Web of Things Architecture

1. Introduction

This document is an explanation about the architecture of “Web Of Things (WoT)”.

The purpose of this document is to provide

(a) a guideline of the mapping between functional architecture and physical devices configuration,

(b) a description of the role and functionality of each logical module,

(c) a reference for where should be standardized.

2. Requirements for functional architecture of WoT

2.1 Flexibility

There is a wide variety of Physical devices configuration for WoT implementations.
Functional WoT architecture should be able to be mapped to and cover all of the variations.

2.2 Compatibility

We have already had many legacy IoT solutions and IoT standardization activities in many business fields.
Functional WoT architecture should provide bridge between legacy IoT solutions and Web technology based on WoT concepts. And it should guarantee to be upper compatible to legacy IoT solutions and current standards.

2.3 Safety and Security

Functional WoT architecture should have the room for providing safety and security functionalities.
In the IoT solutions, once cyber security barrier is hacked, it is more easily led to safety issues than conventional web solutions. That is because hacked IoT devices often treat

heal cycle such as central heating systems, physical moving devices such as cars.

3. Mapping variations

In real world, there are many variations for mapping logical WoT servient to physical devices structure.
This chapter tries to list up informative mapping samples.

3.1 Simple Use case

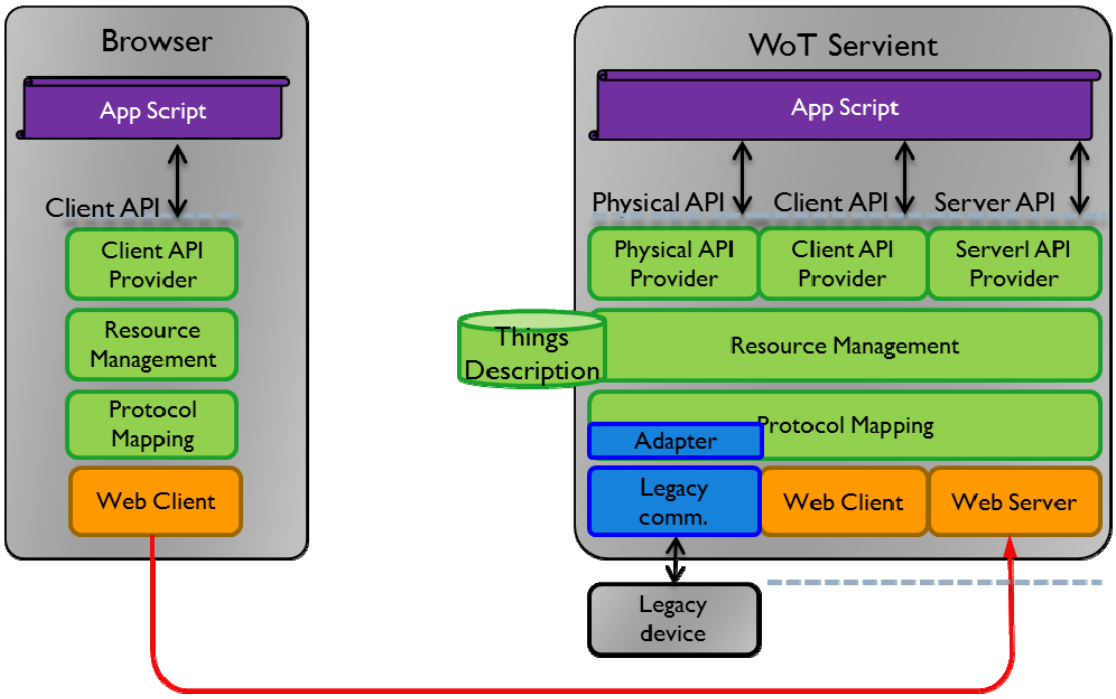

Fig.2 Simple use case

Fig.2 shows very simple use case such that a browser accessed WoT Servient to get some information of legacy device and/or put some parameters to control legacy device.

In this use case, browser's App Script refers Things Description of WoT servient and get information of who it is, what kind of APIs it provides.
Then App Script calls client API and through resource management processes and protocol mapping process, the request is mapped on internet protocol such as HTTP, CoAP and so on.
The protocol accesses Web Server block of WoT servient. After that the request is transferred to App Script in WoT servient through protocol mapping process and

resource management process.
App Script understands what kind of request comes from browser and according to the request, App Script controls legacy device through physical APIs.

3.2 WoT servient on device

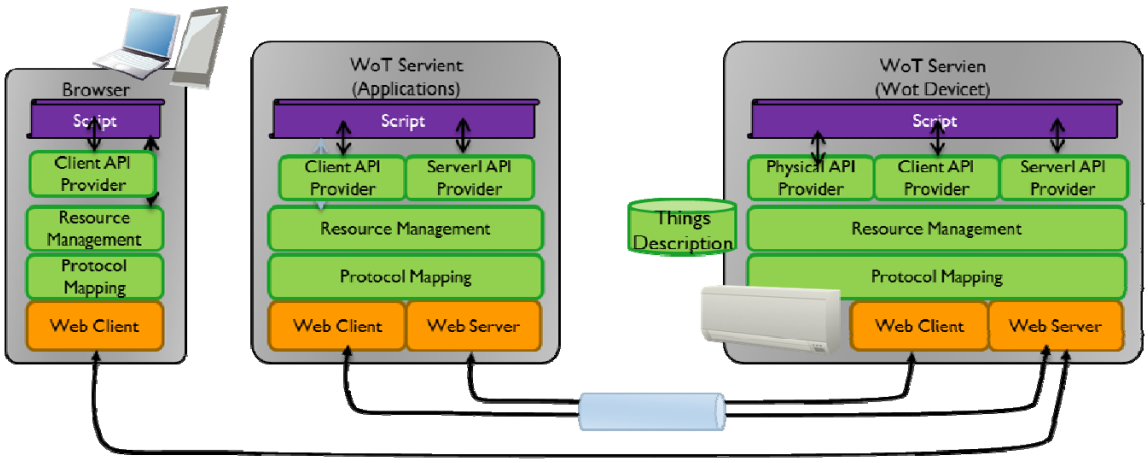

Fig.3 WoT Servient on device itself

The first example is WoT servient on device itself. This is referred as “WoT Device”.
The right most WoT servient in Fig.3 shows an air conditioner which has rich CPU and large memory and provides web server functionality connected directly to internet.

Then the leftmost browser and/or another application on internet can access the air conditioner through internet directly.

3.3 WoT servient on Smartphone

The second example is WoT servient on Smartphone.
Smartphone becomes very popular and it provides gateway functionality which bridges between internet and legacy device without any intermediate hardware.

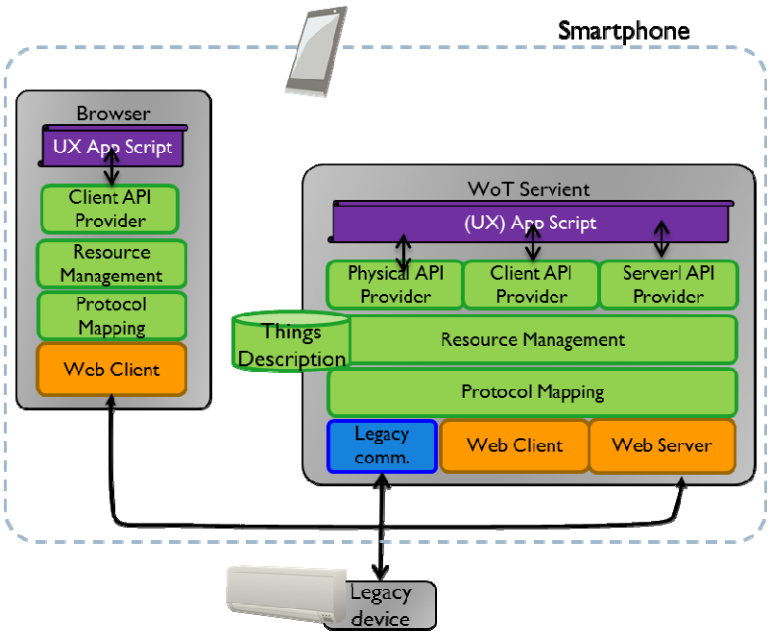

Fig.4 WoT servient on Smartphone (A)

Fig.4 shows an example of WoT servient on Smartphone. In Fig.4, there are independent 2 software modules, one is browser which has user experience to provide interaction, the the other is WoT servient which might not have any user interface and it provides gateway functionality to access legacy device.

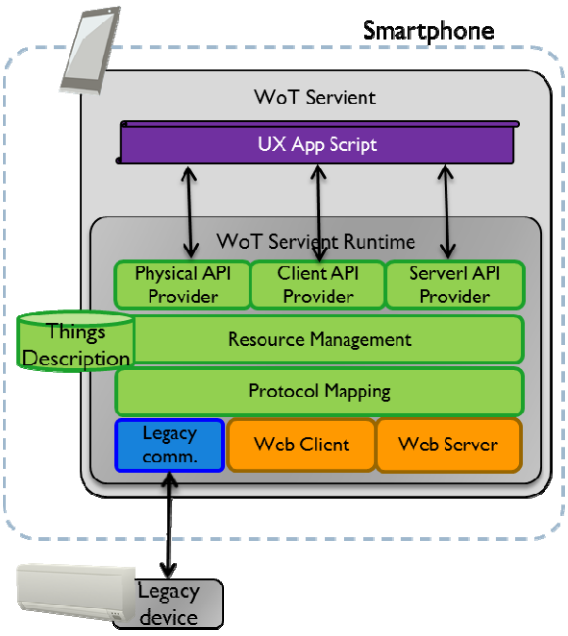

Fig.5 WoT servient on Smartphone (B)

Fig.5 shows another example of WoT servient mapped on smartphone.
In this mapping case, a browser is expanded to include WoT servient functionality. Then there is no need for an app script to call web server block. Instead the client API should be called directly inside.

3.4 WoT servient on Smart Home Hub

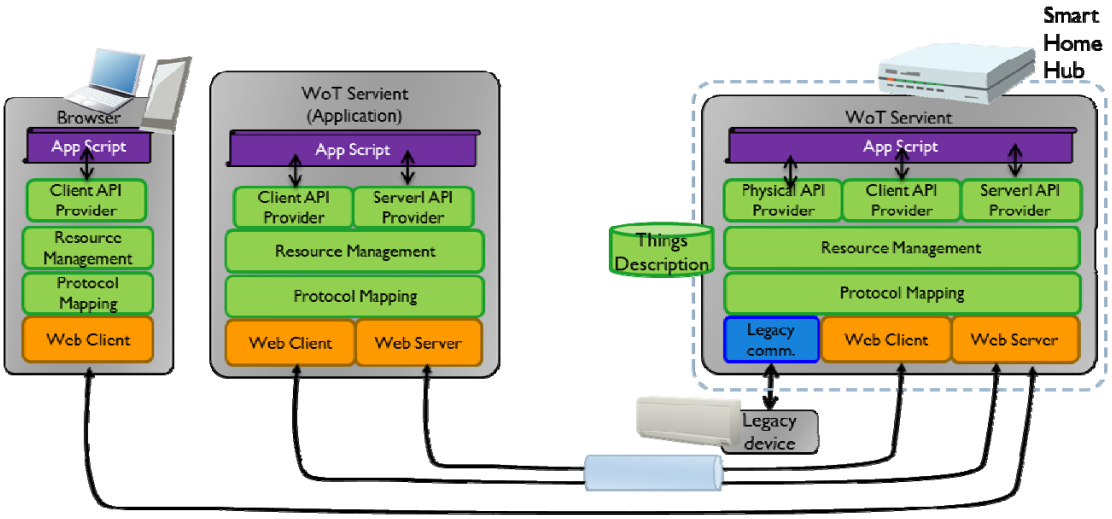

Fig.6 WoT servient on smart home hub

Fig.6 shows an example of WoT servient on smart home hub.
Smart home hub is usually introduced home automation and/or home energy management solution.
Looking at consumer electronics, there are very wide variety of physical communication format such as WiFi, 802.15.4g, Bluetooth low energy, HDPLC and so on. In order to normalize those variations, almost all home systems introduce a smart home hub.
In Fig.6, WoT servient wraps the difference of communicating legacy devices and provides to other clients a universal devices accessing method.
In home inside, as the communication method between WoT servient on smart home hub and clients WiFi is usually adopted.

5.4 WoT servient on Cloud Server

Client Apps can control devices at home through WoT servient on a Smart Home Hub. But the location of client Apps is restricted within home because physical

communication path “WiFi” and/or wired Ethernet between smart home hub and client apps such as browser is limited inside home.

So, controlling devices at home from outside the house, WoT servient from a smart home hub should be mapped to a globally accessible cloud.

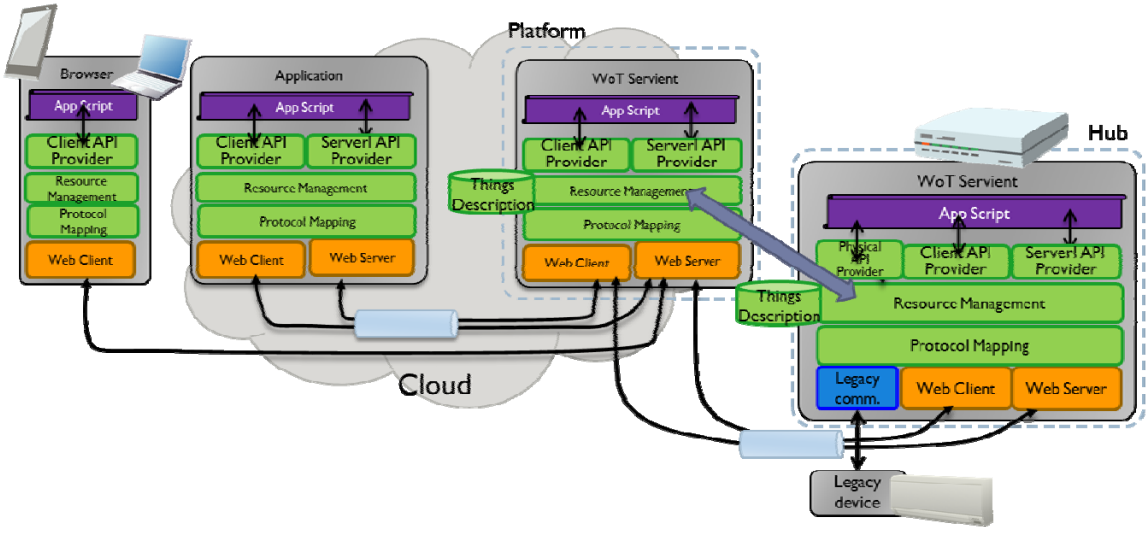

Fig.7 WoT Servient on Cloud Server

Fig.7 shows an example of WoT servient on a cloud server.
In Fig.7 case, a browser accesses WoT servient on the cloud named “platform”. This WoT servient provides Things Description through internet globally. So, wherever browser user is, he/she can access this WoT servient.
WoT servient accepts browser and/or other application’s request through HTTP, CoAP and so on. Then WoT Servient on the cloud server finds out the route to access the WoT servient on a smart home hub. In Fig.7 case, Things Description of WoT Servient on cloud server is mirror of that on the smart home hub.
After finding out the route, WoT Servient on the cloud server transfer browser’s request to WoT Servient on the smart home hub.
After that, the smart home hub processes the request according to Fig.6 case.

In this Fig.7 case, the smart home hub works as

a) Unifier of very wide variety of legacy communication protocols both in the physical and logical view;
b) Firewall between internet as WoT Servient on the cloud server and legacy connected devices at home;

c) Privacy filter which substitutes real image and/or speech, and logs data at home to symbols;
d) Autonomous WoT Servient which provides house inside the server, even if the connection is shut down between internet and the smart home hub;
e) Emergency Apps running in a local environment when the fire alarm and similar event occur.

4. General Description of WoT Servient

In Web of Things (WoT), functional virtual device is named "WoT Servient" which provides the access to, control and get the status and values from IoT physical devices.

General WoT Servient functional architecture is presented in Fig.1

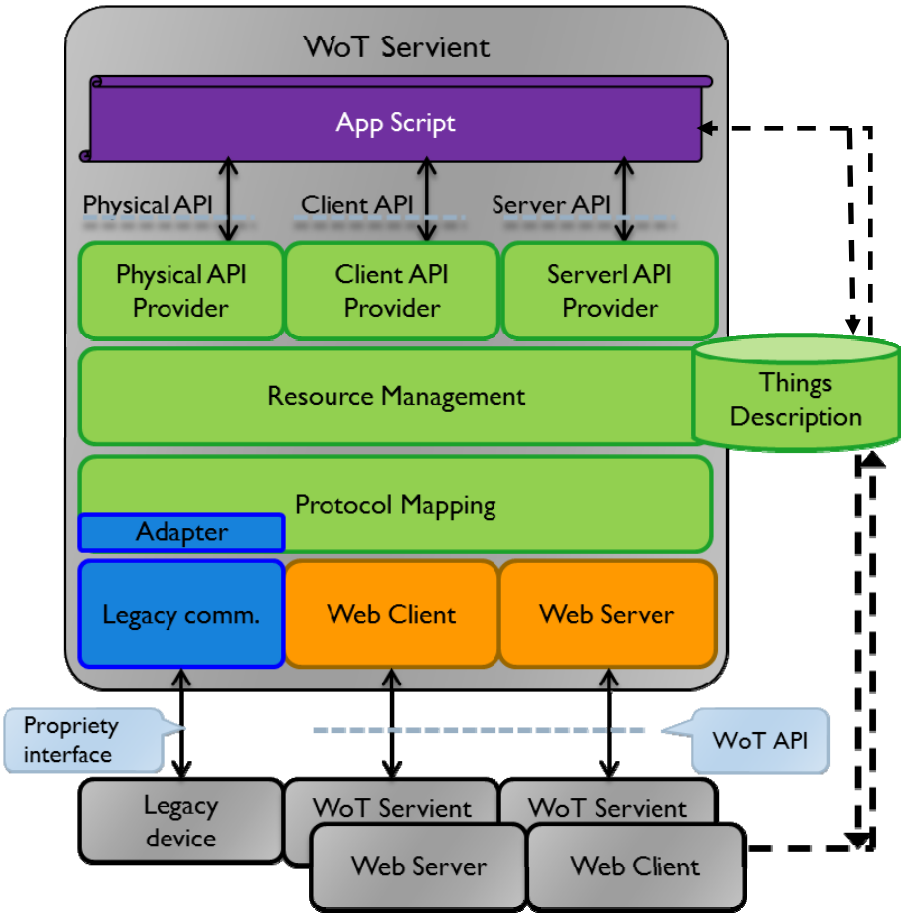

Fig.1 Functional Architecture of WoT Servient

The role and functionality of each module is as follows;

Web Server

Web server accepts requests from networked clients through internet and sends responses to clients.
Examples of protocols between WoT servient and clients include HTTP, CoAP, MQTT and so on.

REST style API can be defined in front of the Web Server module. This API is named "WoT API". WoT API is the subject of the standardization activity. .

Web Client

WoT Servient can access other web servers and/or WoT servients through internet.
In these cases a web client module communicates with other servers via protocols such as HTTP.

When web client modules calls other WoT servients not legacy web servers, the API is WoT API.

Legacy Communication

As described before, currently, there are many IoT services and standards proposed by many organizations.
In order to communicate such legacy devices, WoT servient includes legacy communication module for such protocols as Echonet Lite, QNX, ONVIF, DLNA and so on..

Protocol Mapping

Protocol Mapping block maps API and parameters to communication protocols such as HTTP, CoAP, MQTT and so on.

And in Protocol Mapping block, there is a block named **Adapter** which translates web

based API to legacy communication such as Echonet Lite, QNX, ONVIF, DLNA and so on.

Resource Manager

WoT servient manages many resources.
As server, WoT servient manages clients' request queue for serialization of controlling thing and so on. Provided APIs (resources) are described in Thing Description.

As client, WoT servient caches requesting servers information to accelerate next requesting process.

As things controller, WoT servient manages the inside status of thing and so on.

Whole of those management block is named **Resource Management**.

Things Description

Things Description is a declaration of properties and capabilities of a thing or a legacy device. Examples include the name of thing, APIs (resources) which is provided by WoT servient, parameter data type of each APIs and so on.

Other clients refer WoT servient's "Things Description" to understand the ability of the WoT Servient.

Server API Provider and **Server API**

Sever API is the API for creating server functions. Server API Provider provides utilizes resource management, protocol mapping and son and provides Server APIs.

Client API Provider and **Client API**

Client API is the API for creating client functions. Client API Provider provides utilizes resource management, protocol mapping and son and provides Client APIs.

Physical API Provider and **Physical API**

Physical API is the API for creating physical things controlling functions such as GPIB, I2C and so on. Physical API Provider provides utilizes resource management, protocol mapping and son and provides Physical APIs.

App Script

Calling Server API, Client API and Physical API, application script is created.

5. Conclusion

. A functional architecture for WoT Servient is introduced in Fig.1. This functional architecture can explain well a wide variety of different WoT application scenarios.

As the next stage of the standardization, we should standardize the following 3 items.

1) WoT API mapped to protocols that enable communication between a client and WoT Servient;
2) Things Description to declare properties and capabilities of WoT Servient to the Web;
3) Server API, Client API and Physical API to enable App scripts.