

Regarding IoT capability, a BLE device can directly participate in IoT network by adopting 6LowPAN and connect with routers supporting Bluetooth Smart Internet Gateway.

Regarding security, LE Privacy 1.2 was introduced to prevent Bluetooth smart devices being tracked by untrusted devices. Moreover, it uses FIPS-compliant encryption to secure data transfer.

Regarding speed, the SIG claims the new patch will make BLE 2.5 times faster and capacity of packet will be 10 times larger than previous versions.

iBeacon. iBeacon is a protocol proposed by Apple. It defines the usage of BLE in goods' information transfer. According to BLE's advertising standard, users can set up to 20-bytes payload. Apple proposed the protocol to format advertising data (iOS, 2014). The 20-bytes are divided into three parts: 16-bytes UUID, 2-bytes major value, and 2-bytes minor value. The protocol is designed for business owners who want to push ids of their products to people who are nearby around the store. The technology takes the advantages of BLE's advertising mechanism which can guarantee that those 20 bytes can always be visited by BLE smart ready devices.

Taking advantages of BLE's short radiation range, the iBeacon can push data notification to a BLE smart ready device. The iBeacon aims to use BLE as a tag of real products.

6LowPAN. 6LowPAN is short for IPv6 over Low-Power Wireless Personal Area Networks. So far, the latest version of BLE (SIG, 2014) has proposed 6LoWPAN implementation and Bluetooth Smart Internet Gateways to support IPv6 based communication. This means a remote device can directly control a BLE sensor by IPv6.

3.5.3 Classic Bluetooth vs Bluetooth Low Energy

Based on above introduction, we can draw a conclusion that Classic Bluetooth and Bluetooth Low Energy target different markets. For Classic Bluetooth, it is a desirable solution where higher data rate is required and more power is available. For Bluetooth Low Energy, it is designed for devices with less power than Classic Bluetooth devices. BLE is a better option where small amount of data need to be transferred frequently. Also, the BLE has some features that make it different from other short-range wireless technologies.

3.5.4 Conclusion

So far, the BLE is widely adopted in smart devices, which has brought it to an important position in the market. With the adoption of IPv6, it can join the IoT with little effort. Meanwhile, with more and more extra protocols are developed for it (e.g. iBeacon). The ecosystem of BLE will become more and more robust. The latest changes for BLE are IPv6's capability, packet size and security level. Those improvements meet the evolution of IoT. Bluetooth needs to become more compatible with existing network and can transfer more data in a more secure way. With the development of IoT. It is expected that both the density of sensors and the computing power will increase.

So far, there are two innovation directions for BLE devices:

The first direction goes toward cheaper and simpler. This kind of sensor aims to collect limited amount of data. For example, in a large farm land, the farmer needs to manage temperature and humidity in different spot. In this scenario, many sensors will involve in the network. They will constantly report two types of data to the local hub. In this case, each sensor needs to be as cheap as possible.

The other direction is to integrating sensors in smart devices, where different kinds of data are required to be processed at the same time. Therefore, this type of device is constrained by both battery and bandwidth. The recent updates of BLE improved its performance. Now, developers are more comfortable to develop medium size apps based on integrated sensors.

In the next section, I will discuss the CAP theorem which becomes a principle guiding the design and implementation of the proposed architecture.

3.6 CAP Theorem

The CAP stands for Consistency, Availability, and Partition-tolerance. It is an important concept in any design of a distributed architecture. It was proposed by Brewer in 2000. In 2002, Seth Gilbert and Nancy Lynch proved (Gilbert, S., Lynch, N., 2002) the theorem and pointed out that "It is impossible to achieve all three.". The following are details of those three concepts:

- Consistency: Consistency means operations can only be fully executed or dropped.
- Availability: Availability means a user can be served at any time. For example, a high availability system can be accessed by users at any time even when it is updating or maintaining.
- Partition Tolerance: A system with high partition tolerance can still serve users even if connections between two nodes are lost.

Since the intermittent connectivity is a given condition, mobile applications must achieve high partition tolerance. Therefore, the options for a mobile application are limited between PC (Partition Tolerance and Consistency) and PA (Partition Tolerance and Availability).

In the context of PA (Partition Tolerance and availability), the system and database are separated into different nodes to guarantee both Partition Tolerance and availability. Since the whole data assets are available for each node, a user can access any data when one or more nodes are not available. However, in this scenario, consistency cannot be guaranteed since changes made by the user need to synchronize with the lost node only when the network is available.

In the context of PC (Partition Tolerance and Consistency), if a system's Consistency is required while the system needs partition tolerance, the system can not have high availability. When a node lost connection, other nodes can change data without worrying about the issue of inconsistency. However, in this scenario, users can not access data in the lost node.

The proposed architecture aims to guarantee high availability. Even if a connection between devices are not available, a request should get a response. Therefore, the system must choose PA as its design principle.

CHAPTER 4 DESIGN AND ARCHITECTURE

With the development of smart devices, the fragile network of sensors becomes more and more reliable. Under this context, we expect the CoAP should be supported in those networks. In this research, our main goal is to propose a suitable solution to support CoAP in Non-IP based WPAN.

Currently, implementations of CoAP only support IP based communication. Technologies like BLE (below v4.2) and NFC do not support IP and have their own standards to transfer data. However, with the development of WPAN and CoAP, it is time to think about the possibility to grant CoAP in WPAN. In addition, if CoAP is adopted as a general protocol in WPAN, it can simply the data communication between WPAN and existing networks.

In the remaining section of this chapter, we will introduce our solution: CoAPNonIP architecture.

As shown below, the proposed CoAPNonIP architecture consists of an application layer and a network layer. The application layer focuses on message management and message delivery.

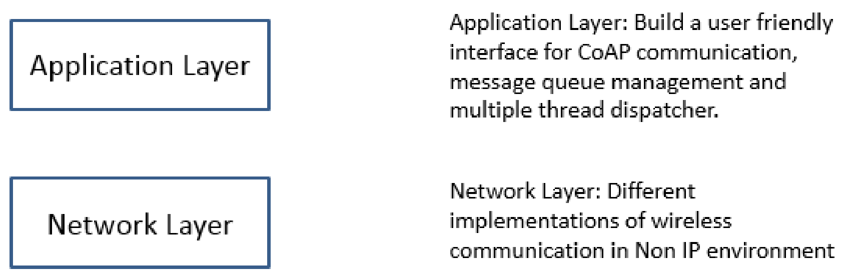

Application Layer
Application Layer: Build a user friendly interface for CoAP communication, message queue management and multiple thread dispatcher.
Network Layer
Network Layer: Different implementations of wireless communication in Non IP environment

Figure 4.1 Layers of proposed architecture