An introduction to

Web of Things Framework

Monday, 20 April 2015

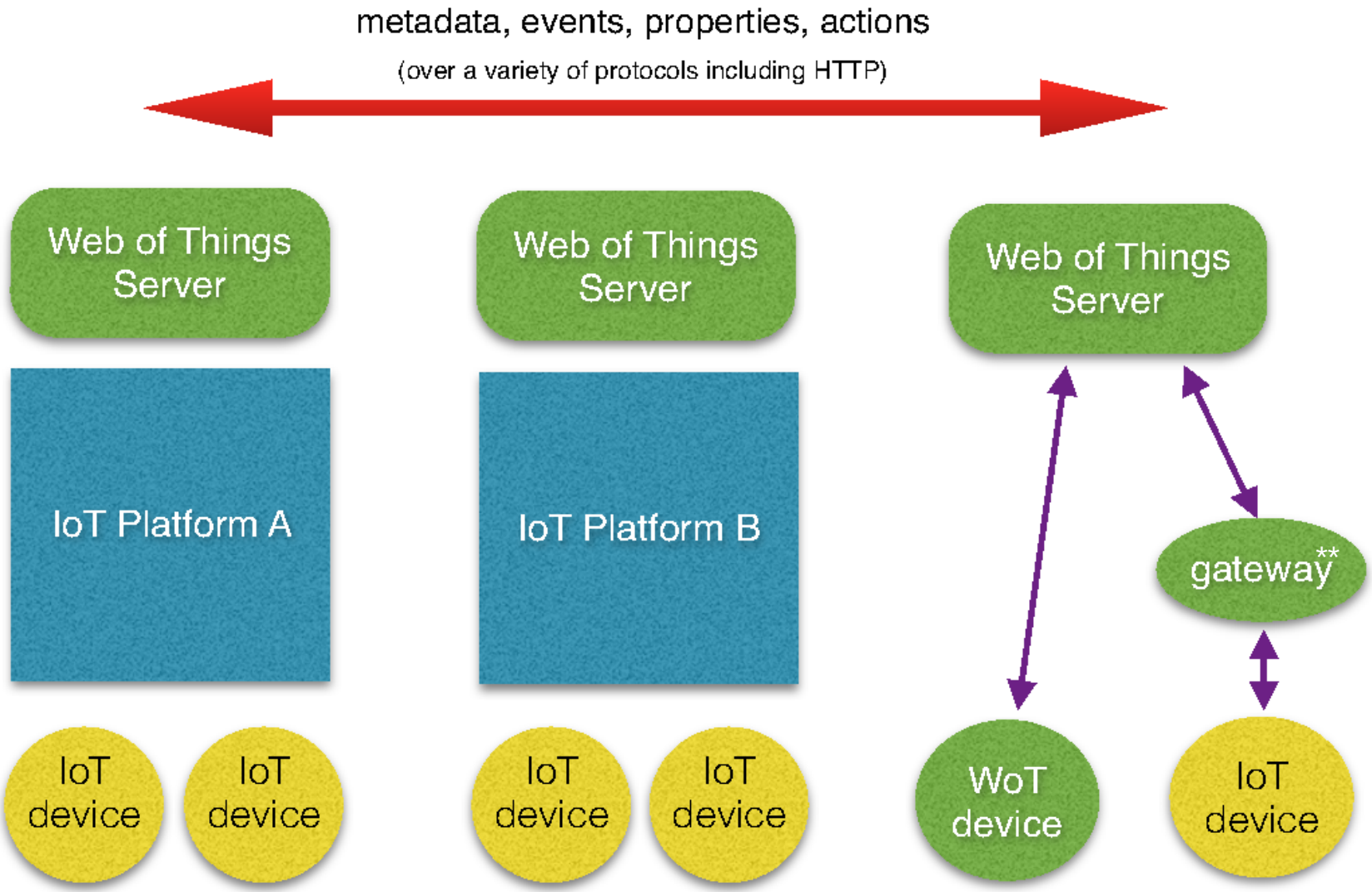
Munich, Germany

Dave Raggett, W3C

The Challenge

- We expect tens of billions of IoT devices within ten years
- But, the Internet of Things is beset with problems
  - Product **silos** that **don't interoperate** with each other
  - Plethora of approaches & **incompatible platforms**
  - Companies seeking to create and control ecosystems
    - Most will fail at this!
  - Locking in data **reduces the value** of the IoT overall
    - Blocks the network effect!
- This is painful for developers
  - Hard to keep track of who is doing what
  - Expensive to learn and port to different platforms
  - Challenging to create services that span domains and platforms

The Web as the Solution



** Gateway defines IoT device abstraction layer

From Pages to Things

- The web of pages is founded upon
  - IRIs for addressing
  - HTTP for access
  - HTML for pages and for discovery
    - Search engines following the links in pages
- Web of Things by analogy with web of pages
  - IRIs for addressing
  - HTTP and other protocols for access
    - No one protocol can satisfy all needs
  - Thing Description Language (TDL)
    - Semantics and data formats as basis for interoperability
    - Relationships to other things as basis for discovery

Web of Things Framework

- Expose IoT platforms and devices through the World Wide Web for a Web of Things
- “Things” as proxies for physical and abstract entities
- Modelled in terms of events, properties and actions
  - What events does this thing generate?
    - *Someone has just rung the door bell*
    - *Someone has just inserted a door key*
  - What properties does this thing have?
    - *Door is open or closed*
  - What actions can we invoke on this thing?
    - *Unlock the door*
  - Thing with on/off property as proxy for a light switch
- With bindings to scripting APIs and protocols

Web of Things Framework

- Standard way to retrieve “thing” descriptions
- Standard format for “thing” descriptions (e.g. JSON-LD)
- Owner, purpose, version, access control, terms & conditions, relationships to other things, security best practices, . . .
  - Giving data owners control over who can access their data and for what purposes – contract between consumer & supplier
- Semantics and data formats for events, properties & actions
- Properties have discrete values, or smoothly changing values that are interpolated between data points, e.g. for robotics
  - Clock sync across controllers: 1-10 mS with NTP, and microseconds with IEEE protocols
- Communication patterns
  - Push, pull, pub-sub, and peer to peer
- Bindings to a range of protocols
  - HTTP, Web Sockets, CoAP, MQTT, STOMP, XMPP, WebRTC

Interacting with a “Thing”

- Representational State Transfer (REST)
  - HTTP GET to retrieve a thing's description
  - HTTP GET to retrieve all properties of a thing
  - HTTP PUT to update all properties of a thing
  - HTTP PATCH to apply changes to some properties
  - HTTP POST to invoke actions on a thing
  - HTTP POST is also used to notify events
    - To proxies or dependent things
- REST can be used with other protocols
  - To send actions to thing within a firewall
  - To distribute updates via pub-sub model

Servers at many scales

Web of Things servers can be realised at many scales from microcontrollers to clouds

Micro-controller: resource constrained, IoT devices or gateways, CoAP, running behind firewall

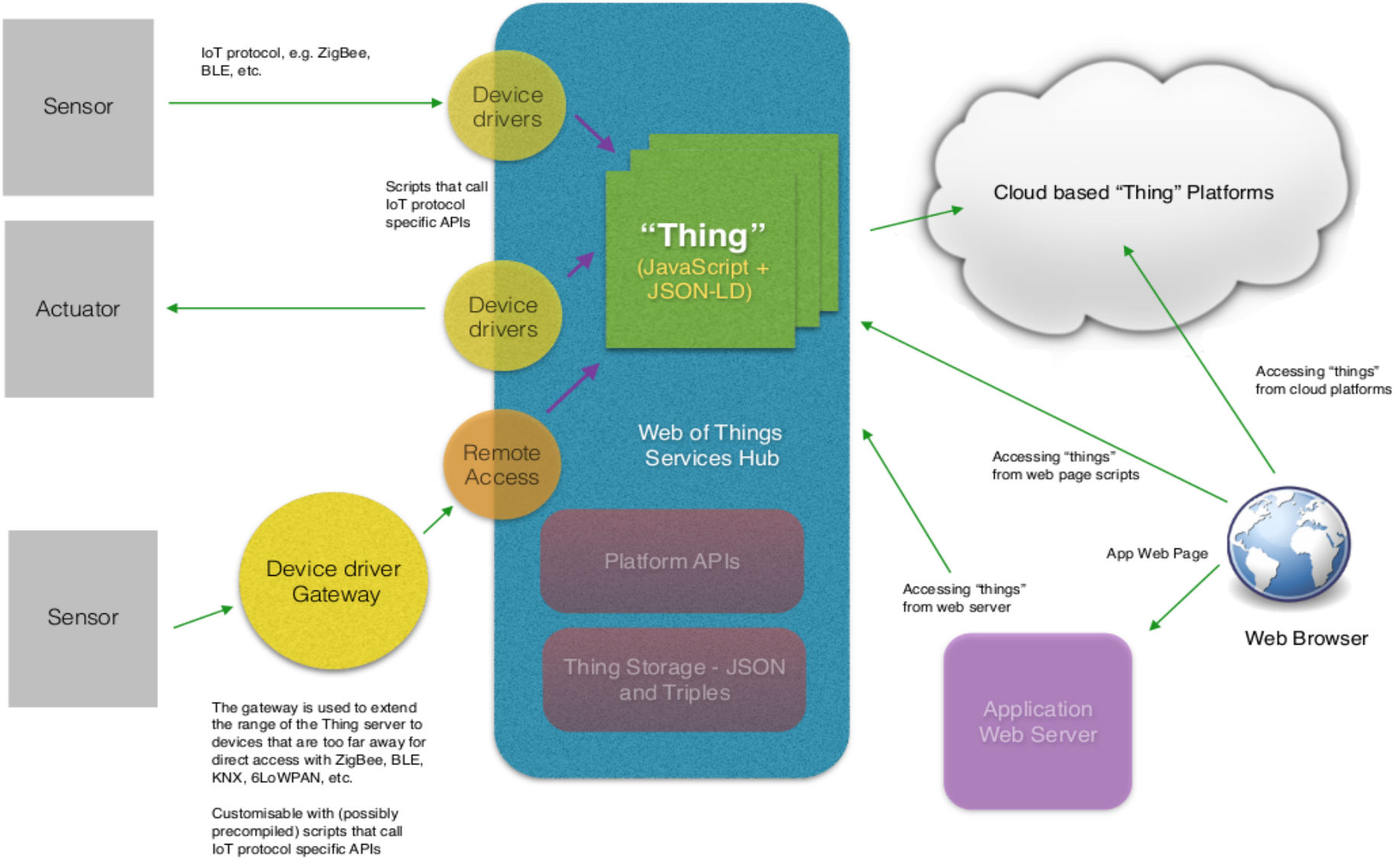
Home Hub: home/office server for access to smart home and wearables, running behind firewall

Smart Phone: personal server for access to smart home and wearables

Cloud-Based: highly scalable server for many users, devices and working with big data

Servers are free to choose which scripting languages they support
Could precompile service behaviour for constrained devices

Example of a Home Hub



Relationships between Things

- “Thing” description includes the relationships to the things that this thing depends upon
  - Server uses this to retrieve descriptions of related things as basis for deciding how to connect to them and expose them to scripts that define this thing's behaviour
  - Enables search engines to index the web of things
  - Supports richer search queries based upon relationships
  - Enables dependency management
    - Perhaps analogous with Linux package management
  - Decouples service behaviour from data protocols
  - Simpler expression of service behaviour via local names for things

End-User Service Creation

- Event-condition-action rules
  - Trigger action upon event if condition is true
  - High level events defined in terms of lower level events
  - Higher level actions defined in terms of lower level actions
    - Ordered and unordered sequences of actions
    - Pre- and Post-conditions
- Simple to use graphical editing tools
- Vocal commands (as with Apple's Siri)
  - *“turn the heating down when I leave home”*

Appeal of JSON-LD

- What makes JSON-LD attractive as basis for the thing description language?
- W3C Recommendation from 16 Jan 2014
  - http://www.w3.org/TR/json-ld/
- Combines simplicity of JSON with the power of the Linked Data and the Semantic Web
  - Out of band profiles and binary JSON formats for short packet protocols
- We would define a core profile for a vocabulary common to all “thing” descriptions
- Implementers would be encouraged to re-use vocabularies for specific application domains
  - These could be defined by industry specific groups
  - Need for better schema/vocabulary languages

Questions?

More details are given in: http://www.w3.org/2015/04/wot-framework.pdf

Thing Descriptions

- Door

```
{
   “@events” : {
      “bell”: null,
      “key”: {
         “valid” : “boolean”
      }
   },
   “@properties” : {
      “is_open” : “boolean”
   },
   “@actions” : {
      “unlock” : null
  }
}
```

- Light switch

```
{
   “@properties” : {
      “on” : {
         “value” : “boolean”,
         “writable” : true
      }
   },
}
```

TDL's default JSON-LD context defines bindings of core vocabulary to IRIs
Data models may be defined explicitly or by reference to an external definition

Questions for discussion:

How to define events in terms of property changes?
How to specify which protocols and encodings are supported?

Thing as Agent

- Thing description

```
{
   @context : {
      @base=”http://….
    },
    “@dependencies” : {
       “door” : “door12”,
       “light” : “switch12”
    }
}
```

- It's behaviour

```
// invoked when service starts

function start () {
    door.observe(“key”, unlock);
}

function unlock(key) {
    If (key.valid) {
        door.unlock();
        light.on = true;
    }
}
```

This “thing” is an agent with no events, properties or actions.

It unlocks the door and turns on the light when a valid key is presented.

n.b. @base defines a base IRI for resolving relative IRIs

Miscellany

- For validation and specification of vocabularies
  - JSON-Schema
  - RDF-Schema
  - OWL
- For efficient transfer of structured data
  - JSON *(defined by RFC7159, ECMA 404)*
    - MessagePack, Universal Binary JSON, etc.
  - Google's Protocol Buffers
  - XML with EXI
- Bindings to protocols need to cover encodings
  - **/.well-known/protocols** for retrieving server's protocol support?
- Actions on things are asynchronous and may return results

Thingsonomies

- The purpose of a “thing” can be defined formally in respect to an ontology
- The purpose can be defined informally using free text, e.g. as one or more tags chosen by the maintainer
- Co-occurrence of tags across many “things” performs an informal expression of semantics
  - In same way as folksonomies for images or blog posts

Thing Descriptions

- Thing descriptions may be static and shared by many “things”
  - These things can define their description by reference
- Some kinds of things may involve descriptions that change over time, e.g. a new owner, or a new physical location for a sensor, …
  - Events signalling changes to metadata?
  - Thing memories that record changes over a thing's lifetime
- Bindings to protocols may involve self tagged data
  - Analogous to “unions” in programming languages
- The properties of a “thing” may include data blobs that have a meaning and a content-type
  - Photo of someone and encoded as image/jpeg

Semantics for Smart Appliances

- Semantic Sensor Network Ontology
  - W3C SSN Incubator Group report
  - SSN Ontology
- Sensor Model Language (SensorML)
  - Developed by Open Geospatial Consortium
- Sensor Markup Language
  - JSON & XML/EXI – IETF draft-jennings-core-senml
- TNO's smart appliance ontology for ETSI M2M
  - Developed on behalf of European Commission

IETF CoRE WG

- CoRE WG with focus on resource oriented applications for constrained IP networks, and responsible for CoAP protocol
  - See tracker page and CoAP website
  - CoAP is based on REST and similar to HTTP
    - GET, PUT, POST, DELETE, OBSERVE methods
  - CoAP is a good fit for the Web of Things
- Resource discovery
  - Unicast or multicast queries
    - Link format (RFC6690) analogous to HTTP Link header
      - Which itself is modelled on HTML's LINK element
      - JSON link format under consideration
    - GET /.well-known/core returns list of resources
- Notifications with push and pub-sub
  - Interested parties register with GET
  - Notifications are sent with OBSERVE method