SONY

Web of Things

Standardization and example use case "Remote health monitoring"

Presentation at IEC System Committee on Active Assisted Living meeting on 17 November 2015

Claes Nilsson / Sony Mobile – Research & Incubation

IoT market fragmentation

- IoT market is fragmented
  - Multitude of organizations specifying Internet of Things solutions
  - Plethora of communication technologies
  - Lack of a shared approach to services

mDNS

IoT fragmentation is holding back the true potential

"Interoperability among IoT systems is required to capture 40 percent of the potential value"

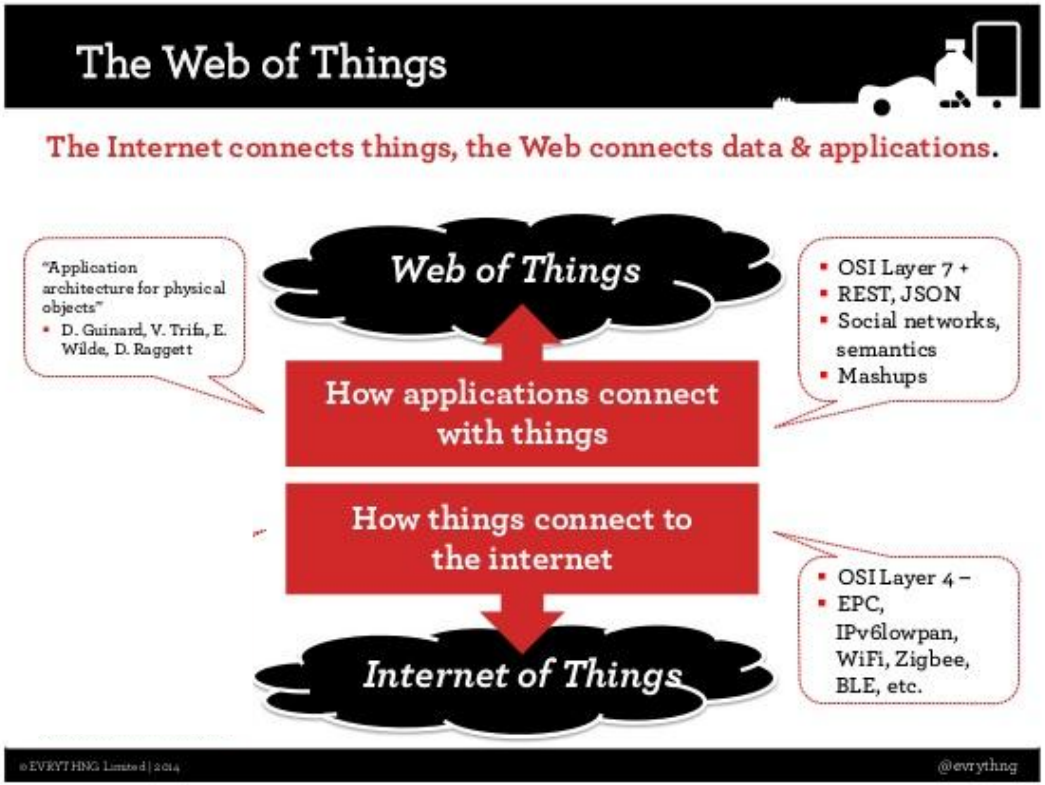
Source: McKinsey THE INTERNET OF THINGS: MAPPING THE VALUE BEYOND THE HYPE

amazon web services™
ARM mbed™
Physical Web
Walk up and use anything
MQTT.ORG
thingsquare
SmartThings
ALLSEEN ALLIANCE
ZigBee Alliance
OPEN INTERCONNECT CONSORTIUM
UPnP™
THREAD
mDNS
EVRYTHNG
ANT+
Bluetooth®

Web of Things

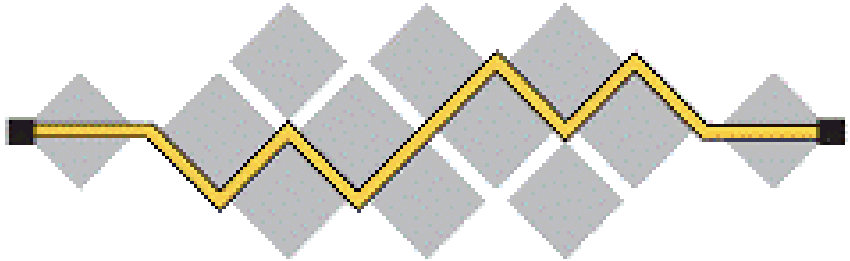
- Web of Things
  - Cross platform application service concept for Internet of Things
  - Re-using existing cloud architecture and Web standards (JavaScript, URLs, HTTP, REST, Web Sockets, WebRTC, JSON, OAuth, etc) to create IoT applications and services

Web of Things versus Internet of Things



Source: Evrythng

SONY

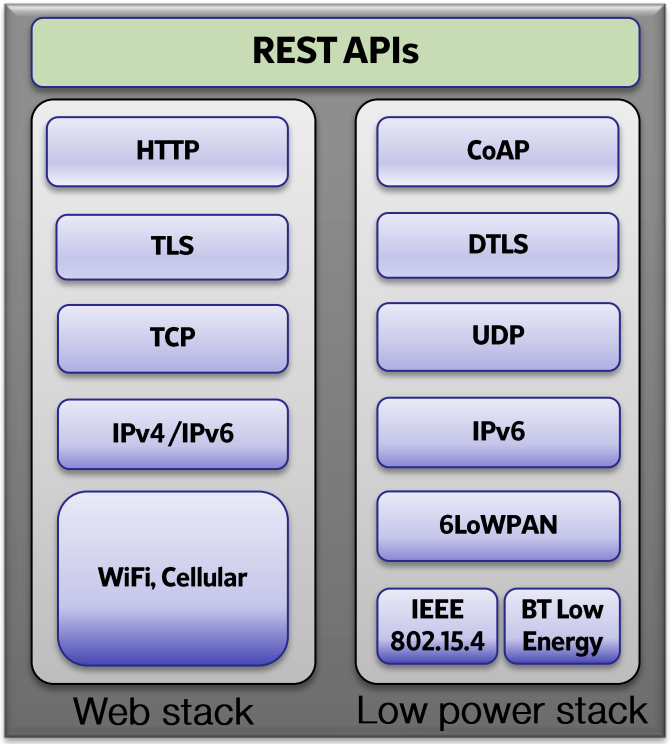


IETF®

SONY
make.believe

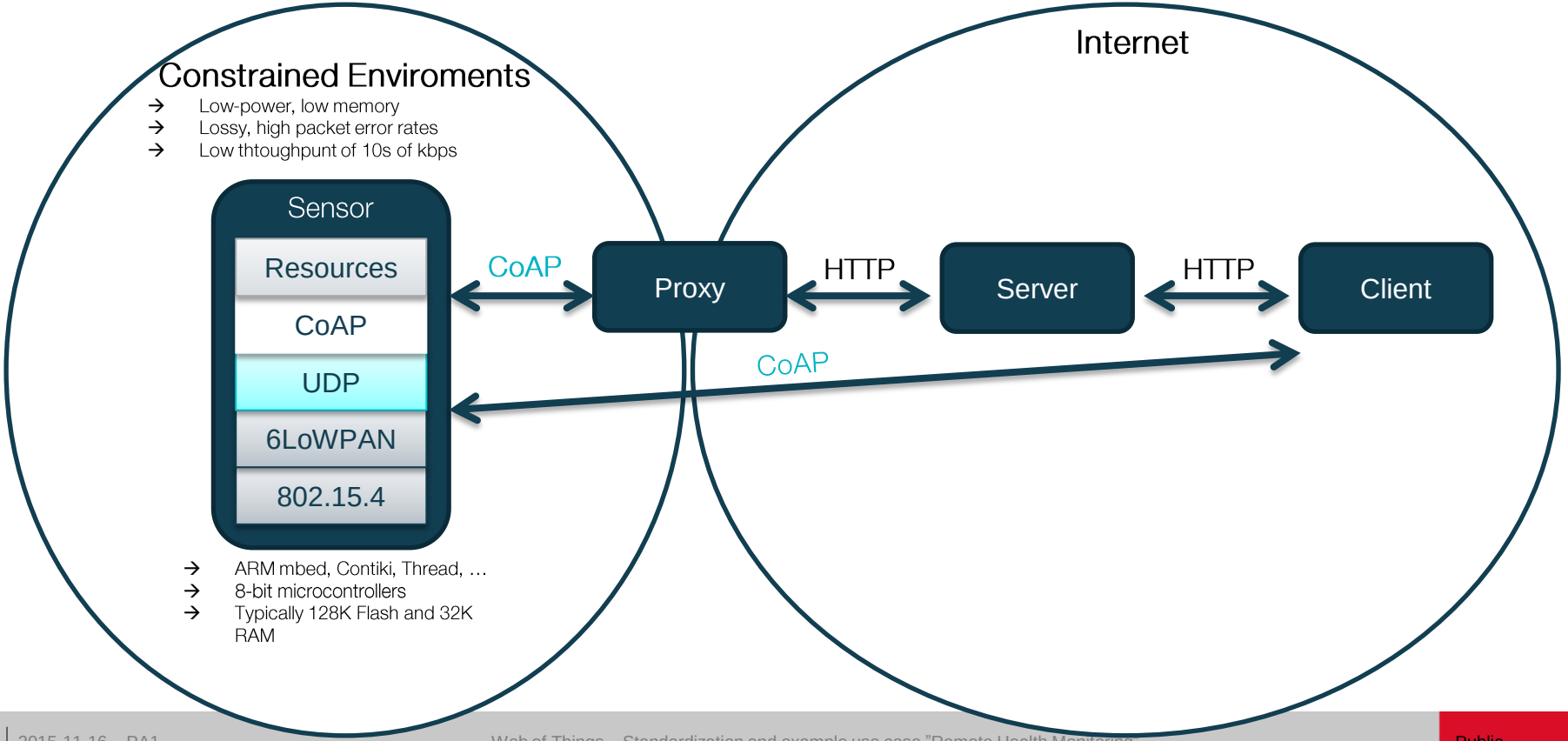
IETF®

- The IETF Constrained RESTful Environments (core) WG has defined a protocol stack for low power, low memory and low cost IoT devices
  - Scalable and distributed architecture
  - Smooth integration with existing cloud services
  - Access control on service layer rather than network layer
  - Supports both mesh and point-to-point networks
  - Examples of implementations:
    - ARM mbed
    - Contiki
    - Thread
  - Also see CoAP Technology



CoAP

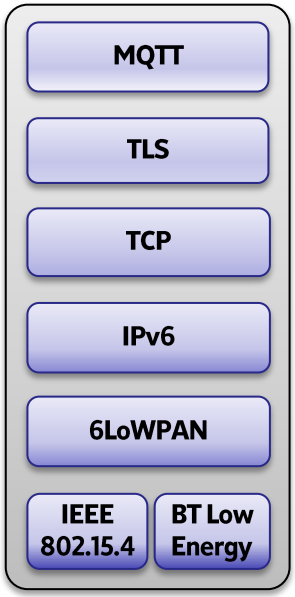
Application layer protocol that enables **web services** for even the most **constrained** devices and networks, while integrating with the **web architecture** and HTTP.

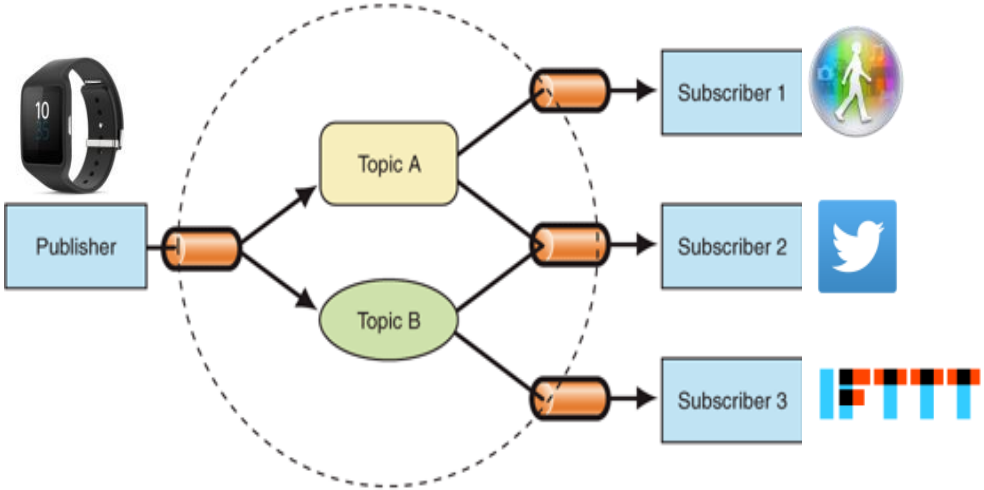


Publish-Subscribe - MQTT

- Publish–Subscribe, “pubsub”, is a messaging pattern where senders of messages, called publishers, do not send the messages directly to specific receivers. Instead, subscribers express interest in one or more message topics, and only receive messages that are of interest, without knowledge of what, if any, publishers there are.

- MQTT is a pubsub protocol for connections with remote locations where a "small code footprint" is required and/or network bandwidth is limited
- HW of typically 32 kb RAM and 128-256 kb Flash.





More protocol options

- Of course there are more options depending on HW capabilities and use cases, e.g.
  - HTTP
  - Web Sockets – Bidirectional communication
  - XMPP – Application protocol for streaming XML elements
  - Security
    - TLS or DTLS (if CoAP is used)
    - OAuth


W3C®

W3C Web of Things Interest Group - Overview

SONY

- Problem: The IoT suffers from fragmentation and product silos
- The mission of the Web of Things Interest Group, is to accelerate the development of open markets of applications and services based upon the role of Web technologies for a combination of the Internet of Things (IoT) with the Web of data.
- This means:

  “Extending the Web from a Web of Pages into a Web of Things”

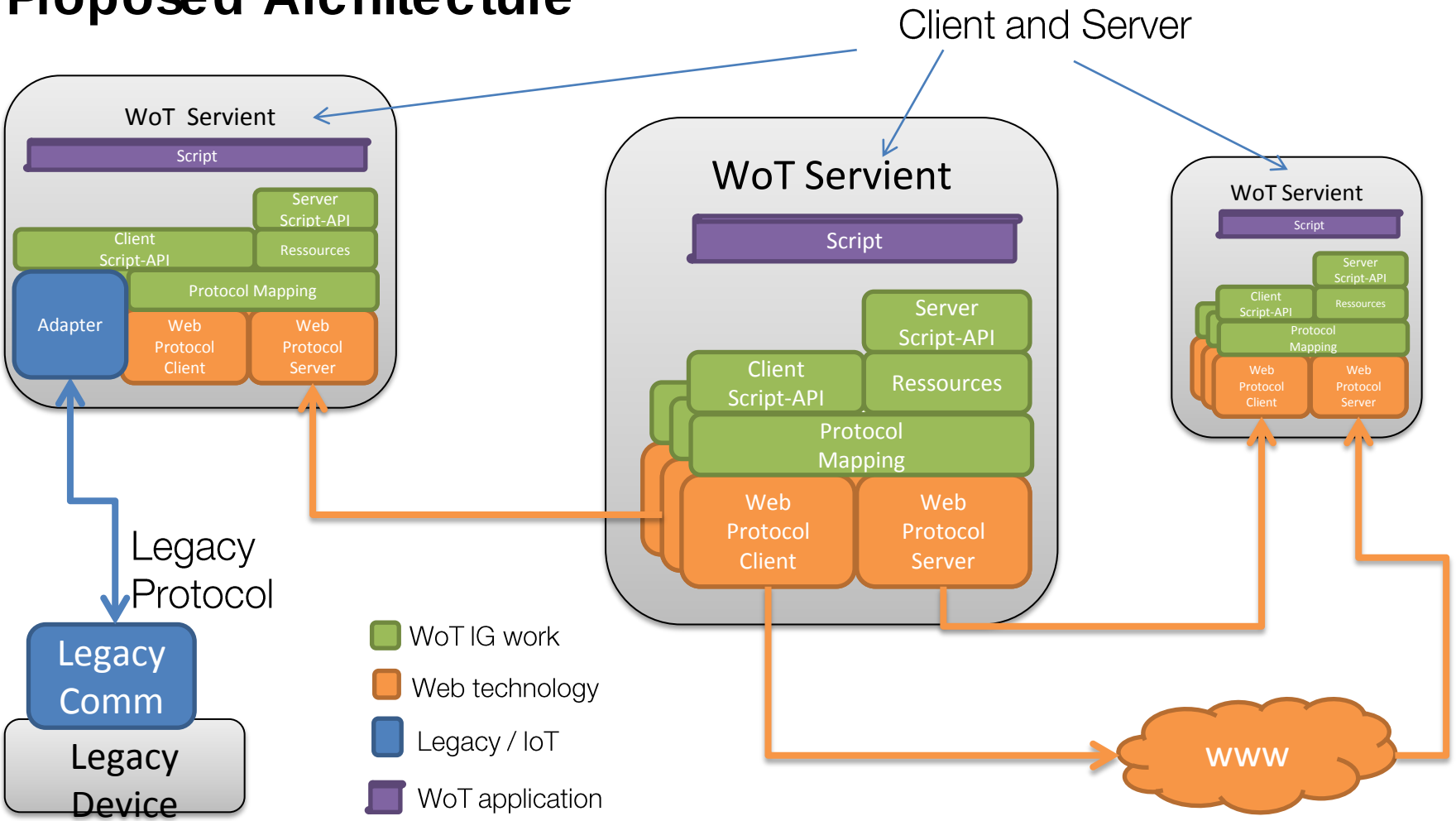
- No normative specifications. This will instead be done in a Web of Things Working Group, draft charter.

Deliverables

- Use Cases and Requirements
- Landscape of Existing Practices and Standards Relevant to the Web of Things
- Guidelines on Best Practices
- Requirements for Open Markets of Products and Services for the Web of Things
- High level architecture for the Web of Things
- End to End Security for the Web of Things

Proposed Architecture

SONY

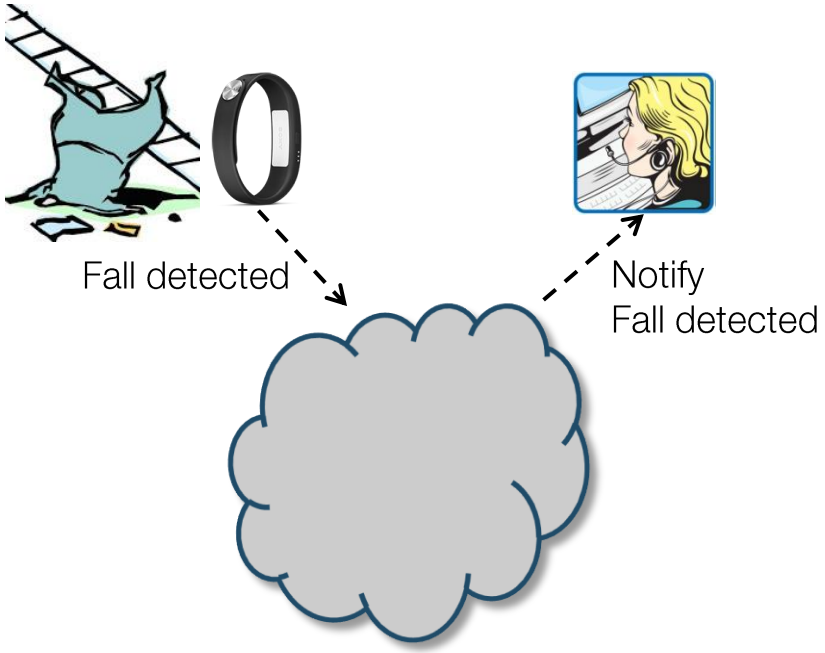


Task Forces

- Thing Description language (TDL)
  - "From HTML for pages to a Thing Description Language for things"
  - States metadata, data types, properties, actions and events for a Thing
  - Examples of TDLs
- Scripting APIs and Protocols
  - API interaction primitives. Example of JS API
  - Mapping to protocols. Example of mapping to CoAP
- Discovery
  - High level simple local and remote discovery independent of the technologies used by the physical things (UPnP, mDNS, Bluetooth discovery, Directories, ….).
  - Example of high level discovery and interaction API.
- Security and Privacy
  - Identify security and privacy gaps that need to be fixed with new standards.

Use case example – Remote health monitoring

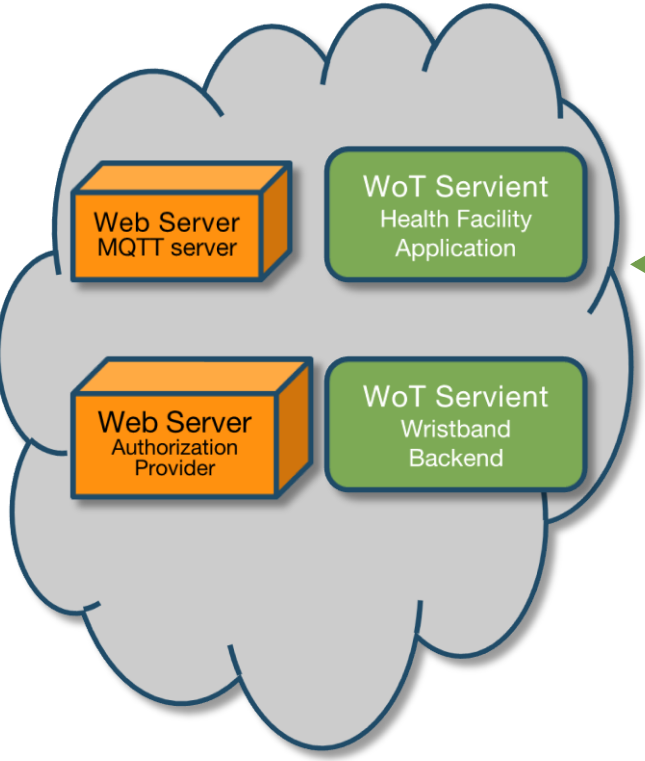
- Description
  - Wristlet monitors falls, heart rate, irregular heart rhythms, blood glucose, etc.
  - Issues health status and alarms to healthcare service
- Requirements
  - Communication must be reliable and secure (encryption and authentication)
  - Power consumption in the wristband must be low to achieve long battery life.
  - User interaction only required at system installation. The user, or another trusted person, e.g. a relative, health care personnel or personal assistant, has to use a web browser to log in to the remote monitoring application and the user has to approve that the application is given access to his/her wristband using an existing authorize system, e.g. OAuth.



Mapping use case to proposed W3C WoT architecture

SONY

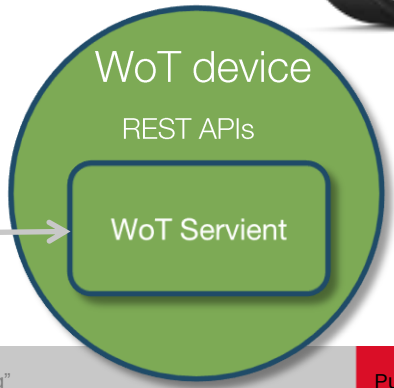
Sensors/Actuators:
- Fall Detection
- Heart Rate
- Heart Rythm
- Blood Glucose
- Vibrator
- .....



WoT protocols (HTTP, CoAP, MQTT…)

Optionally router/proxy depending on connection type

BLE, WiFi, IEEE 802.15.4, 3GPP MTC,….



”Thing”

Examples of properties, actions and events of the Wristband WoT Servient

- Properties
  - Battery Status
- Actions to invoke
  - Start fall detection
  - Vibrate
- Events generated
  - Initiate registration and authorization
  - Fall detected

SONY

Technical details follow

Thing Description for Wristband

```
{
   "@context": "http://www.w3c.org/wot/td#",
   "metadata": {
      "name": "FallDetectionWristband",
      "protocols" : {
         "coap" : {
            "uri" : "coap://????/wristband",
            "priority" : 1
         },
         "http" : {
            "td:uri" : "http://?????/wristband",
            "priority" : 2
         }
      },
      encodings": [
         "JSON"
      ]
   },
```

```
"interactions": [
      {
         "@type": "Action",
         "name": "registerFallDetection",
         "inputData": "xsd:string",
         "outputData": ""
      },
      {
         "@type": "Action",
         "name": "vibrate",
         "inputData": "vibrateType",
         "outputData": ""
      },
      {
         "@type": "Event",
         "outputData": "xsd:string",
         "name": "fallDetected"
      }
   ],
```

```
"td:Data": [
      {
         "vibrateType" :
         {
            "accessToken":"xsd:string",
            "pattern": "xsd:string"
         }
      }
   ]
}
```

Note: This use case example is a ”thing-to-thing interaction” use case so the cloud health monitoring app is also a Thing and needs a Thing Description!

JavaScript API independent of used protocol

```
/* Client=Wristband , Server=Cloud Server */
// Client (Wristband)  connects to Server (cloud).
var thing = wot.connect("coap://falldetection.example.com/");
// Server (cloud) exposes thing (falldetection app)
var metadata = {...} //to be defined by TF-TD
var mything = wot.expose(metadata);
// Client (wristband) , thing is cloud falldetection app. Init  cloud app.
thing.callAction('init',{ "accessToken" : "12345678",
                          "deviceID" : “AnnasDevice", "userID" : “Anna" })
    .then(function(res) {
        console.log("Falldetection app initiated successfully" );
    })
    .catch(function(err) {
        console.err(err),
    });
// Server (cloud) – mything is proxy for falldetection app
mything.registerAction('init',
  function(params) {
    //....init falldetection app
  });
```

```
/* Server=Wristband,  Client=Cloud Server */
// Client (cloud) connects to Server (wristband) by dynamic IP/port
var thing = wot.connect("x.x.x.x:yyyy/wristband");
//Server (wristband) exposes thing
var metadata = {...} //to be defined by TF-TD
var mything = wot.expose('Wristband',metadata);
/* Actions */
// Client (cloud), thing is a remote wristband
thing.callAction('registerFallDetection',{ "accessToken" : 212345678"})
    .then(function(res) {
        console.log("Falldetection registerered");
    })
    .catch(function(err) {
        console.err(err),
    });
// Server (wristband) - mything is a proxy for a falldetection sensor
mything.registerAction('registerFalldetection',
      register_falldetection_params,
  function(accessToken) {
    //... check accessToken and start detecting falls
  });
```

```
/* Server=Wristband, Client=Cloud Server */
/* Events: A client implementation can subscribe and
      register callbacks for events, a server implementation
      can emit events. */
// Client (cloud), thing is remote wristband
thing.subscribe('fallDetected',function(evt) {
    console.log("Wristband " + evt.accessToken +
      " detected fall at “+ evt.time);
});
//Server (wristband) - mything is a proxy for a
      // falldetection sensor. Emit fall detected event
mything.emit('fallDetected',{ "accessToken" : "12345678"});
```

Mapping to CoAP

/* Action wristband/startFalldetection */

PUT coap://x.x.x.x:5683//wristband/startFalldetection

Payload Request:

```
{
    "accessToken" : "string"
}
```

Payload Response: Empty

/* Action wristband/vibrate */

POST coap://x.x.x.x:5683//smartband/vibrate

Payload Request:

```
{
    "accessToken":"string",
    "pattern": "string"
}
```

Payload Response: Empty

/* Event fallDetected. Message from device to cloud server */

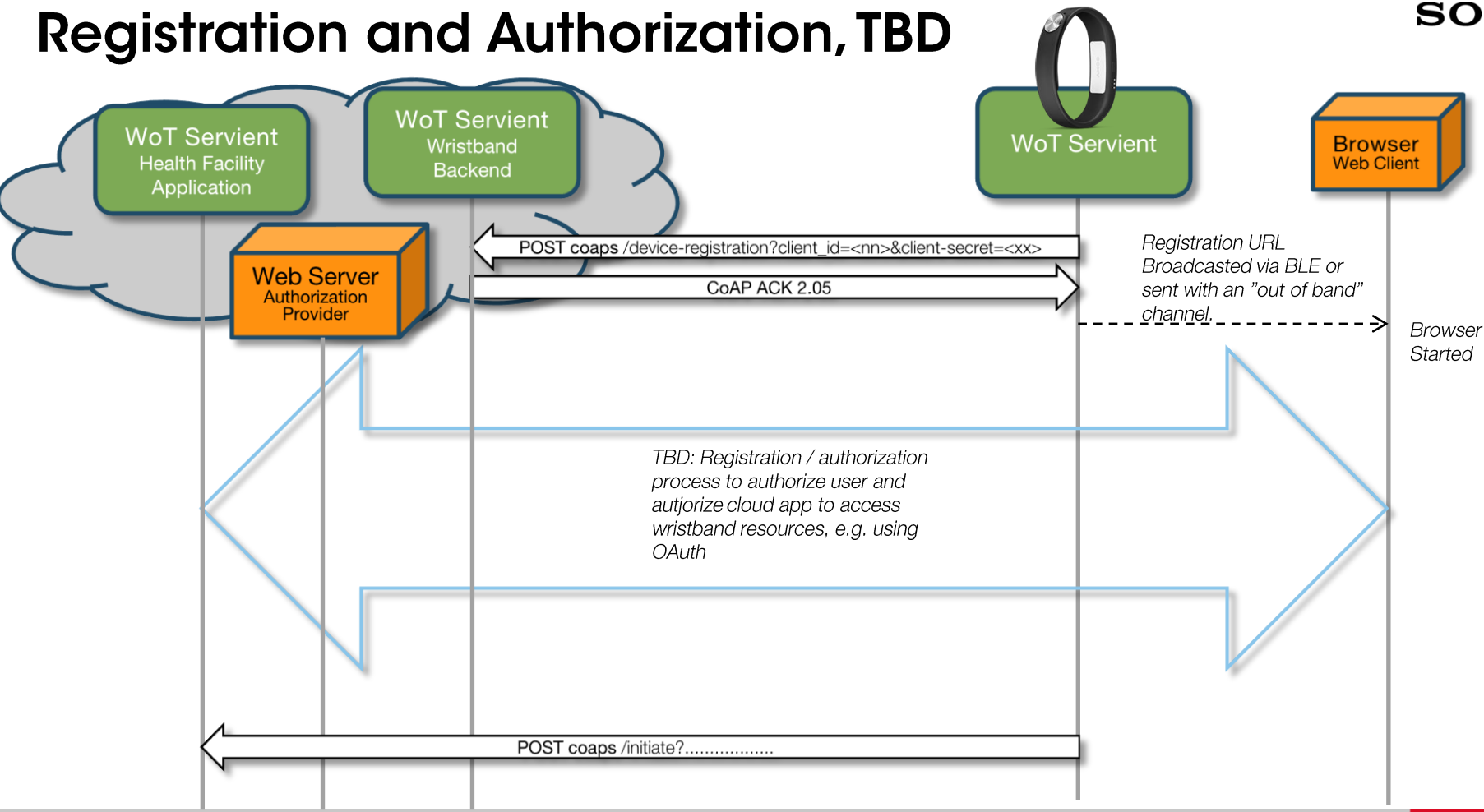
POST coap://x.x.x.x:5683//wristband/falldetected

Payload Request:

```
{
    "accessToken":"string"
}
```

Payload Response: Empty

Registration and Authorization, TBD



Fall Detection scenario example: CoAP

Wristband is registered and user has authorized the Health Facility app to access the resources of the wristband (an access-token is available)

PUT coaps /wristband/startFalldetection Payload: {“accessToken” : “12345678” }

CoAP ACK 2.05

.
.
.

POST coap /sensors/falldetected Payload: {“accessToken” : “12345678” }

Fall happens!

Rescue action is started. Confirm that fall has been detected by vibrating wristlet.

CoAP ACK 2.05

POST coaps /wristband/vibrate Payload: {“accessToken” : “12345678”, “pattern”:”500,200,500”}

CoAP ACK 2.05

Fall Detection scenario example: CoAP + MQTT