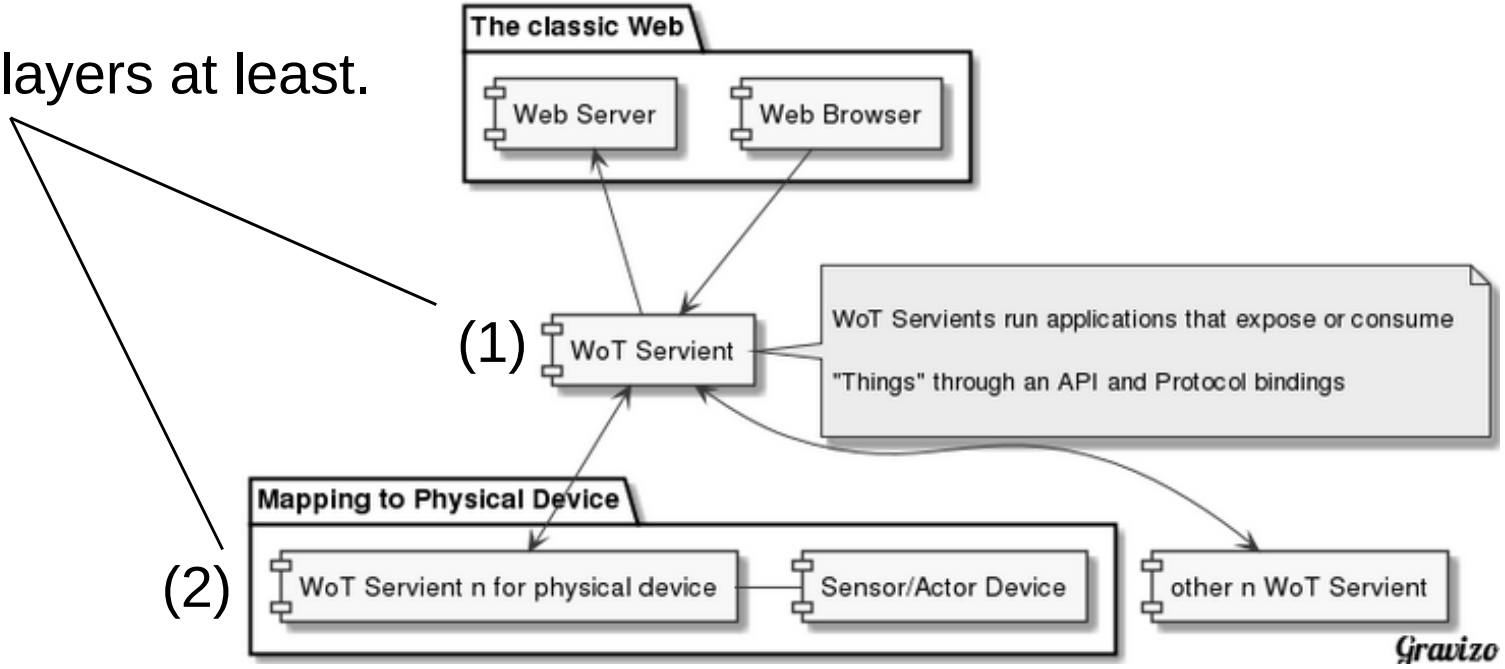
Panasonic

WoT Skeleton Architecture (1)

I agree there should be 2 layers at least.



(2) is easy to understand, “WoT Servient n for physical device” is virtual device n.
This Wot Servient includes

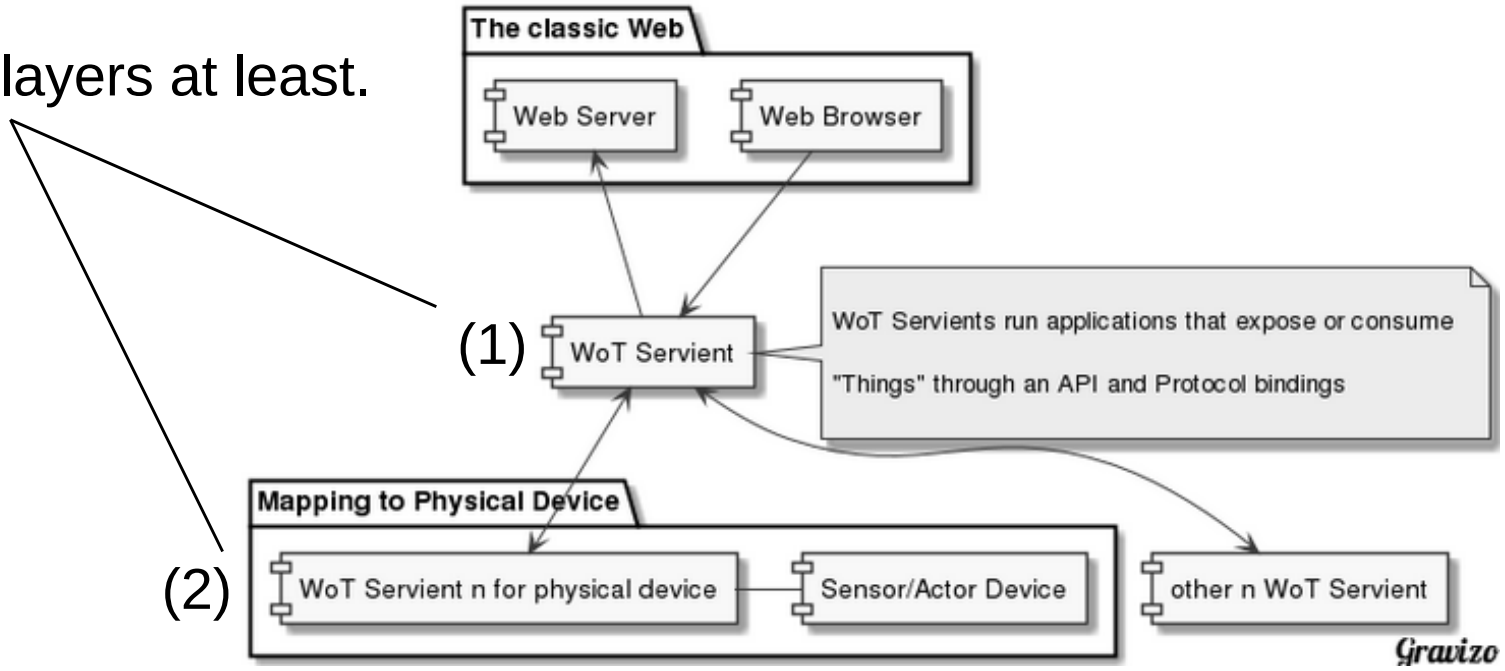
- Thing description
- Access method to Sensor/Actor Device
- Notification* method from Sensor/Actor Device
- REST API to operate a thing and/or a thing description(PUT/GET/Notify(?))

**Sensor status changes not only by upper direction but by environment change and/or local Actor operation, so notification method should be introduced, I think.*

Panasonic

WoT Skeleton Architecture (2)

I agree there should be 2 layers at least.



(1) is required to manage multiple (2) virtual devices.
- Create (2) and register (2) entry into (1)
- Delete (2) entry from (1)
- REST API to manage things and/or things description(PUT/GET/Notify(?))
- if required, Access control is bound to (1)

If there is only one and access free Sensor/Actor Device logically for a service, then (1) might be omitted.

Panasonic

Treatment of Implementation

As described in scope of TF-AP, implementation is treated as “reference” for mapping WoT servient to physical device.

But, during discussion, everyone talks based on each own implementation experience, then the discussion sometimes seems to be confused.

I’d like to propose to try to make it clear that
- listing up implementation models and WoT servient mapping
- confirmation on what kind of I/F our TF-AP should define as API

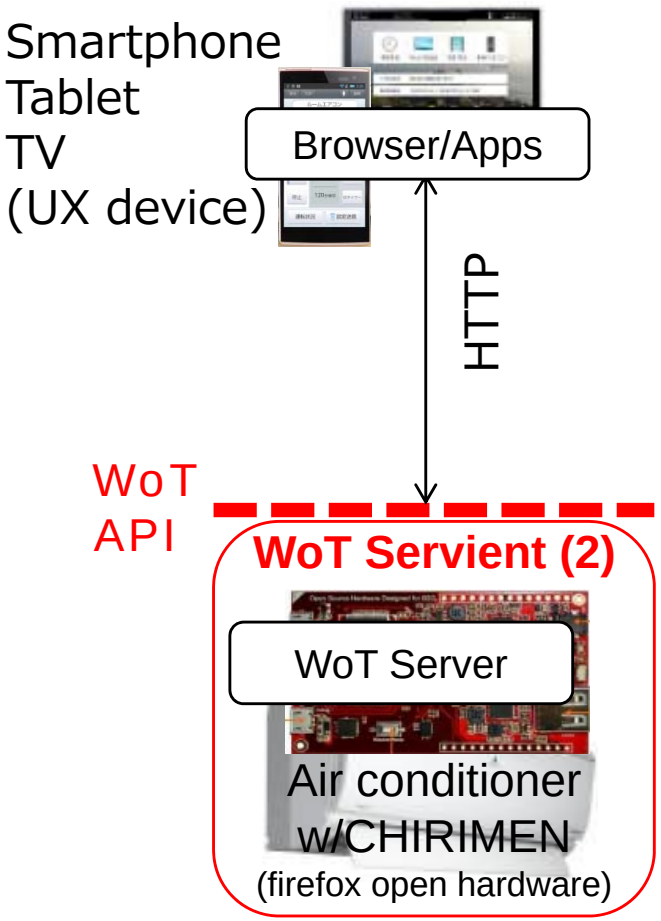
In next 2 slides, I try to figure out 4 different type of implementation of WoT servient(2).
(a) Web centric implementation, that is, a physical device itself is WoT Servient(2).
(b) Smartphone centric implementation, that is, smartphone connects to a physical device and smartphone includes both UX apps and GW.
(c) Hub centric implementation, that is, hub connects to a physical device and hub includes GW and provides REST API to other UX devices.
(d) Cloud centric implementation, that is, hub connects to a physical device and includes GW, cloud binds hub and UX devices and provides REST API.

**GW don’t have to be physical box. GW is the logical module providing following functions.*
- *media conversion (ethernet / WiFi / 802.14.4x / BTLE /…)*
- *protocol conversion (HTTP / CoAP / IEEE1888 / Alljoyn / Echonet Lite / ZigBEE /…)*

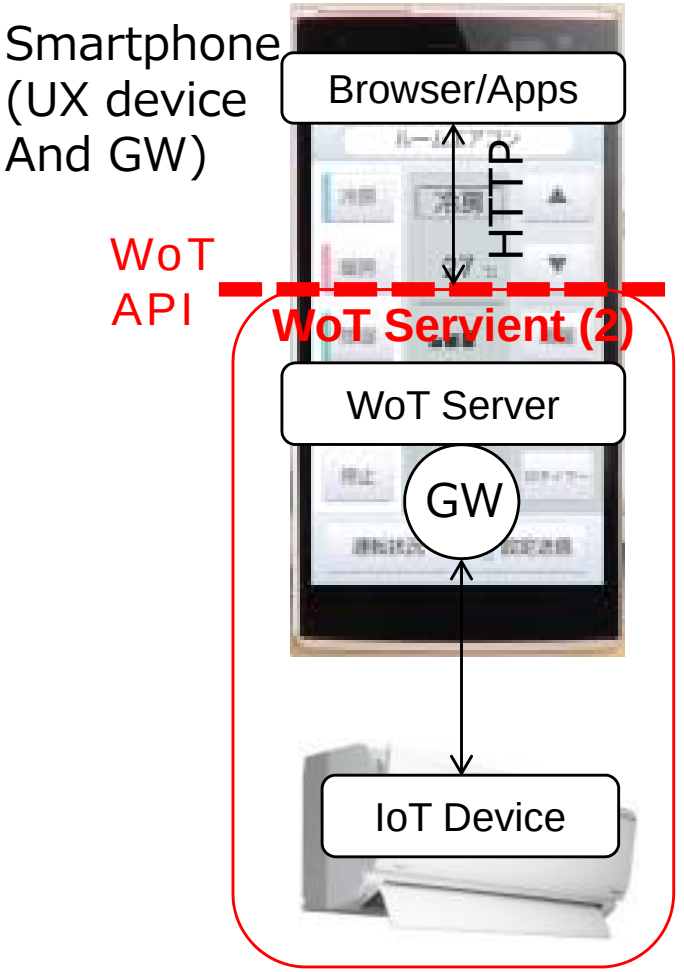
Panasonic

Implementation Model (1)

Web Centric

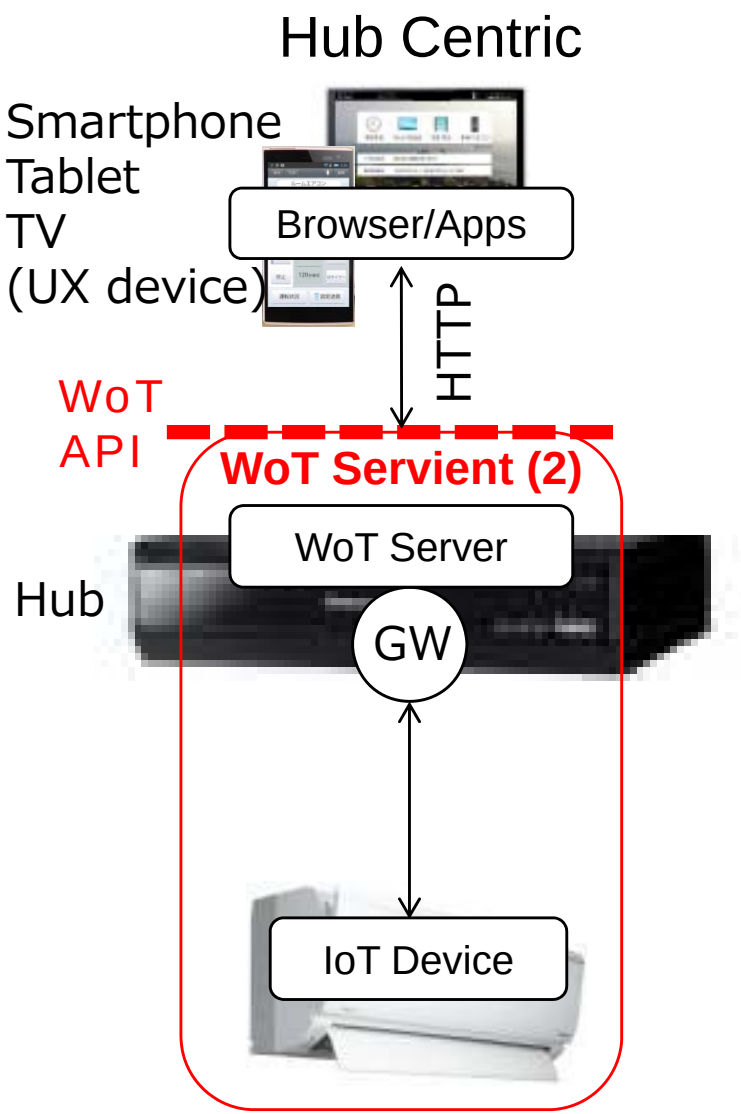


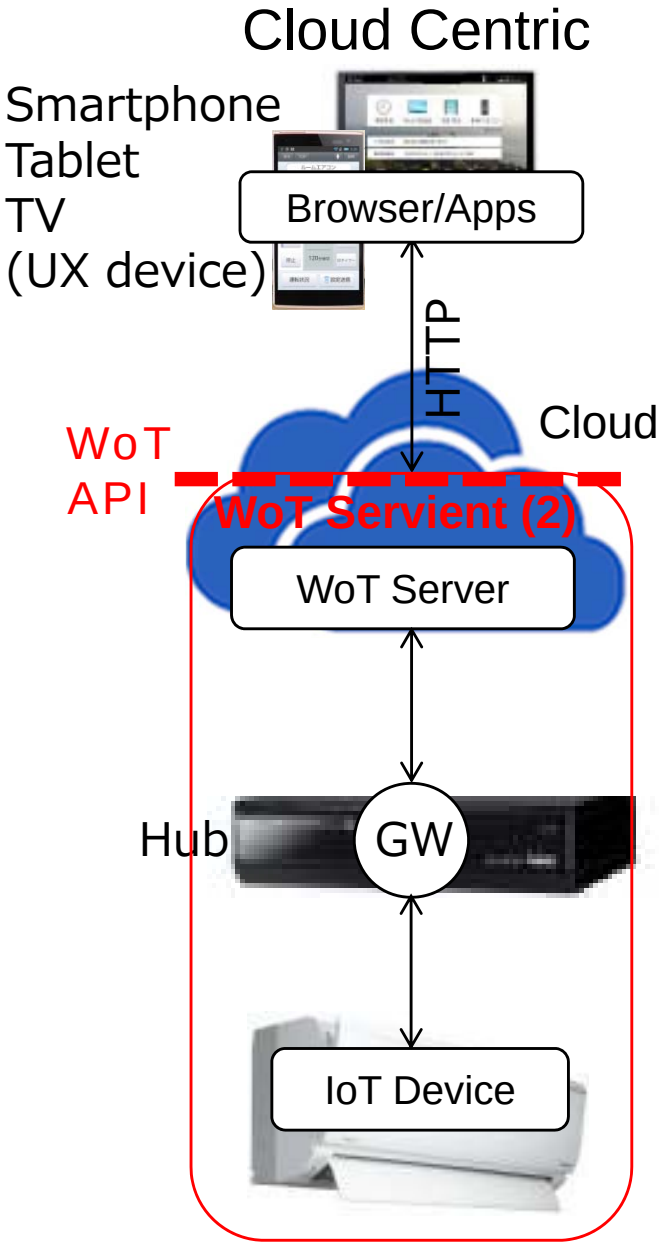
Smartphone Centric



Panasonic

Implementation Model (2)





Implementation models are different, but WoT API can be defined as same API.