Introduction to Resource-Oriented Applications in Constrained Networks

Zach Shelby

Tutorial Overview

• Powering M2M with a Web of Things
• So What are Web Services?
• CoRE - Constrained RESTful Environments
  – Constrained Application Protocol Basics
  – Observation
  – Block-transfer
  – Discovery
• Semantic Soup
• Further Information
The Web of Things

What are Web Services?
The Web Architecture

Web Resource Identification

Universal Resource Name (URN)

Universal Resource Identifier (URI)

Universal Resource Locator (URL)
The Web Service Paradigm

A REST Request
An HTTP Request

![HTTP Request Diagram](image)

See RFC2616 - Hypertext Transfer Protocol v1.1

CoRE - Constrained RESTful Environments
CoRE Requirements

See draft-shelby-core-coap-req

The CoRE Architecture
The Constrained Application Protocol

- Embedded web transfer protocol (coap://)
- Asynchronous transaction model
- UDP binding with reliability and multicast support
- GET, POST, PUT, DELETE methods
- URI support
- Small, simple header < 10 bytes
- Subset of MIME types and HTTP-compatible response codes
- Optional observation, block transfer and discovery

What CoAP is (and is not)

- CoAP is
  - A RESTful protocol
  - Both synchronous and asynchronous
  - For constrained devices and networks
  - Specialized for M2M applications
  - Easy to proxy to/from HTTP
- CoAP is not
  - A replacement for HTTP
  - General HTTP compression
  - Separate from the web
The Transaction Model

• Transport
  – CoAP is defined for UDP

• Messaging
  – Simple message exchange between end-points
    – CON, NON, ACK, RST

• REST
  – Request/Response piggybacked on messages
  – Method, Response Code and Options (URI, content-type etc.)

Message Header

Ver - Version (1)
T - Transaction Type (Confirmable, Non-Confirmable, Acknowledgement, Reset)
OC - Option Count, number of options after this header
Code - Request Method (1-10) or Response Code (40-255)
Message ID - Identifier for matching responses
Option Header

<table>
<thead>
<tr>
<th>option delta</th>
<th>length</th>
<th>for 0..14</th>
</tr>
</thead>
</table>

| option delta | 1 1 1 1 | length - 15 |

Option Delta - Difference between this option type and the previous

Length - Length of the option value (0-270)

Value - The value of Length bytes immediately follows Length

Options

<table>
<thead>
<tr>
<th>No.</th>
<th>C/E</th>
<th>Name</th>
<th>Format</th>
<th>Length</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Critical</td>
<td>Content-Type</td>
<td>uint</td>
<td>1-2 B</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>Elective</td>
<td>Max-Age</td>
<td>uint</td>
<td>0-4 B</td>
<td>60</td>
</tr>
<tr>
<td>3</td>
<td>Critical</td>
<td>Proxy-Uri</td>
<td>string</td>
<td>1-270 B</td>
<td>(none)</td>
</tr>
<tr>
<td>4</td>
<td>Elective</td>
<td>ETag</td>
<td>opaque</td>
<td>1-8 B</td>
<td>(none)</td>
</tr>
<tr>
<td>5</td>
<td>Critical</td>
<td>Uri-Host</td>
<td>string</td>
<td>1-270 B</td>
<td>(see below)</td>
</tr>
<tr>
<td>6</td>
<td>Elective</td>
<td>Location-Path</td>
<td>string</td>
<td>1-270 B</td>
<td>(none)</td>
</tr>
<tr>
<td>7</td>
<td>Critical</td>
<td>Uri-Port</td>
<td>uint</td>
<td>0-2 B</td>
<td>(see below)</td>
</tr>
<tr>
<td>8</td>
<td>Elective</td>
<td>Location-Query</td>
<td>string</td>
<td>1-270 B</td>
<td>(none)</td>
</tr>
<tr>
<td>9</td>
<td>Critical</td>
<td>Uri-Path</td>
<td>string</td>
<td>1-270 B</td>
<td>(none)</td>
</tr>
<tr>
<td>11</td>
<td>Critical</td>
<td>Token</td>
<td>opaque</td>
<td>1-8 B</td>
<td>(empty)</td>
</tr>
<tr>
<td>15</td>
<td>Critical</td>
<td>Uri-Query</td>
<td>string</td>
<td>1-270 B</td>
<td>(none)</td>
</tr>
</tbody>
</table>
### Request Examples

- **Confirmable Request**
  - CoAP Client: `CON [0x1a5] GET /light`
  - CoAP Server:
  - Piggy-backed Response: `ACK [0x1a5] 2.05 Content "<light>..."

- **Non-confirmable Request**
  - CoAP Client: `NON [0x1a5] POST /reading`
  - CoAP Server: 

### Dealing with Packet Loss

- **CoAP Client**
  - `CON [0x1a] GET /humidity`
  - Timeout

- **CoAP Server**
  - `CON [0x1a] GET /humidity`
Normal Response

Bits and bytes...
Caching

- CoAP includes a simple caching model
  - Cacheability determined by response code
- Freshness model
  - Max-Age option indicates cache lifetime
- Validation model
  - Validity checked using the Etag Option
- A proxy often supports caching
  - Usually on behalf of a sleeping node,
  - and to reduce network load

Proxying and caching
Observation

Block transfer

See draft-ietf-core-observe

See draft-ietf-core-block
Resource Discovery

• Service Discovery
  – Leave this to e.g. DNS-SD
• Resource Discovery with CoRE Link Format
  – Web linking as per RFC5988
  – Discovering the links hosted by CoAP servers
  – GET /.well-known/core
  – Returns a link-header style format
    • URL, relation, type, interface, content-type etc.
• See draft-ietf-core-link-format

Resource Discovery

CoAP Client

CON [0x6f6] GET /.well-known/core

ACK [0x7f6] 2.05 Content "<light>..."

CoAP Server

</light>;rt="Illuminance";ct=0,
</s/maastr.xml>;title="Maastricht weather";ct=1,
</s/maastr/temp>;title="Temperature in Maastricht";ct=1,
</s/oulu.xml>;title="Oulu weather";ct=1,
</s/oulu/temp>;title="Temperature in Oulu";ct=1,
</s/temp>;rt="Temperature";ct=0
Semantic Soup

• So how to use CoRE in real applications?
• Resources need meaningful naming (rt=)
• A resource needs an interface (if=)
  – See draft-vial-core-link-format-wadl
• A payload needs a format (EXI, JSON etc.)
• Deployment or industry specific today
  – oBIX, SensorML, EEML, sMAP etc.
• What can we make universal?
• What should be market specific?
• How do we enable innovation?

Further Information

• draft-ietf-core-coap
• draft-ietf-core-block
• draft-ietf-core-observe
• draft-ietf-core-link-format
• RFC5988 – Web Linking
• Ongoing work in the CoRE WG
  – Security bootstrapping, resource directory, group communications, congestion control etc.