Implementation of CoAP and its Application in Transport Logistics

Koojana Kuladinithi, Olaf Bergmann, Thomas Pötsch, Markus Becker, Carmelita Görg

koo@comnets.uni-bremen.de
bergmann@tzi.org
tpoetsch@uni-bremen.de
mab@comnets.uni-bremen.de
cg@comnets.uni-bremen.de

TZI, University Bremen, Germany

IP+SN2011, 11th of April 2011
Outline

Introduction

Machine-2-Machine Communication

Implementation of CoAP libcoap
  libcoap & Contiki
  libcoap & TinyOS
  Evaluation of libcoap
  CoAP vs HTTP

Conclusions & Outlook
Introduction

- IETF CoRE Working Group
- Constrained Application Protocol (CoAP)
- RESTful protocol
CoAP in a nutshell

- Methods: CON, NON, ACK, RST
- Pre-processed URI in different options
- Resource discovery built-in / .well-known/core
- Caching/Proxying
- HTTP like response codes
- Mapping to HTTP
Machine-2-Machine Communication

▶ http://www.intelligentcontainer.com
## CoAP Resources (1/2)

<table>
<thead>
<tr>
<th>Resource</th>
<th>GET</th>
<th>PUT</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>/st</td>
<td>X</td>
<td></td>
<td>Temperature</td>
</tr>
<tr>
<td>/sh</td>
<td>X</td>
<td></td>
<td>Humidity</td>
</tr>
<tr>
<td>/sv</td>
<td>X</td>
<td></td>
<td>Voltage</td>
</tr>
<tr>
<td>/r</td>
<td>X</td>
<td></td>
<td>Temperature, humidity and voltage together</td>
</tr>
<tr>
<td>/l</td>
<td>X</td>
<td>X</td>
<td>LEDs</td>
</tr>
<tr>
<td>/ck</td>
<td>(X)</td>
<td>X</td>
<td>AES Encryption Key</td>
</tr>
</tbody>
</table>

**Table:** CoAP Resources on Sensor Nodes
CoAP Resources (2/2)

<table>
<thead>
<tr>
<th>Resource</th>
<th>GET</th>
<th>PUT</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>/ni</td>
<td></td>
<td>X</td>
<td>Inform about node integration into 6LoWPAN/RPL network</td>
</tr>
<tr>
<td>/warntemplow</td>
<td></td>
<td></td>
<td>Below Warning</td>
</tr>
<tr>
<td>/warntenphi</td>
<td>X</td>
<td></td>
<td>Above Warning</td>
</tr>
</tbody>
</table>

Table: CoAP Resources on the FSU
libcoap

- [http://libcoap.sourceforge.net](http://libcoap.sourceforge.net)
- Implements
  - draft-ietf-core-coap-03
  - draft-ietf-core-link-format-01
  - draft-ietf-core-block-00
  - draft-ietf-core-observe-00
  - draft-bormann-coap-misc-06
- Provides
  - Sample server
  - Sample client
- Participated in several plug-fests of the CoRE working group
libcoap & Contiki

- CoAP for Sensinode N740
- Based on uIP
- Contiki & libcoap stripped to fit into 3 memory banks of 32 KB each
- TCP removed
- Shared global variables
- libcoap: ~ 12 KB ROM
- rest-coap shipped with Contiki: ~ 26 KB ROM
libcoap & TinyOS

- Installation instructions at http://docs.tinyos.net/index.php/CoAP
- CoAP server and client
- Only GET + PUT support
  (POST + DELETE do not fit component model)
- Based on TinyOS blip-1.0
  (working on blip-2.0)
- Block & observe drafts not supported yet (time.h \(\rightarrow\) Timer)
- Synchronous/asynchronous a.k.a immediate/deferred
  a.k.a. piggy-backed/separate supported through TinyOS timers
- multiple end-points on different ports possible
libcoap & TinyOS components
LibCoAPAdapterC translates between callbacks and call/signal
LibCoAPAdapterC wired to UDPSocketC’s
Resources wired, but registered with URI at boot time
Generic index calculated from URI
Interfaces (1/3)

interface LibCoAP {
    command error_t bind(uint16_t port);
    command coap_tid_t send(coap_context_t *ctx,
                            struct sockaddr_in6 *dst,
                            coap_pdu_t *pdu,
                            int free_pdu);
    event void read(struct sockaddr_in6 *from,
                    void *data,
                    uint16_t len,
                    struct ip_metadata *meta);
}

Becker: CoAP, IP+SN 2011
Interfaces (2/3)

interface ReadResource {
    command error_t get(coap_tid_t id);
    event void getDone(error_t result,
        coap_tid_t id,
        uint8_t asyn_message,
        uint8_t* val,
        uint8_t buflen);
    event void getPreAck(coap_tid_t id);
}

- E.g. for Temperature, Humidity, Voltage
- Default handlers catch not supported methods on resources and return the appropriate error code
Interfaces (3/3)

interface WriteResource {
    command error_t put(uint8_t* val, uint8_t buflen);
    event void putDone(error_t result);
}

- Readable and writable resources implement both interfaces
- E.g. for Led, FlashStorage
CoapReadResourceC translates between `val_t` and `uint8_t`.

CoAPBuffer{Volt|Hum|Temp}TranslateC transform buffer reading to SI units using fixed-point calculations.

Over-the-air only fixed point values (SI value * 100) are transmitted.

The corresponding side does not need to know which sensor and its characteristic line.

Characteristic line might be discoverable by resource discovery and resource description of link-format standard.
## Evaluation of libcoap (Time & Size)

<table>
<thead>
<tr>
<th>Resource</th>
<th>Type</th>
<th>Retrieval Time</th>
<th>Num. of Bytes Transmitted</th>
</tr>
</thead>
<tbody>
<tr>
<td>/st</td>
<td>GET</td>
<td>297.04 ms</td>
<td>223 bytes</td>
</tr>
<tr>
<td>/sh</td>
<td>GET</td>
<td>143.57 ms</td>
<td>119 bytes</td>
</tr>
<tr>
<td>/sv</td>
<td>GET</td>
<td>92.69 ms</td>
<td>119 bytes</td>
</tr>
<tr>
<td>/r</td>
<td>GET</td>
<td>369.99 ms</td>
<td>229 bytes</td>
</tr>
<tr>
<td>/l</td>
<td>GET</td>
<td>69.55 ms</td>
<td>117 bytes</td>
</tr>
<tr>
<td>/l</td>
<td>PUT</td>
<td>71.12 ms</td>
<td>116 bytes</td>
</tr>
<tr>
<td>/ck</td>
<td>PUT</td>
<td>101.51 ms</td>
<td>142 bytes</td>
</tr>
</tbody>
</table>

- GET /st and /r are deferred/asynchronous/separate → higher retrieval time and number of transmitted bytes
- With coap-04 not necessary to be deferred, RESPONSE_TIMEOUT now 2 s
Evaluation of **libcoap** (ROM)

- **UDPEcho** - CoAP without resources: 3708 Byte
- Increase to CoAP with resources is mostly because of additional components
CoAP vs HTTP (RTT)

- Test on GPRS, not 802.15.4, because of
  - HTTP not available on 802.15.4
  - CoAP wasn’t available for TinyOS at that time
  - CoAP might be of interest for M2M on GPRS as well
  - Similar RTT for multi-hop 6LoWPAN

<table>
<thead>
<tr>
<th>Access Method</th>
<th>Time (sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Apache2-Firefox</td>
<td>38.774</td>
</tr>
<tr>
<td>Apache2-Epiphany</td>
<td>31.972</td>
</tr>
<tr>
<td>Apache2-wget</td>
<td>2.660</td>
</tr>
<tr>
<td>Apache2-bareHTTP client</td>
<td>3.032</td>
</tr>
<tr>
<td>bareHTTP(TCP)</td>
<td>3.076</td>
</tr>
<tr>
<td>bareHTTP(UDP)</td>
<td>1.104</td>
</tr>
<tr>
<td>CoAP</td>
<td>1.029</td>
</tr>
</tbody>
</table>
CoAP vs HTTP (#Bytes)

- Firefox downloads favicon
- Firefox, Epiphany, and wget add user agents
- Apache2 adds Content-Type etc.
- Bare server and clients are less chatty
- UDP reduces a lot
- CoAP has retransmission
CoAP vs HTTP (#Bytes)

<table>
<thead>
<tr>
<th>Header</th>
<th>HTTP/TCP</th>
<th>HTTP/UDP</th>
<th>CoAP/UDP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Link Layer</td>
<td>160</td>
<td>32</td>
<td>32</td>
</tr>
<tr>
<td>IP</td>
<td>200</td>
<td>40</td>
<td>40</td>
</tr>
<tr>
<td>TCP/UDP</td>
<td>340</td>
<td>16</td>
<td>16</td>
</tr>
<tr>
<td>HTTP/CoAP</td>
<td>181</td>
<td>41</td>
<td>17</td>
</tr>
<tr>
<td>Data</td>
<td>4</td>
<td>4</td>
<td>2</td>
</tr>
</tbody>
</table>

Table: Separation of Bytes at each Layer

- Note: IP without 6LoWPAN
Conclusions

- CoAP for M2M enabled logistic applications
- `libcoap` for Contiki and TinyOS
- `libcoap` is a generic library, specific custom-made implementations might be smaller
- Adaptations necessary, e.g. split-phase operation
- More compact than HTTP/TCP, but reliable on UDP
- JNI interface to `libcoap` for Java available, and possibly other languages
Outlook

- Measurement of CoAP in deployments
- Port to blip-2.0
- Move to coap-05/06
- Simulation with TOSSIM