Web of Things

Easier, more accessible
descriptions for Web developers

Dave Raggett <dsr@w3.org>

Developer Feedback

- Much of the commercial innovation for the IoT is being done by start-ups and SMEs
- For a strong standard that is widely adopted by commercial innovators we need to involve them in the design choices
  - If we don't we risk producing a standard that isn't successful
- Outreach with tutorials, online testbeds, online surveys, direct contact and open source implementations
- Only advance to W3C Proposed REC when we see which approach gets widespread adoption
  - Let the market decide which serialisation should be a standard

Current Practices

- The current practices document describes an approach to thing descriptions based upon JSON-LD and JSON Schema
- The Plugfests have shown that it works, but there are drawbacks
  - Relatively complex and hard to understand compared to conventional JSON
  - JSON Schema isn't yet a formal standard that we can normatively reference from W3C Recommendations
  - Unclear relationship between JSON Schema and Linked Data
- This perceived complexity is likely to deter widespread adoption and defer realising the full potential of the Web of Things!
  - As well as making it harder to attract new Members to WoT WG
- Why not use simple JSON with regular mapping to Linked Data?

Mozilla Web Things

- Simple, obvious use of JSON for thing descriptions
- Properties, actions, events and links
- Name, type, unit, description and href
- But unclear relation to Linked Data and semantics
- Only designed for REST

See: http://iot.mozilla.org/wot/

```
{
        "name": "WoT Pi",
        "type": "thing",
        "description": "A WoT-connected Raspberry Pi",
        "properties": {
                "temperature": {
                        "type": "number",
                        "unit": "celsius",
                        "description": "An ambient temperature sensor",
                        "href": "/things/pi/properties/temperature”
                }
        },
        "actions": {
                "reboot": {
                        "description": "Reboot the device”
                }
        },
        "events": {
                "reboot": {
                         "description": "Going down for reboot”
                }
        },
        "links": {
                "properties": "/thing/pi/properties",
                "actions": "/things/pi/actions",
                "events": "/things/pi/events",
                "websocket": "wss://mywebthingserver.com/things/pi”
        }
}
```

My proposal

- Simple obvious JSON, very similar to Mozilla's proposal
  - Proof of concept server implemented with NodeJS
- Rich set of data types + app defined types
- Regular mapping to Linked Data using @context
- Distinction between interaction models and semantic models
  - Interaction model: properties, actions and events
  - Semantic model: the kinds of things and their relationships as a basis for discovery, composition, validation and adaptation
- Applicable to broad range of IoT standards, protocols and communication patterns – not restricted to REST
  - Tested against device definitions for OCF, oneM2M and preliminary study of ECHOnet and BACnet
  - Further work planned for Bluetooth and LwM2M

Formal Basis as Linked Data

Type system defined in Linked Data for a serialization format agnostic approach

- Properties, actions, events and metadata
- Core types
  - Boolean, Number, Integer, String, Enumeration
- Compound types
  - Object, Collection, Vector, Thing
- Unions of types for run-time flexibility
- Basic constraints
  - Ranges, Patterns, Read-only vs writeable, optional vs required
- Application defined named types
  - To improve clarity and encourage re-use
- More details at
  - https://github.com/w3c/wot/blob/master/proposals/dsr-td/td-draft-spec-dsr.md

Mapping JSON to Linked Data

- JSON object property names treated as either predicates or names according to whether the depth of JSON object nesting is odd or even
- Predicates mapped to RDF URIs via @context
  - No need for @context if default context is sufficient
- Enumerations represented as JSON Arrays
- Short cut when you only need to state property type
- JSON object property reserved terms
  - “types” reserved for application defined types
  - “properties” reserved for properties
  - “actions” reserved for actions
  - “events” reserved for events
- Names must not start with “@”
  - Enables properties to be directly mapped to object properties using getters and setters
- More details at
  - https://github.com/w3c/wot/blob/master/proposals/dsr-td/json-to-ld-dsr.md

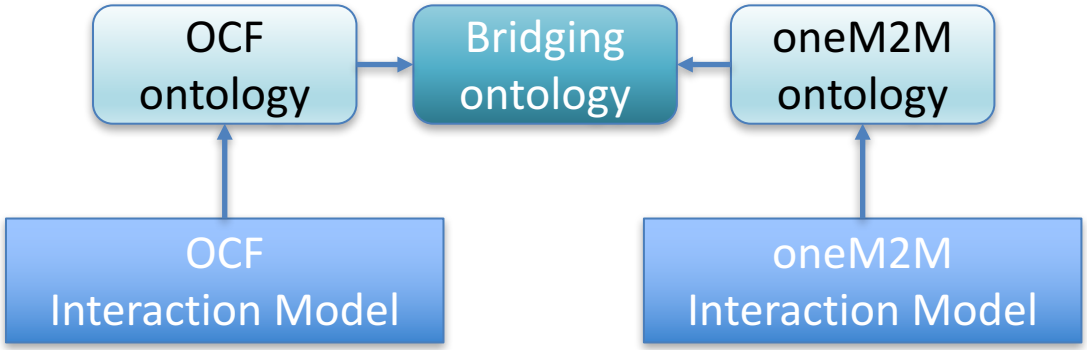
Benefits of Semantic Models

- Interaction models describe the properties, actions and events exposed to applications
- You don’t know what the properties mean, except perhaps from the name and human language description
- Semantic models describe the meaning and relationships between things
- This allows providers and consumers to agree on the meaning for the data that they exchange
- Enabling discovery based upon the purpose of things, e.g. air conditioner, light, light switch, and the context in which they occur
- Designing compositions of services with the confidence that they will work as intended
- Adapting the user interface and service logic to deal with variations in capabilities from one device vendor/standard suite to another

Semantic Models

- Each thing is an instance of the class of things
- Things may be declared as instances of other more specific kinds of things using *rdf:type* predicate, or as a subclass with *rdfs:subClassOf*
- Ditto for properties, actions and events
- Syntactic compositions as a collection of interfaces describing the interaction model
  - Common feature of IoT standards suites such as OCF and oneM2M
    - Proposed td:import predicate for importing an RDF graph
- Semantic compositions as a collection of capabilities describing what a thing does
  - Annotating interaction models with the roles that properties, actions events play for each capability
    - Need to explore and evaluate alternative approaches for this

Bridging Communities



- Different vendors and IoT standards groups will inevitably choose different capabilities for similar appliances
  - Home appliances: OCF, oneM2M, ECHOnet
    - Motion detector, air conditioner, refrigerator, etc.
  - Vendors want to differentiate their products from their competitors
- Bridging ontologies can help applications to adapt to differences
  - Discovery using generic terms rather than terms specific to particular IoT standards suites
  - Contrast this with “upper ontologies” which define underlying dimensions etc.

Exploring the Design Space

- I have already generated Linked Data interaction models for OCF (OIC 1.1) and oneM2M (HAIM)
- I am in the process of doing this for ECHOnet and plan to follow on with Bluetooth and LwM2M
- The next step will be to craft the corresponding semantic models for each IoT standard suite
- This will pave the way to crafting models that bridge the semantic models for different IoT suites
  - Also important as a basis for discovery across different standards suites
- I then want to experiment with JavaScript based smart applications that can adapt both the user interface and the service logic to deal with variations across devices defined for different IoT suites
  - Proof of concept implementation using NodeJS servient with drivers for range of IoT suites
  - Building upon my JavaScript library for operating on RDF graphs
- Related work on abstract things as smart compositions of services

Protocol Bindings

- Assume that each server will have one or more drivers for the IoT standards and protocols it supports
- Each driver recognises specific metadata
  - Some metadata is applicable across drivers
- OCF specific driver vs generic REST driver
  - Former requires less metadata than latter
  - Only a few drivers are needed for each server
  - Drivers could be installed upon demand
- Driver metadata to be defined in collaboration with the corresponding standards development organisations
  - OCF, oneM2M, ECHONet, Bluetooth SIG, OPC Foundation, …

Miscellaneous Metadata

- Anticipate the need for standardised metadata for such matters as
  - Security, access control, privacy, safety and trust
  - Service level agreements including payments
  - Automated negotiation of terms and conditions
- Domain specific metadata
  - e.g. plan for WoT CG to work on vocabularies for smart homes and mobility

WEB OF THINGS

Demo