

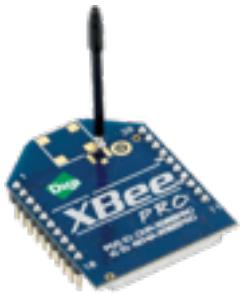
# Getting Started with a W3C WoT Project

RIOT Summit, Berlin, Germany, 2016

# What is the Web of Things?

## Application Layer

## Internet of Things : **Connectivity**



IEEE 802.15.4



Ethernet



Wi-Fi



Bluetooth



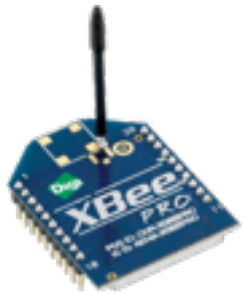
LoRa

...

# What is the Web of Things?



## Internet of Things : Connectivity



IEEE 802.15.4



Ethernet



Wi-Fi



Bluetooth



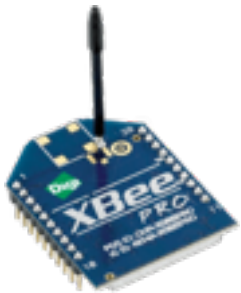
LoRa

...

# What is the Web of Things?

## Web of Things: Applications

## Internet of Things : Connectivity



IEEE 802.15.4



Ethernet



Wi-Fi



Bluetooth



LoRa

...

# W3C WoT Mission

**Not to be yet another standard**



Web of Things



...

“interconnecting existing Internet of Things platforms  
and complementing available standards”

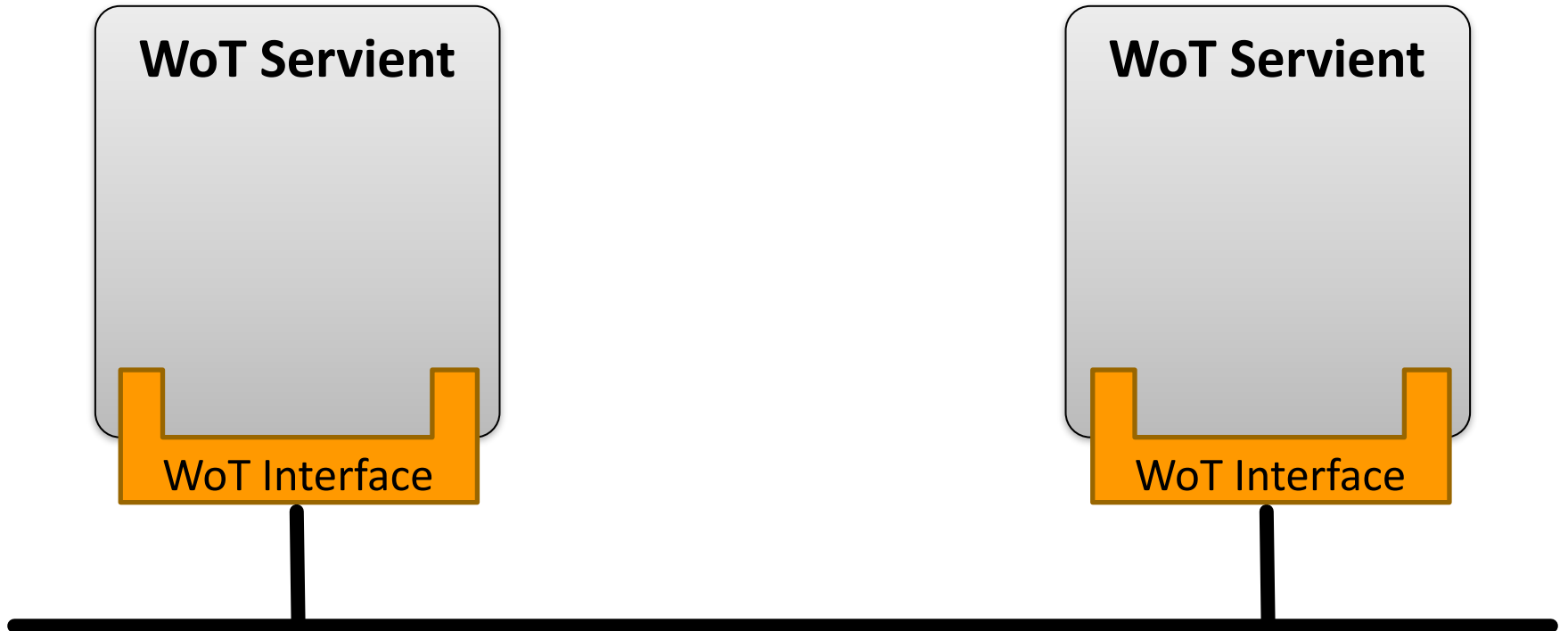
IoT Platforms and Protocol Bindings

<http://w3c.github.io/wot/current-practices/wot-practices.html#wot-interface>

# **WoT INTERFACE**

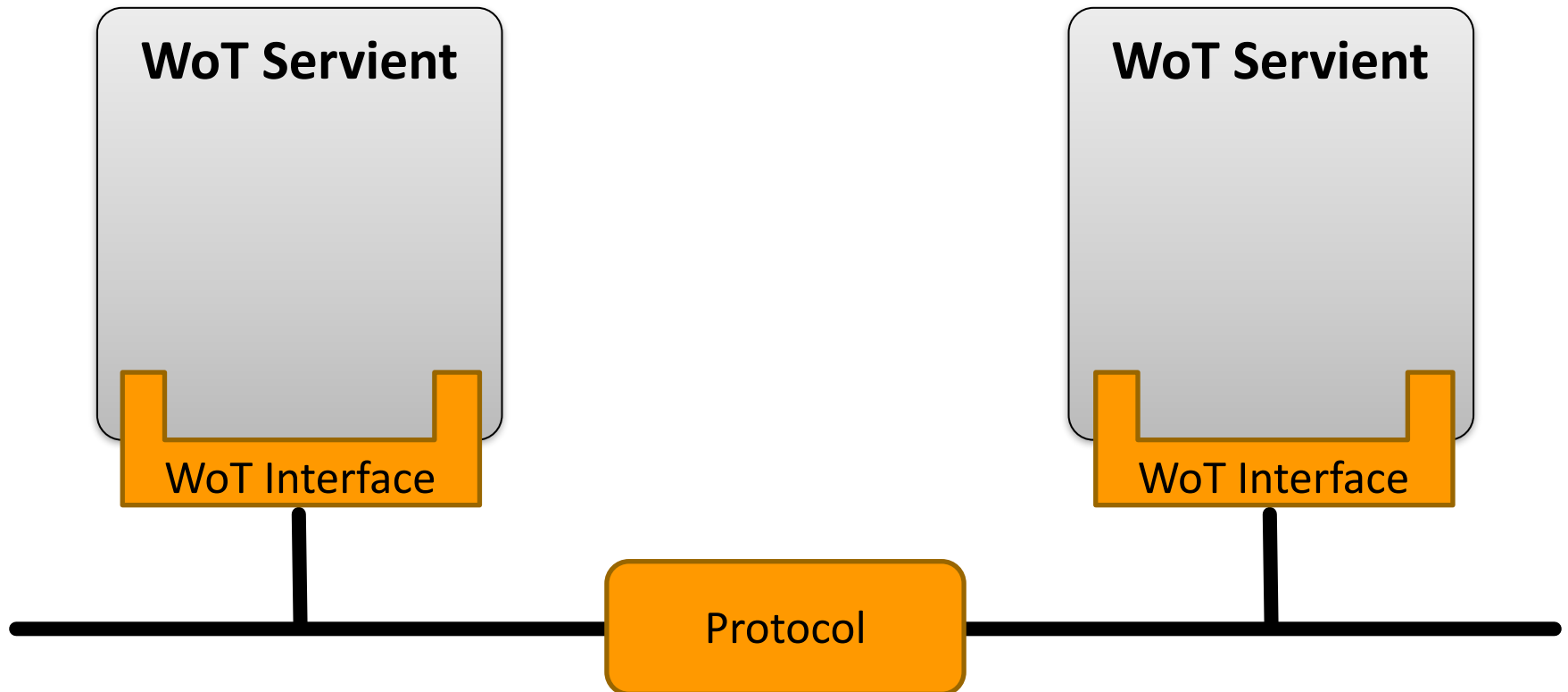
# WoT Interface

- Interface exposed by Servients to the network



# WoT Interface

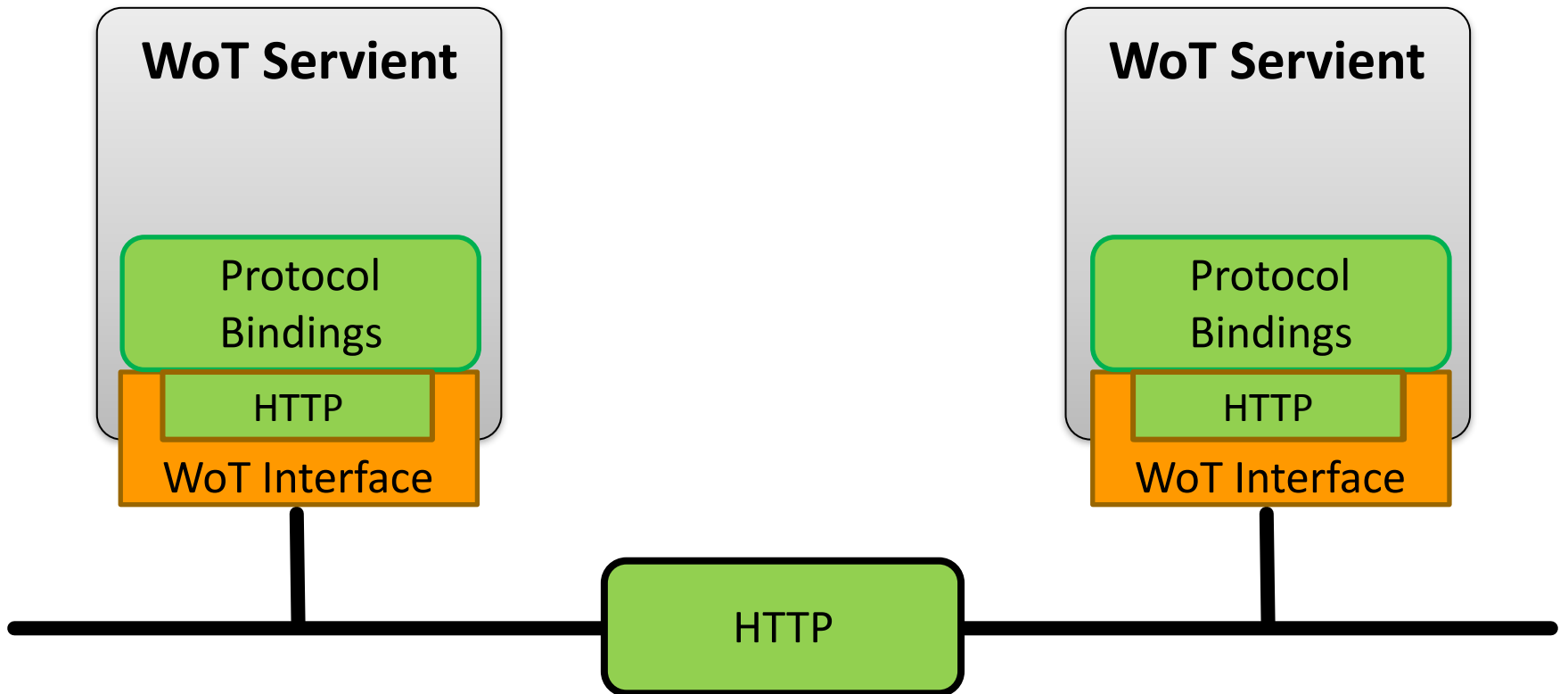
- Interface exposed by Servients to the network





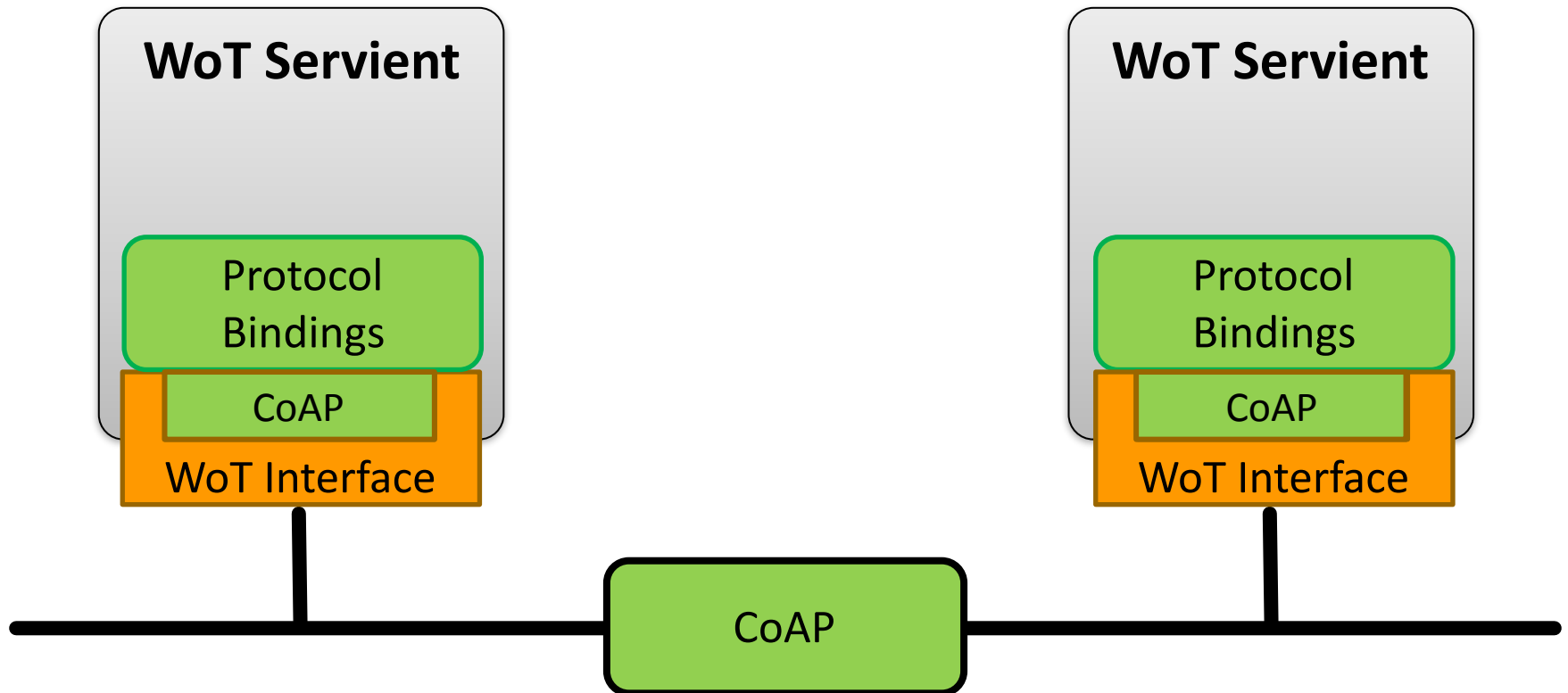
# Protocol Bindings

- Interface can be bound to various protocols



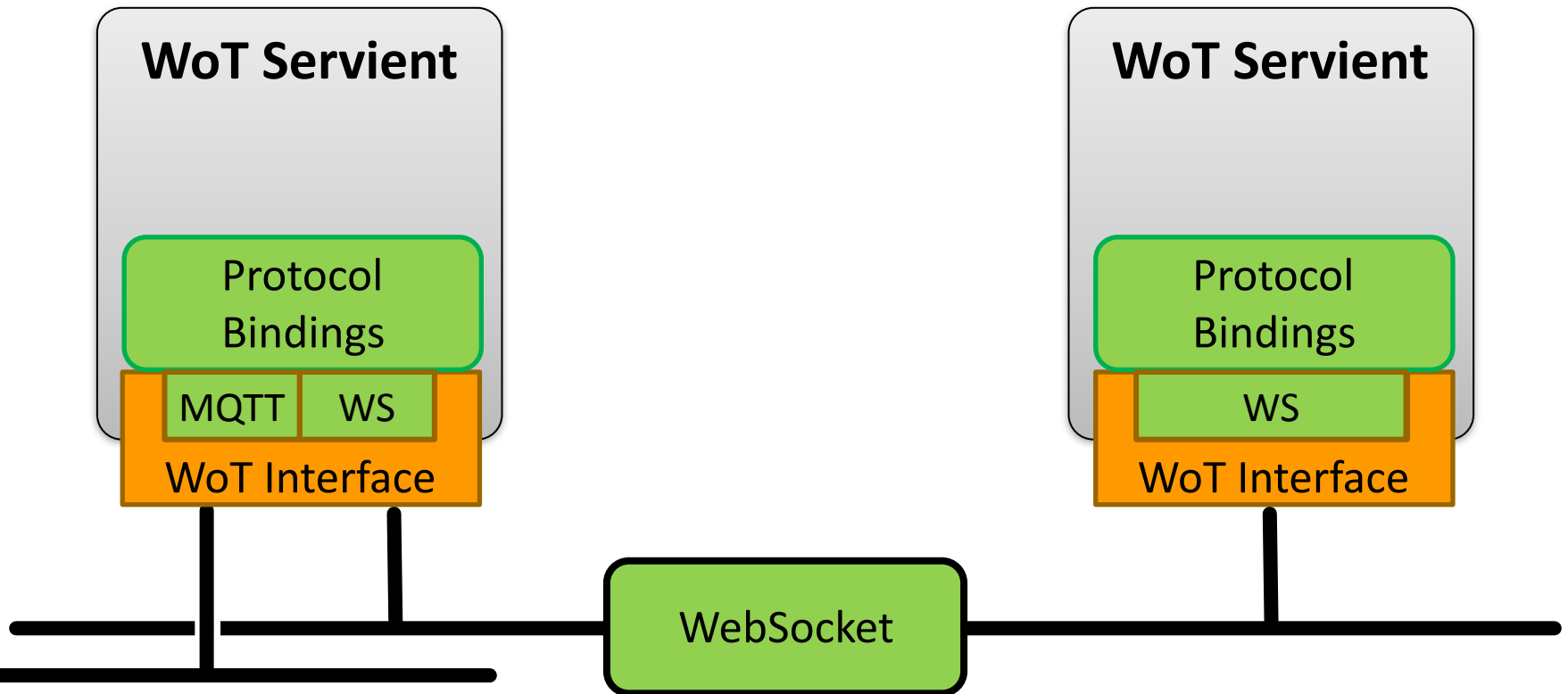
# Protocol Bindings

- Interface can be bound to various protocols



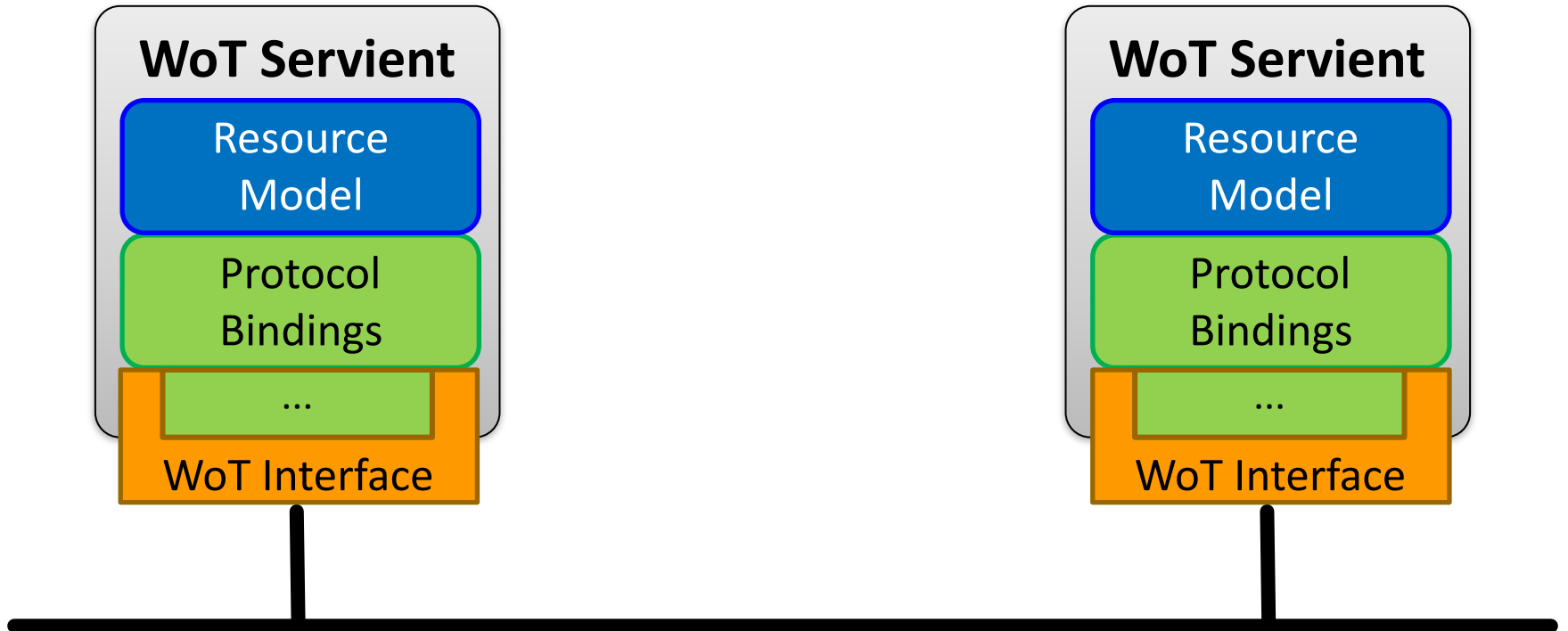
# Protocol Bindings

- Multiple bindings possible on Things



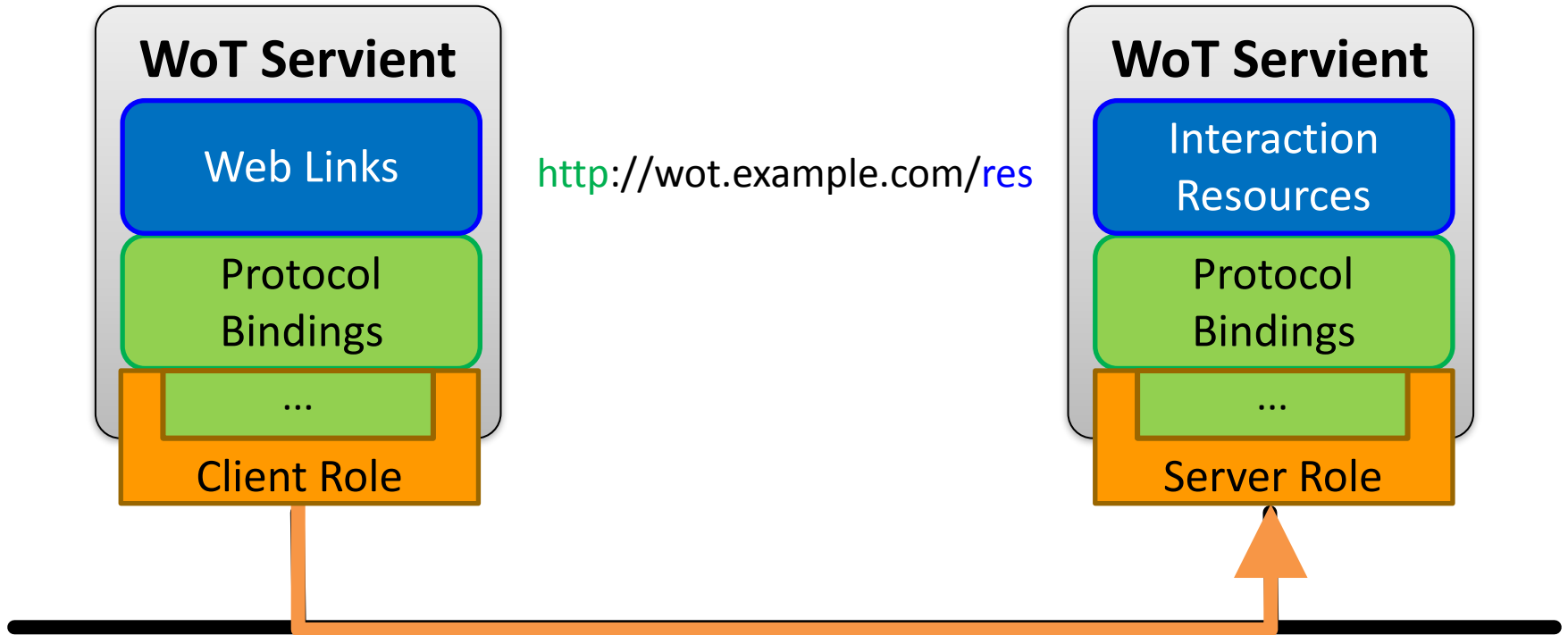
# Resource Model

- Interaction points are Web resources



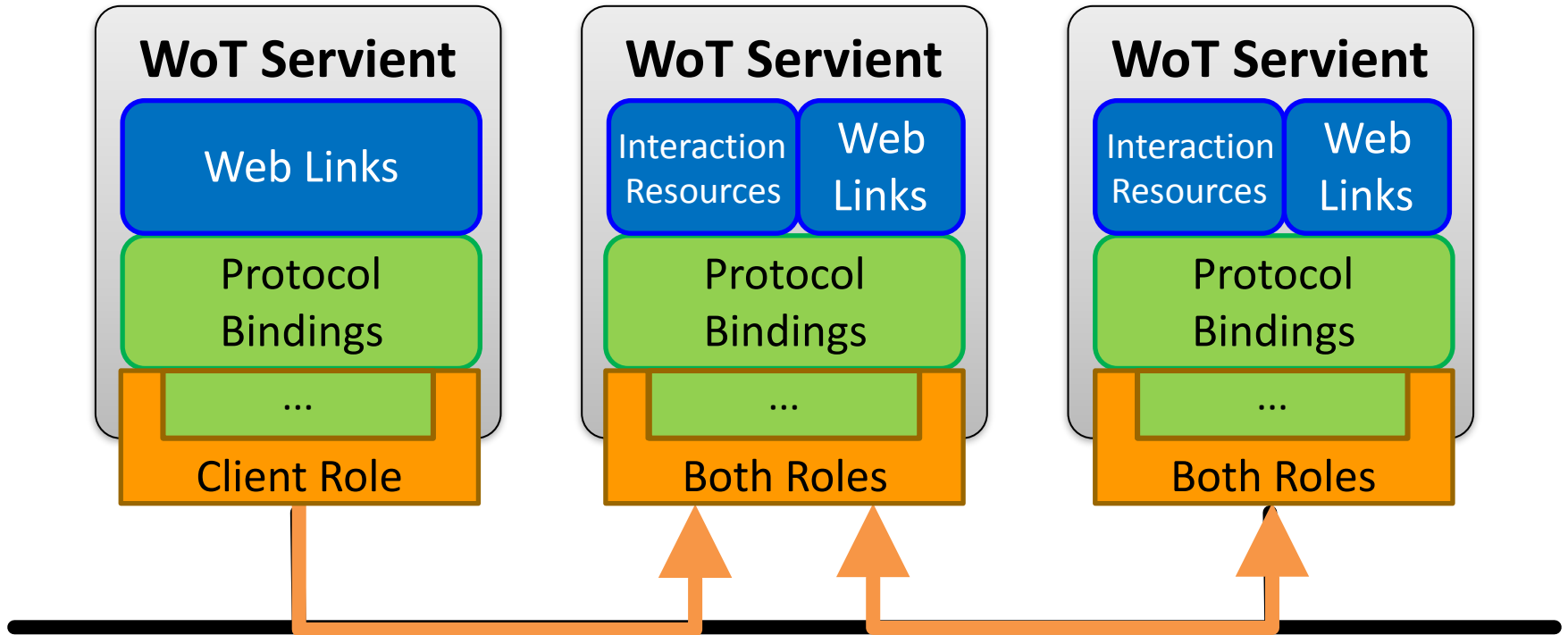
# Servient Role

- Exposing Things are in server role
- Consuming Things are in client role



# Servient Role

- Usually both roles at the same time → *Servient*



Metadata and Interactions

<http://w3c.github.io/wot/current-practices/wot-practices.html#thing-description>

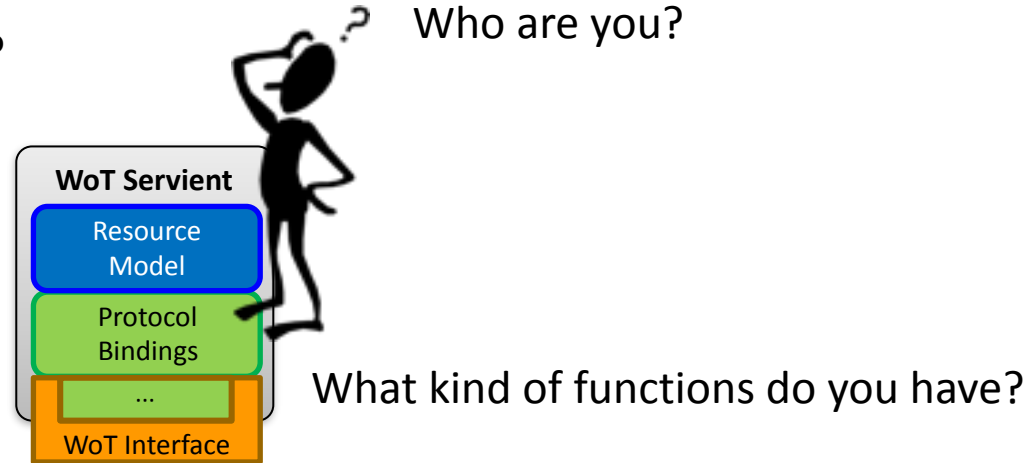
# **THING DESCRIPTION**

# How to Interact with WoT Servients?

What kind of data do you serve?

Who are you?

How can I access the data/function?



What kind of protocols/encodings do you support?

Are there some security constrains?

→ W3C Thing Description

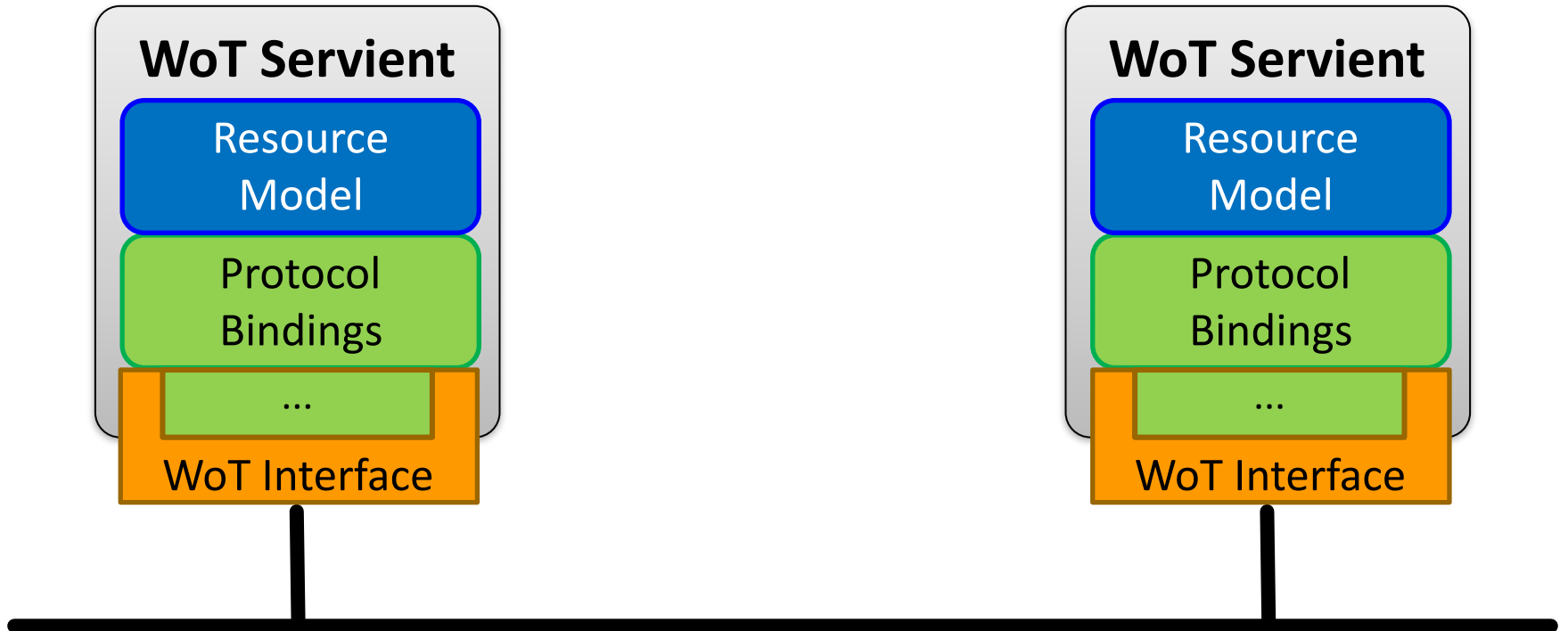


# Semantic Description

- Reach interoperability through Linked Data vocabularies
  - subject, predicate, object triples
  - rooted in the RDF model
- W3C Thing Description
  - describes WoT Interface to interact with Things
  - extensible with domain-specific vocabulary
  - different serializations possible

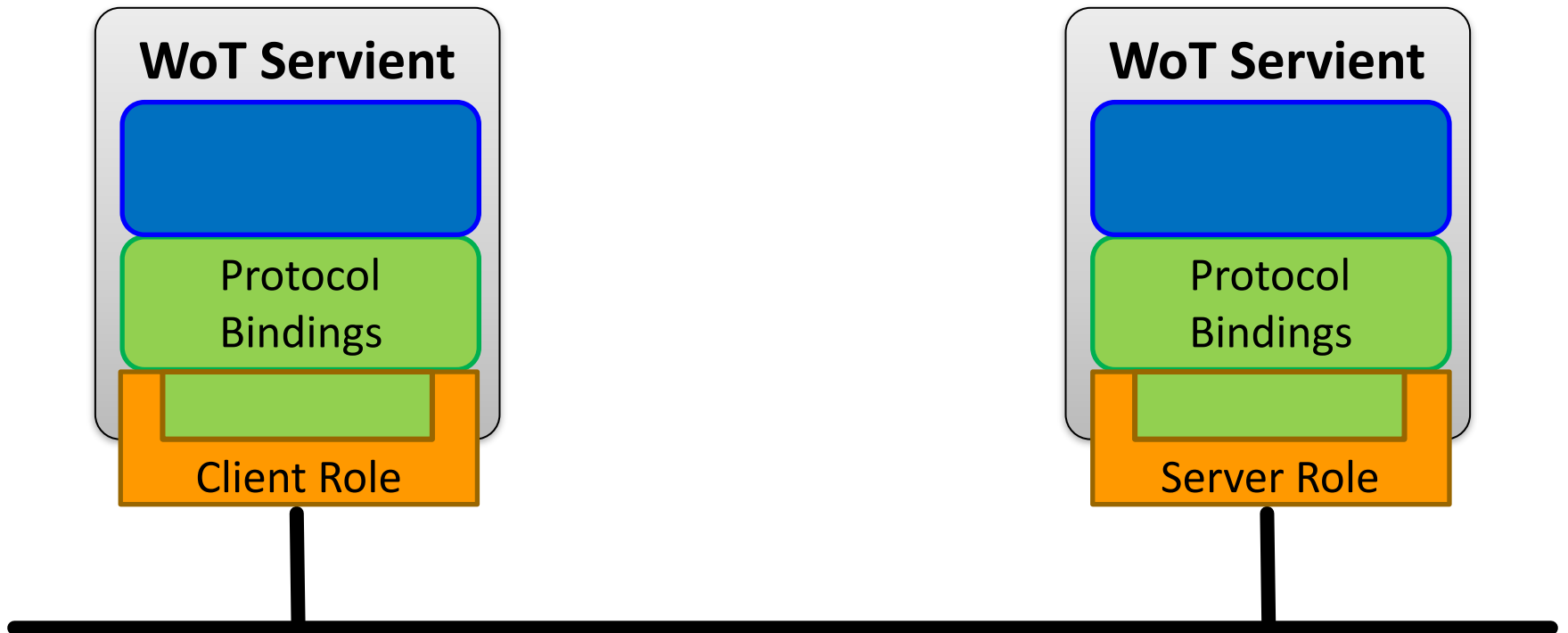
# Thing Description (TD)

- Describes Thing metadata and interactions



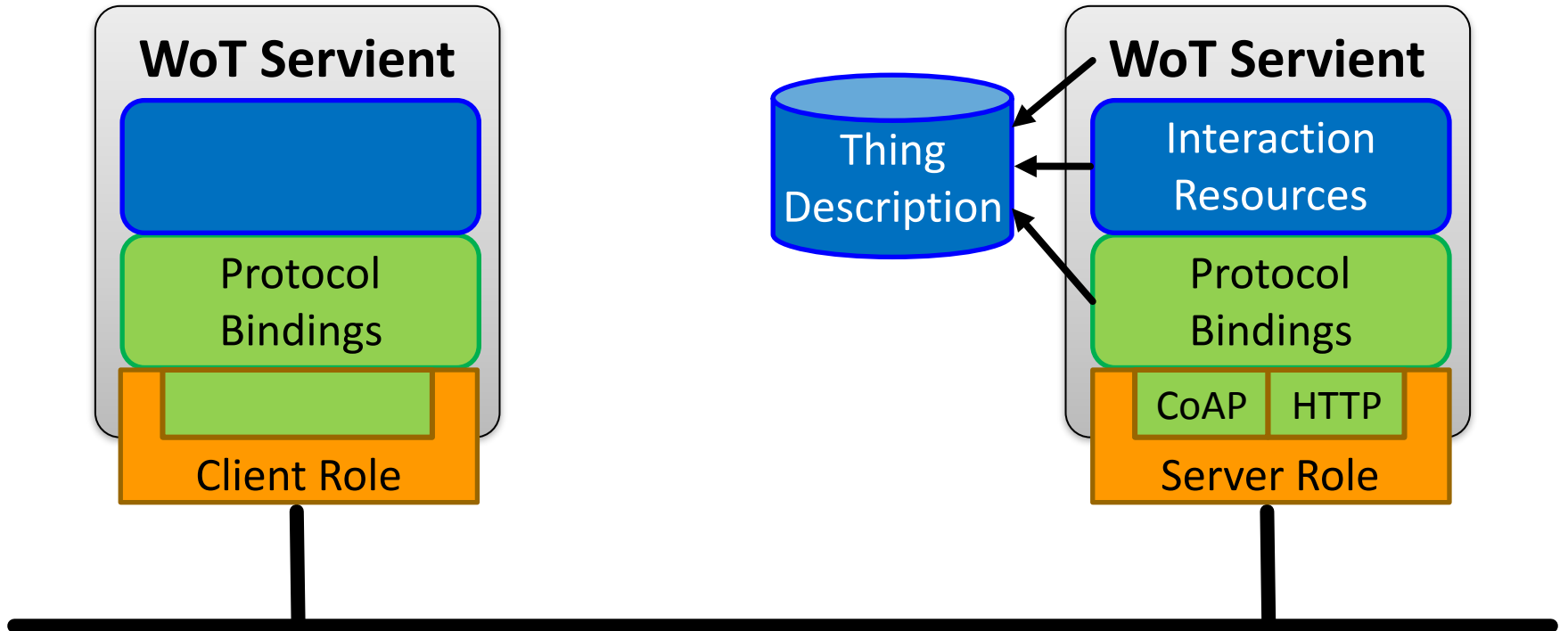
# Thing Description (TD)

- Consuming Things are in client role
- Exposed Things are in server role



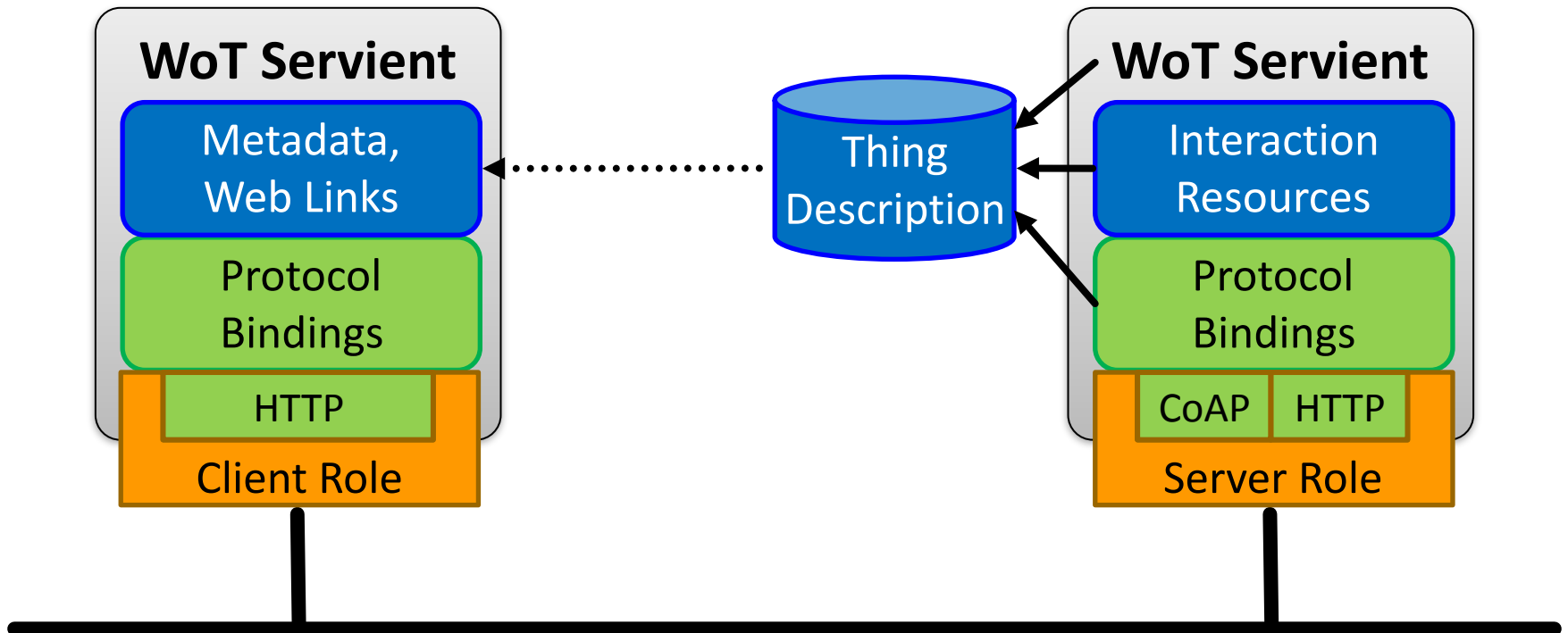
# Thing Description (TD)

- Exposed Things provide Thing Description



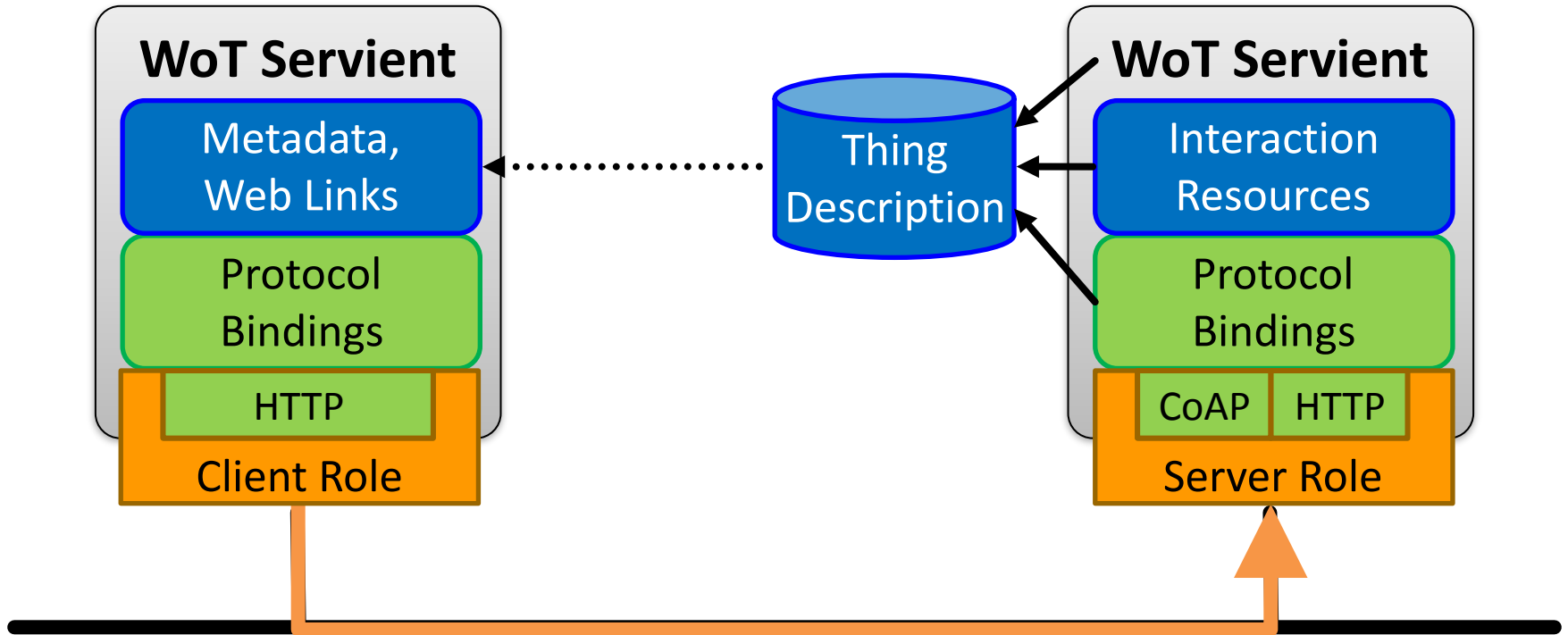
# Thing Description (TD)

- Consuming Things learn WoT Interface from TD



# Thing Description (TD)

- Thing-to-thing communication



# Thing Description (TD)

- Default serialization is JSON-LD
  - based on well established JSON format
  - different implementations and tools available
  - @context defines vocabularies
  - See TD example

# TD Example

```
{
  "@context": [
    "http://w3c.github.io/wot/w3c-wot-td-context.jsonld",
    { "actuator": "http://example.org/actuator#" }
  ],

  "@type": "Thing",
  "name": "MyLEDThing",

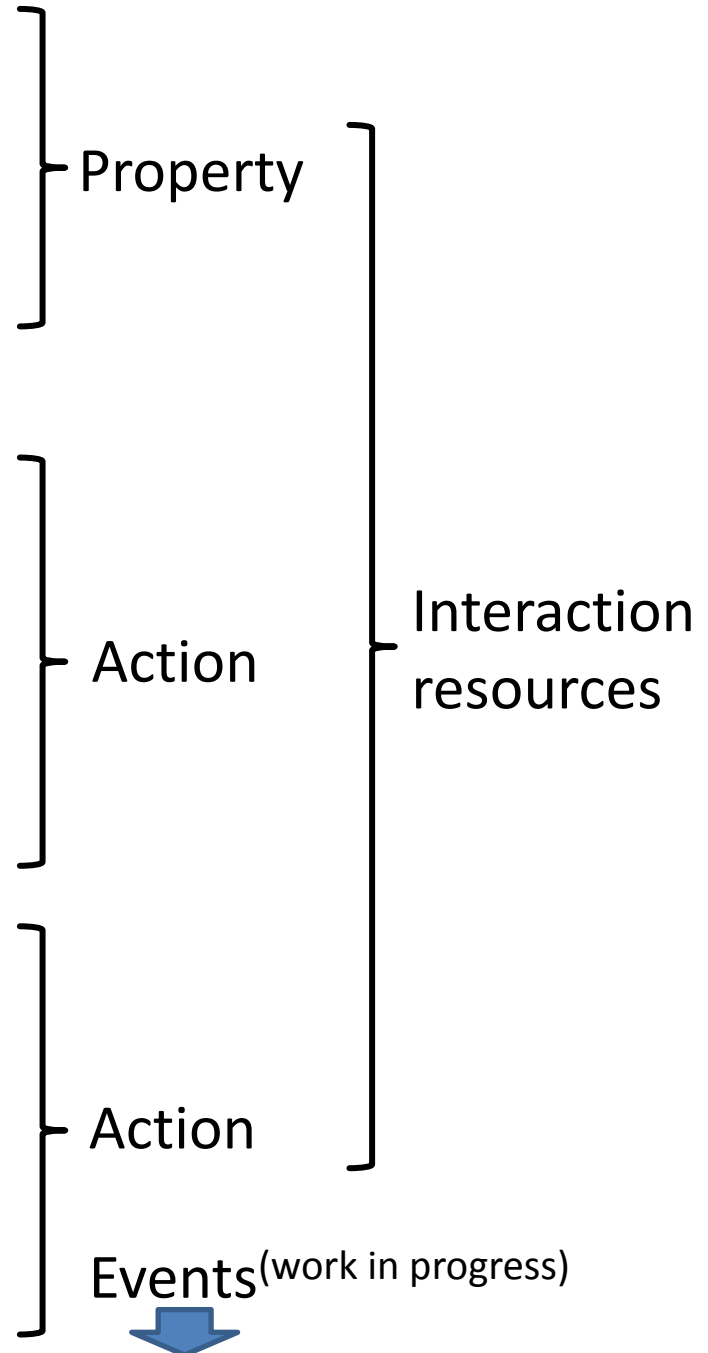
  "uris": [
    "coap://myled.example.com:5683/",
    "http://mything.example.com:8080/myled/"
  ],

  "encodings": ["JSON", "EXI"],
  "security": {
    "cat": "token:jwt",
    "alg": "HS256",
    "as": "https://authority-issuing.example.org"
  },

  "properties": [
```



```
"properties": [
  {
    "@type": "actuator:onOffStatus",
    "name": "status",
    "valueType": { "type": "boolean" },
    "writable": true,
    "hrefs": [ "pwr", "status" ]
  }
],
"actions": [
  {
    "@type": "actuator:fadeIn",
    "name": "fadeIn",
    "inputData": {
      "valueType": { "type": "integer" },
      "actuator:unit": "actuator:ms"
    },
    "hrefs": [ "in", "led/in" ]
  },
  {
    "@type": "actuator:fadeOut",
    "name": "fadeOut",
    "inputData": {
      "valueType": { "type": "integer" },
      "actuator:unit": "actuator:ms"
    },
    "hrefs": [ "out", "led/out" ]
  }
]
```



# Type System

- Default currently based on JSON Schema  
<http://w3c.github.io/wot/current-practices/wot-practices.html#type-system>
- Best start with simple types
  - boolean
  - integer
  - number
  - string
- Other systems can be plugged in under “valueType” field

# How to Create a TD?

- Manually copy, paste, and modify
  - <http://w3c.github.io/wot/current-practices/wot-practices.html#td-examples>
  - or look into the TD repository  
<http://vs0.inf.ethz.ch:8080>  
(development repository, sometimes offline)
- Generate from development framework
  - TD serialization based on the interactions provided

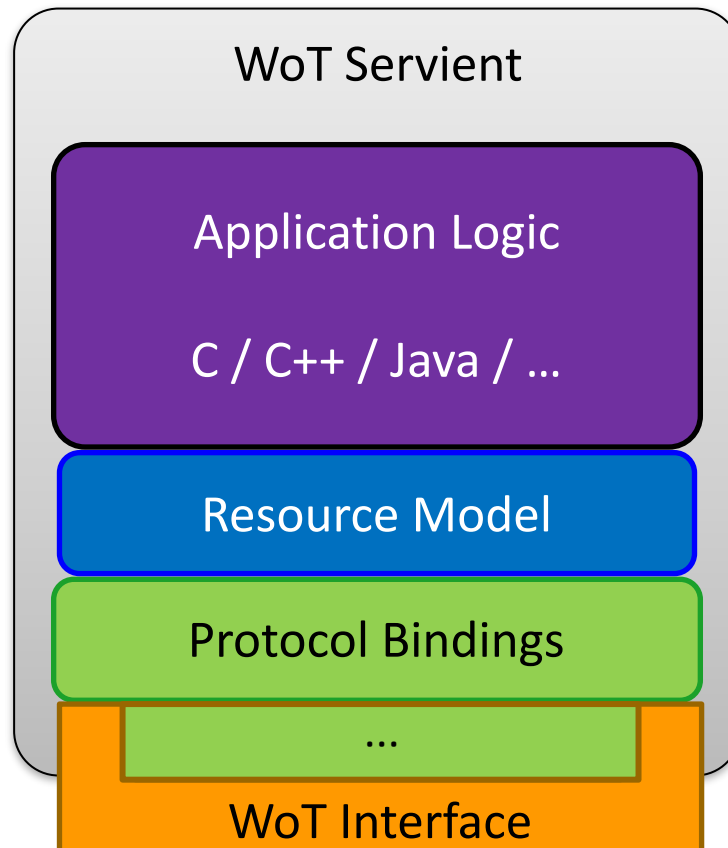
Runtime Environment and Portable Apps

[http://w3c.github.io/wot/current-practices/  
wot-practices.html#scripting-api](http://w3c.github.io/wot/current-practices/wot-practices.html#scripting-api)

# SCRIPTING API

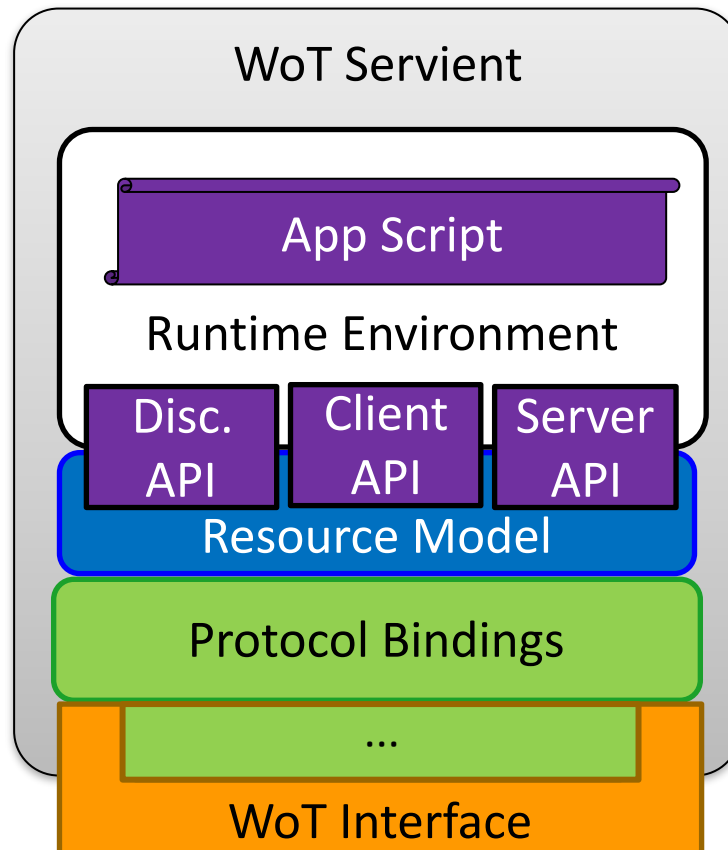
# Without Scripting API

- Application logic often implemented natively



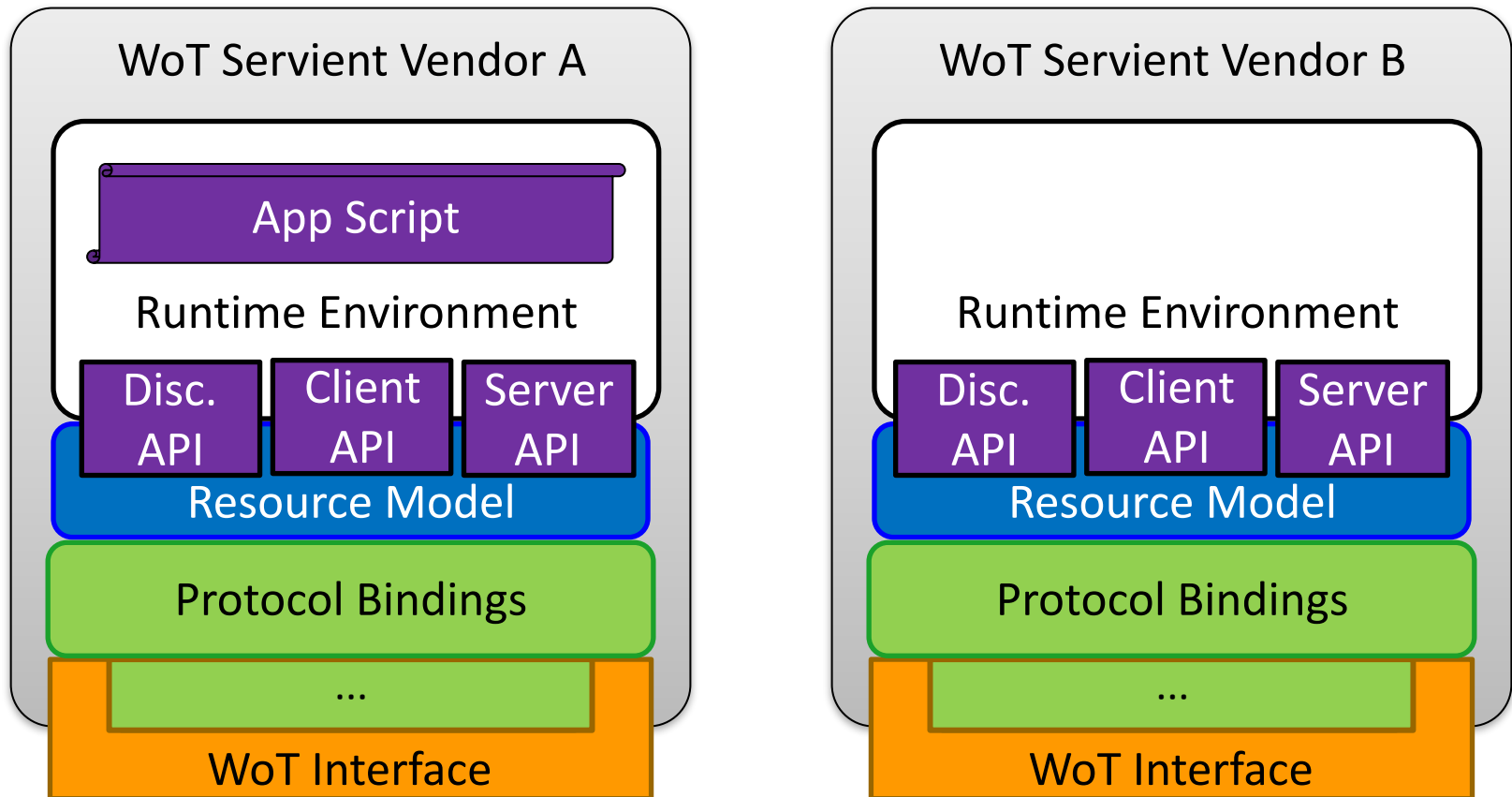
# Scripting API

- Common runtime enables portable apps



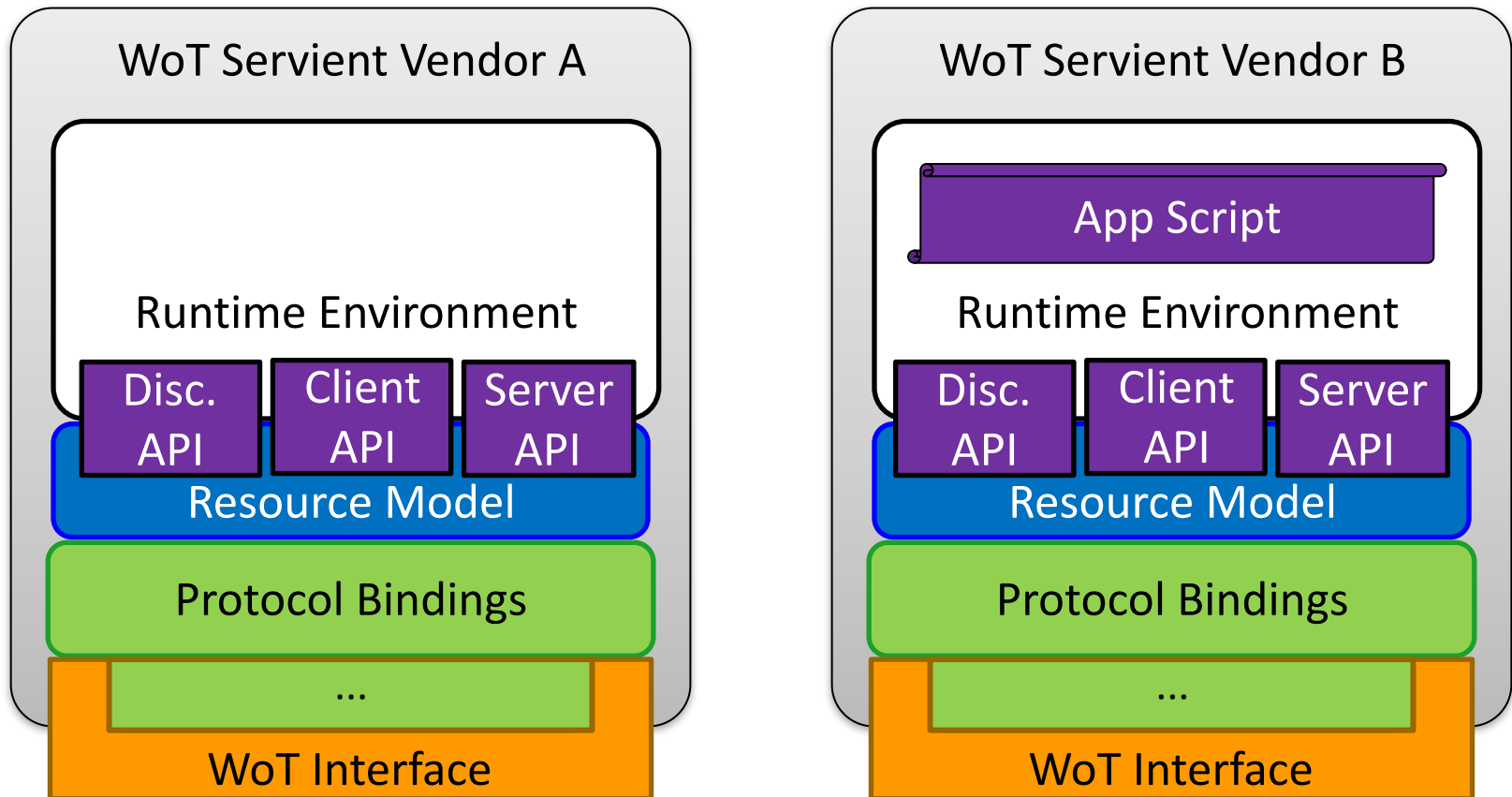
# Scripting API

- Common runtime enables portable apps



# Scripting API

- Common runtime enables portable apps





# WoT Root Element (with Discovery)

```
interface WoT {  
    Promise<sequence<ConsumedThing>> discover(ThingFilter filter);  
    Promise<ConsumedThing> consumeDescription(Object td);  
    Promise<ConsumedThing> consumeDescriptionUri(DOMString uri);  
    Promise<ExposedThing> createThing(DOMString name);  
    Promise<ExposedThing> createFromDescription(Object td);  
    Promise<ExposedThing> createFromDescriptionUri(DOMString uri);  
};
```

# Client API: ConsumedThing

```
interface ConsumedThing {  
    readonly attribute DOMString name;  
    Promise<any> getProperty(DOMString propertyName);  
    Promise<any> setProperty(DOMString propertyName, any newValue);  
    Promise<any> invokeAction(DOMString actionName, any parameter);  
    ConsumedThing addListener(DOMString eventName,  
                               ThingEventListener listener);  
    ConsumedThing removeListener(DOMString eventName,  
                                  ThingEventListener listener);  
    ConsumedThing removeAllListeners(DOMString eventName);  
    Object getDescription();  
};
```

# Server API: ExposedThing

```
interface ExposedThing {  
    readonly attribute DOMString name;  
    ExposedThing addProperty(DOMString name, object type);  
    ExposedThing addAction(DOMString name, Object input, Object output);  
    ExposedThing addEvent(DOMString name, Object output);  
    Promise<any> getProperty(DOMString propertyName);  
    Promise<any> setProperty(DOMString propertyName, any newValue);  
    Promise<any> emitEvent(DOMString eventName, any payload);  
    ExposedThing onUpdateProperty(DOMString n, PropertyChangeListener cb);  
    ExposedThing onInvokeAction(DOMString actionName, ActionHandler cb);  
    Object getDescription();  
};
```

# Script Example (Consume Thing)

```
WoT.consumeDescriptionUri("http://servient.example.com/things/counter")
  .then(function(counter) {
    counter
      .invokeAction("increment", {}).then(function() {
        console.log("incremented");
        counter
          .getProperty("count").then(function(count) {
            console.log("new count state is " + count);
          });
      })._catch(console.error);
  })
  ._catch(function(err) {
    console.error(err);
  });
```

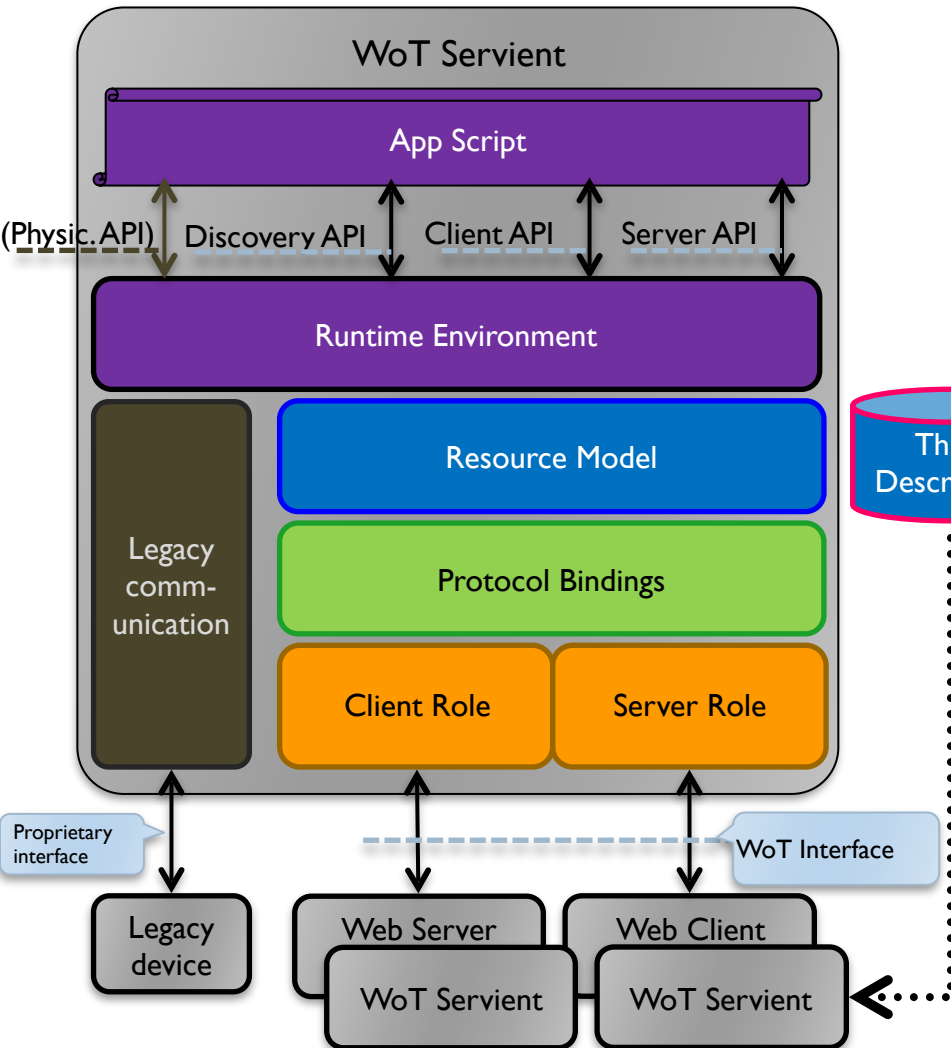
# Script Example (Expose Thing)

```
WoT.newThing("counter")
  .then(function(thing) {
    thing
      .addProperty("count", {"type": "integer"})
      .addAction("increment")
      .onInvokeAction("increment", function() {
        console.log("incrementing counter");
        var value = thing.getProperty("count") + 1;
        thing.setProperty("count", value);
        return value;
      });
    thing
      .setProperty("count", 0)
  });
```

W3C Web of Things

# **SUMMARY**

# Thing Implementation: WoT Servient



## Application Logic:

Can consume remote Things through the Client API, local hardware and connected legacy devices through a Physical API (t.b.d.), and expose Things through the Server API. To allow portable app scripts, the Servient must provide a runtime environment.

## Resource Model:

Provides a common abstraction with uniform interface across the different protocols. Like the Web, it allows to identify and address interaction points through URIs.

## Thing Description (TD):

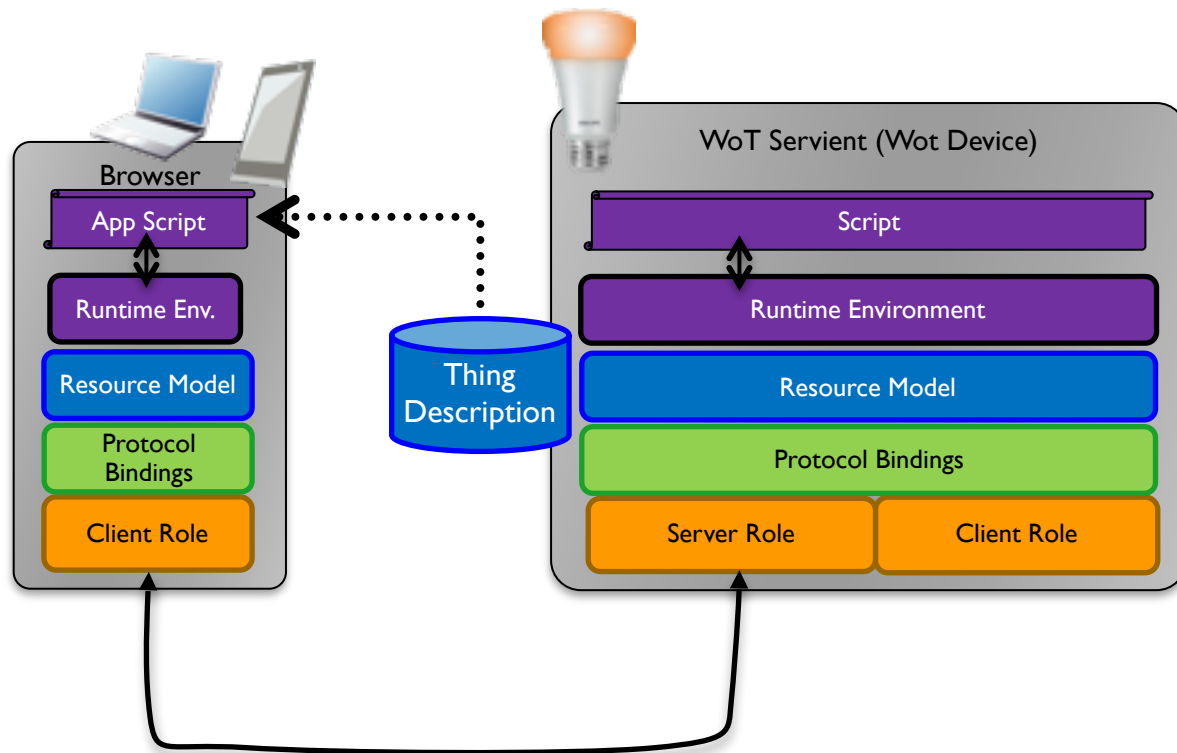
Declares WoT Interface for interaction and provides (semantic) metadata for the Thing. TD is used by WoT clients to instantiate local software object of the Thing.

## Protocol Binding:

Converts abstract interactions with Things to different protocols using the information from TD.

# WoT Servient on Thing Itself

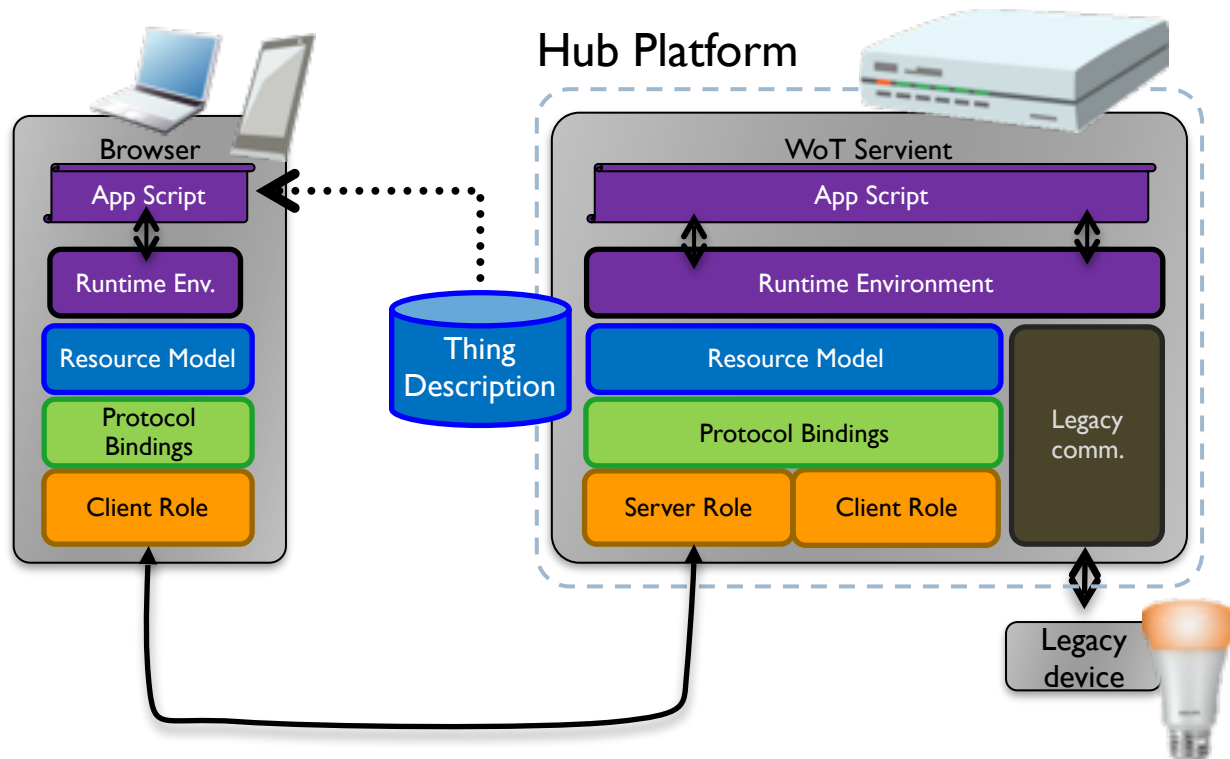
- Native WoT Things host a Servient directly
- TD is provided by Thing or supporting host on the Web





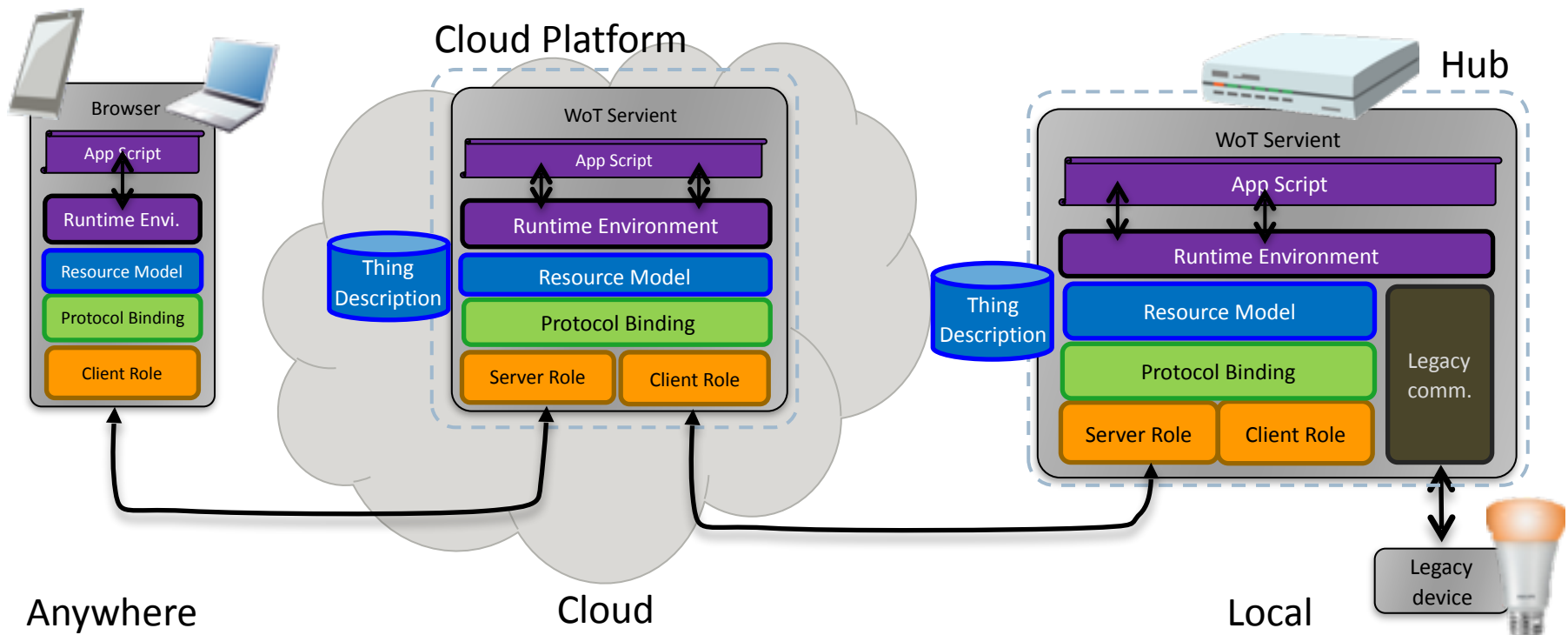
# WoT Servient on Integration Hub

- WoT Servients can run on hubs (e.g., smartphone, gateway)
- Multiple Servients can be instantiated through sandboxed apps
- Apps can act as agents/proxies for legacy devices



# WoT Servient in the Cloud

- A cloud mirror (device shadow) enables scalable remote access
- Is synchronized with local Servient
- Can forward interactions and cache data



# Online Resources

- Interest Group
  - <https://www.w3.org/WoT/IG/>
  - <https://lists.w3.org/Archives/Public/public-wot-ig/> (subscribe to mailing list)
- Documents (for implementers)
  - <http://w3c.github.io/wot/architecture/wot-architecture.html>
  - <http://w3c.github.io/wot/current-practices/wot-practices.html> (living document)
  - Beijing 2016 Release:  
<http://w3c.github.io/wot/current-practices/wot-practices-beijing-2016.html>
- GitHub (documents and proposals)
  - <https://github.com/w3c/wot>
- Wiki (organizational information: WebConf calls, Face-to-Face meetings, ...)
  - [https://www.w3.org/WoT/IG/wiki/Main\\_Page](https://www.w3.org/WoT/IG/wiki/Main_Page)
- WoT Projects (implementing WoT Current Practices)
  - <https://github.com/thingweb/>
  - <https://github.com/mkovatsc/wot-demo-devices>
  - Please add yours!

W3C WoT F2F Beijing 2016

# PLUGFEST

# Scenario 1 – ‘Hello WoT’

WoT TD interpreter  
for human interaction



Setup servient  
interaction  
based on TD



**SIEMENS**

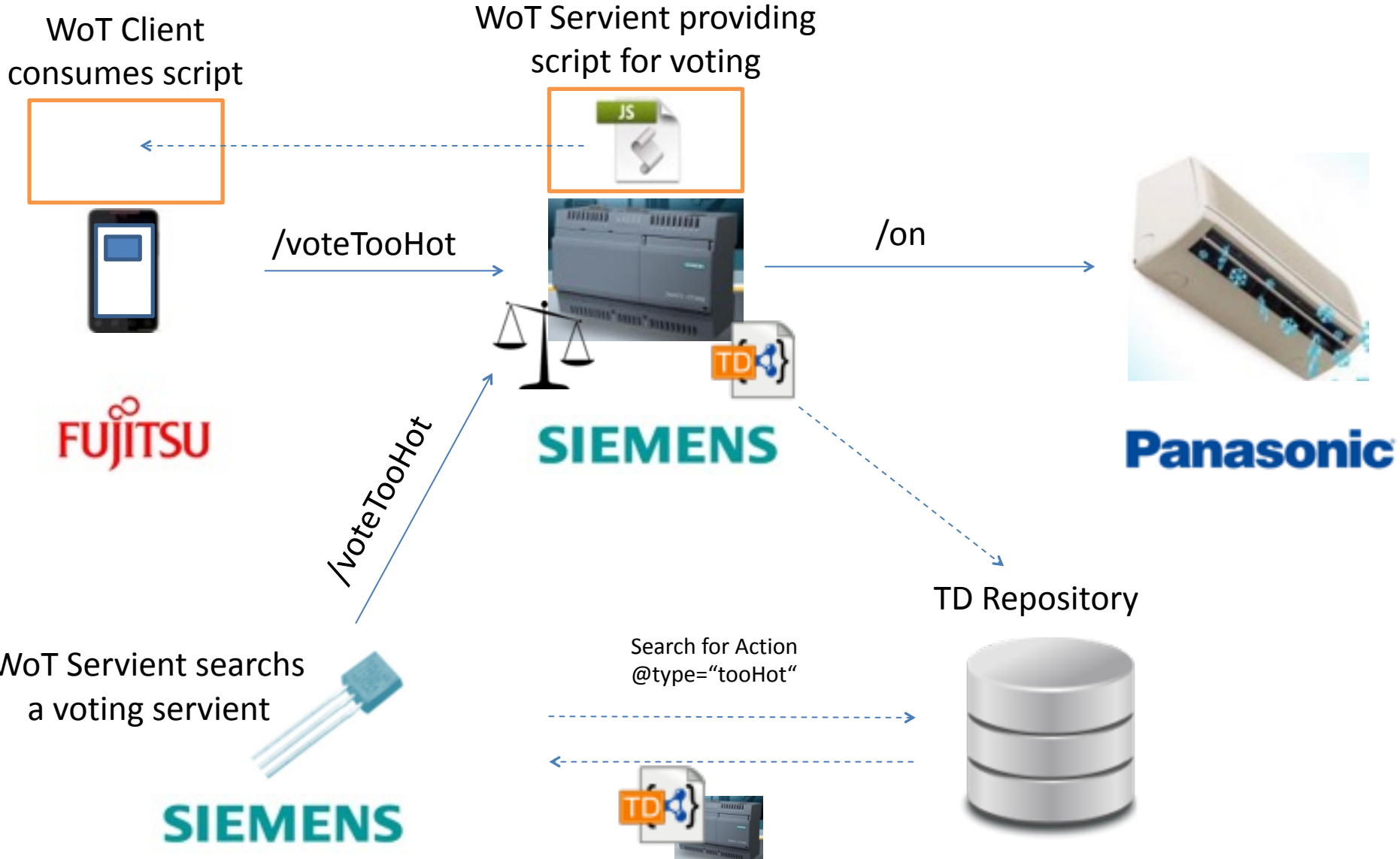
/on



**Panasonic**

/voteTooHot

# Scenario 2 – ‘Full WoT’



# Scenario 3 – ‘Mini Automation’

Consume brightness sensor  
to control curtain



**SIEMENS**



**FUJITSU**

# Online Resources

- Current Practices (Beijing Release)
  - <http://w3c.github.io/wot/current-practices/wot-practices-beijing-2016.html>
- Organization Wiki
  - [https://www.w3.org/WoT/IG/wiki/F2F\\_meeting,\\_July\\_2016,\\_China,\\_Beijing#PlugFest](https://www.w3.org/WoT/IG/wiki/F2F_meeting,_July_2016,_China,_Beijing#PlugFest)
- Test Cases
  - <https://github.com/w3c/wot/blob/master/plugfest/2016-beijing/plugfest-test-cases-beijing-2016.md>
- Report Template
  - <https://github.com/w3c/wot/blob/master/plugfest/2016-beijing/TestCaseCoverage.xlsx>  
(t.b.d.)