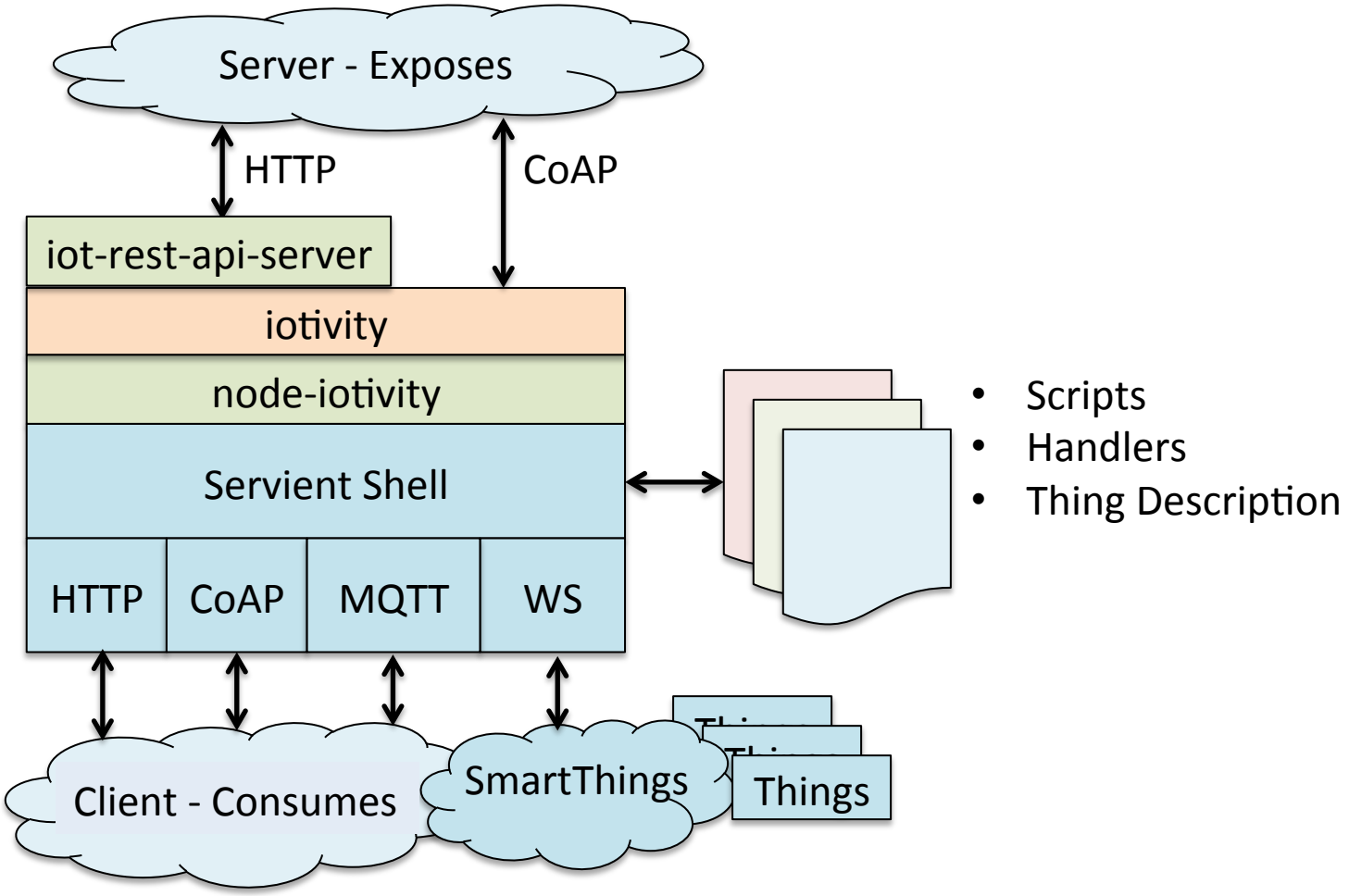
WoT Servient using SmartThings and Iotivity

May 23, 2016

WoT Servient using SmartThings and Iotivity

- Expose SmartThings capabilities as OCF resource types with WoT Thing Description
- HTTP, CoAP, and MQTT Protocol Bindings
- Uses iotivity as the resource layer
- node-iotivity: existing nodejs API and binding
- iot-rest-api-server: existing HTTP proxy for iotivity
- New nodejs based servient shell provides a client + server scripting API and abstract transfer layer

SmartThings + Iotivity WoT Servient



Some Questions

- What does a SmartThings TD look like?
- What does a SmartThings/OCF Resource Type look like?
- Can iotivity expose Thing Descriptions?
- How does the action collection pattern work?
- How does the subscription pattern work?

Mapping SmartThings Capabilities

- W3C WoT Thing Descriptions
  - WoT Thing models a SmartThings Thing, at the device level
  - Capability-level granularity would be better for applications and security
- OCF Resource Types and Collections
  - Attributes and Commands mapped using the WoT Interaction model and abstract transfer binding
  - https://github.com/hyperstate/hyperstate-docs/blob/master/wot-18may.pdf
  - Resource Directory + device bridge architecture

Modeling SmartThings Capabilities

- Use oneiota to create a canonical model for an abstract SmartThings Capability with attribute and command type mappings to read/write/ observe properties and actionInstance collections
- It should be fairly easy to create the capability types by customizing the abstract model for capability types
- Generate WoT Thing Description templates automatically from Capability Models
- Can we bypass oneiota in the chain and construct resource instances from TD instances?

Use case for SmartThings Models

- IoTivity SmartThings Bridge
- A SmartThing is a collection of Capabilities
- Each Capability has a model
- Discover SmartThings Things and construct IoTivity resource instances from capabilities
- How does oneiota help?