Performance of the Constrained Application Protocol for Wireless Sensor Networks

ITG Fachgruppen 5.2.1/5.2.4

Thomas Pötsch
thp@comnets.uni-bremen.de

University Bremen, Germany

29th of September 2011
Outline

Introduction

CoAP Implementation in TinyOS

Evaluation of CoAP in TinyOS
  Measurements and Simulations
    Single-Hop
    Multi-Hop
  Comparison of CoAP and HTTP
  Memory Usage on TelosB Nodes

Conclusions & Outlook
CoAP - Constrained Application Protocol

- CoAP is a RESTful transfer protocol for resource-constrained networks and nodes in embedded and wireless sensor systems

- Lightweight application-protocol, similar to HTTP:
  - Client/Server model
  - Methods: GET, PUT, POST and DELETE
  - Resources identified by Uniform Resource Identifier (URI)
  - Message types: CON, NON, ACK, RST
  - Response includes response code (e.g. 404 / Not Found)
  - Built-in resource discovery /.well-known/core
  - Exponential back-off
CoAP - Main Features

- Low header overhead and complexity
- Constrained web protocol fulfilling M2M requirements
- Asynchronous message exchange (Deferred)
- URI (Uniform Resource Identifier) and content-type support
- Mapping to HTTP by use of proxies
CoAP Implementation **libcoap**

- **libcoap** is a C-Implementation of CoAP by Olaf Bergmann (TZI)
- Currently based on draft-ietf-core-coap-03, October 2010, development ongoing on coap-07
- Implements
  - draft-ietf-core-link-format-01
  - draft-ietf-core-block-00
  - draft-ietf-core-observe-00
  - draft-bormann-coap-misc-06
- Including
  - Sample server and client
- Available at: [http://libcoap.sourceforge.net](http://libcoap.sourceforge.net)
Implementation of CoAP in TinyOS

CoAP implementation on TelosB nodes using `libcoap`

- Based on TinyOS blip-1.0, later ported to blip-2.0
- Flexible configuration of Resources, UDP port and IPv6 address (Makefile)
- Multiple end-points on different ports possible
- Generic index calculation from URI for wiring of components
- Registration of URIs at boot time
TinyOS CoAP Server and Client

- **CoAP Server**
  - Asynchronous message exchange (Token, Message ID)
  - Full GET + PUT support for numerous resources (POST + DELETE do not fit in TinyOS component model)
  - Read- and Write-Interface with fixed-point sensor value calculation
  - Binary content type support
  - Several resources are implemented

- **CoAP Client**
  - Resource /ni for node integration and /ri for route debugging
Deployment of CoAP in the Intelligent Container

Option 4: End-to-end CoAP

CoAP

Payload
CoAP
UDP
IP
6LoWPAN
Constrained Links

802.15.4 based WSN

Sensor Node

6LOWPAN edge router

Cellular/Satellite/WLAN

Freight Supervision Unit (FSU)

Telematic

Internet

Option 1: Proprietary, Option 2: CoAP

Option 3: HTTP

Payload
CoAP/HTTP
UDP/TCP
IP
Links

Backend Software
## CoAP Resources

<table>
<thead>
<tr>
<th>Resource</th>
<th>Deferred</th>
<th>GET</th>
<th>PUT</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>/st</td>
<td>X</td>
<td>X</td>
<td></td>
<td>Temperature</td>
</tr>
<tr>
<td>/sh</td>
<td>X</td>
<td></td>
<td></td>
<td>Humidity</td>
</tr>
<tr>
<td>/sv</td>
<td>X</td>
<td></td>
<td></td>
<td>Voltage</td>
</tr>
<tr>
<td>/r</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>Temperature, Humidity and Voltage LEDs</td>
</tr>
<tr>
<td>/l</td>
<td>X</td>
<td></td>
<td>X</td>
<td>AES Encryption Key</td>
</tr>
<tr>
<td>/ck</td>
<td>(X)</td>
<td></td>
<td>X</td>
<td>Routing Table (blip-2.0 only)</td>
</tr>
<tr>
<td>/ni</td>
<td></td>
<td>X</td>
<td></td>
<td>Inform about node integration into 6LoWPAN network</td>
</tr>
<tr>
<td>/ri</td>
<td></td>
<td></td>
<td>X</td>
<td>Inform about node’s Routing Table (blip-2.0 only)</td>
</tr>
</tbody>
</table>
Performance of the Constrained Application Protocol for Wireless Sensor Networks

CoAP Implementation at a Glance

For further information:

http://docs.tinyos.net/tinywiki/index.php/CoAP
Measurements and Simulations

- Based on blip-2.0 using RPL
- 100 independent measurements
- Request interval 2 seconds
Measurements and Simulations

- Based on blip-2.0 using RPL
- 100 independent measurements
- Request interval 2 seconds

Measurements:

- Channel 12 (interfered) and 26 (free)
- Values captured with Wireshark
- Message sizes are without 6LoWPAN compression
Measurements and Simulations

- Based on blip-2.0 using RPL
- 100 independent measurements
- Request interval 2 seconds

Measurements:
- Channel 12 (interfered) and 26 (free)
- Values captured with Wireshark
- Message sizes are without 6LoWPAN compression

Simulations:
- Ideal channel conditions are assumed
- Values captured via debug messages
- LED resource / only
Performance of the Constrained Application Protocol for Wireless Sensor Networks

Measurements - Single-Hop

CoapBlip

6LoWPAN

PppRouter

Client
# Performance of the Constrained Application Protocol for Wireless Sensor Networks

## Measurements - Single-Hop

![Diagram showing CoapBlip, 6LoWPAN, and PppRouter connected to a Client via PPP and Socket Interface]

<table>
<thead>
<tr>
<th>Resource</th>
<th>Deferred</th>
<th>Method</th>
<th>Mean RTT $\bar{x}$ (Standard Deviation s)</th>
<th># of bytes</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>Ch. 26</td>
<td>Ch. 12</td>
</tr>
<tr>
<td>/st</td>
<td>X</td>
<td>GET</td>
<td>295.64 ms (2.4 ms)</td>
<td>302.38 ms (12.1 ms)</td>
</tr>
<tr>
<td>/sh</td>
<td></td>
<td>GET</td>
<td>133.07 ms (4.2 ms)</td>
<td>141.72 ms (10.4 ms)</td>
</tr>
<tr>
<td>/sv</td>
<td></td>
<td>GET</td>
<td>74.59 ms (4.4 ms)</td>
<td>83.55 ms (10.7 ms)</td>
</tr>
<tr>
<td>/r</td>
<td>X</td>
<td>GET</td>
<td>363.39 ms (2.8 ms)</td>
<td>367.01 ms (15.9 ms)</td>
</tr>
<tr>
<td>/l</td>
<td></td>
<td>GET</td>
<td>55.65 ms (4.3 ms)</td>
<td>62.97 ms (10.9 ms)</td>
</tr>
<tr>
<td>/ck</td>
<td></td>
<td>GET</td>
<td>64.52 ms (3.7ms)</td>
<td>74.19 ms (10.1 ms)</td>
</tr>
<tr>
<td>/l</td>
<td></td>
<td>PUT</td>
<td>55.46 ms (4.7 ms)</td>
<td>60.75 ms (9.9 ms)</td>
</tr>
<tr>
<td>/ck</td>
<td></td>
<td>PUT</td>
<td>108.35 ms (4.0 ms)</td>
<td>118.87 ms (12.5 ms)</td>
</tr>
</tbody>
</table>

- GET: Get
- PUT: PUT

- *Deferred:* X means deferred or asynchronous, → higher retrieval time and number of transmitted bytes

- *With coap-04:* /st and /r do not need to be deferred, response timeout now 2 s
Measurements - Single-Hop

<table>
<thead>
<tr>
<th>Resource</th>
<th>Deferred</th>
<th>Method</th>
<th>Mean RTT $\bar{x}$ (Standard Deviation s)</th>
<th># of bytes</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>Ch. 26</td>
<td>Ch. 12</td>
</tr>
<tr>
<td>/st</td>
<td>X</td>
<td>GET</td>
<td>295.64 ms (2.4 ms)</td>
<td>302.38 ms (12.1 ms)</td>
</tr>
<tr>
<td>/sh</td>
<td></td>
<td>GET</td>
<td>133.07 ms (4.2 ms)</td>
<td>141.72 ms (10.4 ms)</td>
</tr>
<tr>
<td>/sv</td>
<td></td>
<td>GET</td>
<td>74.59 ms (4.4 ms)</td>
<td>83.55 ms (10.7 ms)</td>
</tr>
<tr>
<td>/r</td>
<td>X</td>
<td>GET</td>
<td>363.39 ms (2.8 ms)</td>
<td>367.01 ms (15.9 ms)</td>
</tr>
<tr>
<td>/l</td>
<td></td>
<td>GET</td>
<td>55.65 ms (4.3 ms)</td>
<td>62.97 ms (10.9 ms)</td>
</tr>
<tr>
<td>/ck</td>
<td></td>
<td>GET</td>
<td>64.52 ms (3.7 ms)</td>
<td>74.19 ms (10.1 ms)</td>
</tr>
<tr>
<td>/l</td>
<td></td>
<td>PUT</td>
<td>55.46 ms (4.7 ms)</td>
<td>60.75 ms (9.9 ms)</td>
</tr>
<tr>
<td>/ck</td>
<td></td>
<td>PUT</td>
<td>108.35 ms (4.0 ms)</td>
<td>118.87 ms (12.5 ms)</td>
</tr>
</tbody>
</table>

- GET /st and /r are deferred/asynchronous → higher retrieval time and number of transmitted bytes
- With coap-04 /st and /r do not need to be deferred, response timeout now 2 s
Measurements - Single-Hop cont.
Performance of the Constrained Application Protocol for Wireless Sensor Networks

Measurements - Single-Hop cont.

Single-Hop Round-Trip Time for CoAP on Channel 26 using blip-2.0

Single-Hop Round-Trip Time for CoAP on Channel 12 using blip-2.0
Simulations - Single-Hop

CoapBlip (Root Node)  CoapBlip

80 m
Simulations - Single-Hop

CoapBlip (Root Node) to CoapBlip

Simulated Single-Hop Round-Trip Time for Resource /l

- Simulated RTT
- Simulated Mean

# Measurements vs. Round-Trip Time [ms]
Comparison of Measurements and Simulations - Single-Hop

- Additional $\sim 40$ ms due to PPP stack
- Standard Deviations show similar values
Comparison of Measurements and Simulations - Single-Hop

- Additional ~40 ms due to PPP stack
- Standard Deviations show similar values
Measurements - Multi-Hop

<table>
<thead>
<tr>
<th>Hops</th>
<th>Mean RTT $\bar{x}$ (Standard Deviation $s$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>56.34 ms (4.2 ms)</td>
</tr>
<tr>
<td>2</td>
<td>74.51 ms (46.0 ms)</td>
</tr>
<tr>
<td>3</td>
<td>113.91 ms (72.9 ms)</td>
</tr>
<tr>
<td>4</td>
<td>374.99 ms (558.5 ms)</td>
</tr>
</tbody>
</table>
Measurements - Multi-Hop

<table>
<thead>
<tr>
<th>Hops</th>
<th>Mean RTT $\bar{x}$ (Standard Deviation $s$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>56.34 ms (4.2 ms)</td>
</tr>
<tr>
<td>2</td>
<td>74.51 ms (46.0 ms)</td>
</tr>
<tr>
<td>3</td>
<td>113.91 ms (72.9 ms)</td>
</tr>
<tr>
<td>4</td>
<td>374.99 ms (558.5 ms)</td>
</tr>
</tbody>
</table>
Measurements - Multi-Hop cont.

![Graph showing measured multi-hop round-trip time for CoAP on Channel 26 using blip-2.0](image-url)
Measurements - Multi-Hop cont.

**Measured Multi-Hop Round-Trip Time for CoAP on Channel 26 using blip-2.0**

**Cumulative Distribution Function for Multi-Hop Round-Trip-Time on Channel 26 using blip-2.0**
Simulations - Multi-Hop

CoapBlip (Root Node)  CoapBlip  CoapBlip  ...  CoapBlip

fec0::1  fec0::2  fec0::3  fec0::N

80 m  80 m

Simulated Multi-Hop Round-Trip Time of a Line-Scenario with 17 Nodes
Simulations - Multi-Hop

Simulated Multi-Hop Round-Trip Time of a Line-Scenario with 17 Nodes
Comparison of Measurements and Simulations - Multi-Hop

In real measurements:

- Additional \(~40\) ms due to PPP stack
- Retransmissions occur on Link-Layer due to lower SNR
- Additional CoAP retransmissions after 1 s
Comparison of CoAP and HTTP (General Setup)

Reliable connection:

CoapBlip 6LoWPAN IPBasestation CoAP Example Client
Comparison of CoAP and HTTP (General Setup)

Reliable connection:

CoapBlip

6LoWPAN

TCPEchoWithHTTP

6LoWPAN

CoAP Example Client

IPBasestation

bareHTTPClient - TCP

IPBasestation
Comparison of CoAP and HTTP (General Setup)

Reliable connection:

CoapBlip

6LoWPAN

TCP EchoWithHTTP

6LoWPAN

Unreliable connection:

UDP EchoWithHTTP

6LoWPAN

bareHTTPClient - UDP

bareHTTPClient - TCP
Round-Trip Time of CoAP and HTTP

Round-Trip Time for HTTP/TCP, HTTP/UDP and CoAP/UDP on Channel 26
Round-Trip Time of CoAP and HTTP

Round-Trip Time for HTTP/TCP, HTTP/UDP and CoAP/UDP on Channel 26

Round-Trip Time for HTTP/TCP, HTTP/UDP and CoAP/UDP on Channel 12
Summary of the HTTP and CoAP Comparison

<table>
<thead>
<tr>
<th></th>
<th>HTTP/TCP</th>
<th>HTTP/UDP</th>
<th>CoAP/UDP</th>
</tr>
</thead>
<tbody>
<tr>
<td># messages</td>
<td>9</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td># bytes</td>
<td>681 bytes</td>
<td>217 bytes</td>
<td>111 bytes</td>
</tr>
<tr>
<td>(\bar{\chi} \RTT \text{ (s)})</td>
<td>Ch. 26 278.30 ms (7.2 ms)</td>
<td>Ch. 12 189.33 ms (3.9 ms)</td>
<td>Ch. 12 65.46 ms (4.4 ms)</td>
</tr>
<tr>
<td></td>
<td>287.16 ms (19.3 ms)</td>
<td>193.59 ms (20.7 ms)</td>
<td>72.70 ms (6.2 ms)</td>
</tr>
</tbody>
</table>

CoAP performs better than HTTP/TCP

Message size: \(\sim 6\) times lower

RTT: \(\sim 4\) times lower
Summary of the HTTP and CoAP Comparison

<table>
<thead>
<tr>
<th></th>
<th>HTTP/TCP</th>
<th>HTTP/UDP</th>
<th>CoAP/UDP</th>
</tr>
</thead>
<tbody>
<tr>
<td># messages</td>
<td>9</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td># bytes</td>
<td>681 bytes</td>
<td>217 bytes</td>
<td>111 bytes</td>
</tr>
<tr>
<td>( \bar{x} ) RTT (s)</td>
<td>Ch. 26 278.30 ms (7.2 ms)</td>
<td>Ch. 12 287.16 ms (19.3 ms)</td>
<td>Ch. 12 65.46 ms (4.4 ms)</td>
</tr>
<tr>
<td></td>
<td>Ch. 12 189.33 ms (3.9 ms)</td>
<td>Ch. 12 193.59 ms (20.7 ms)</td>
<td>Ch. 12 72.70 ms (6.2 ms)</td>
</tr>
</tbody>
</table>

- CoAP performs better than HTTP/TCP
- Message size: \( \sim 6 \) times lower
- RTT: \( \sim 4 \) times lower
Memory Usage of CoAP on TelosB Nodes

- Difference UDPEcho - CoAP Server without resources: 934 Byte
- Increase of CoAP with resources is mostly because of additional components (Sensors, URIs, Translate Components)
Conclusions and Outlook

- **libcoap** and TinyOS adaptation is default implementation of CoAP in TinyOS
- **libcoap** is a generic library; specific custom-made implementations might be smaller
- CoAP implementation more compact and bandwidth effective than HTTP/TCP, but highly reliable on UDP
- Suitable for multi-hop when response timeout is increased
- RTTs are influenced by channel conditions

Outlook

- Integration of new versions of **libcoap**, e.g. coap-07
- Interoperability with other hardware platforms and programming languages, e.g. Java
- Further multi-hop measurements in real scenarios, e.g. cargo container
Thank you for your attention
CoAP vs HTTP (Message Exchange HTTP)

Message sequence for UDP and TCP:

- **Three-way handshake (Establishing)**
- **Request:**
  
  ```
  GET /l HTTP/1.0
  
  Content-Type: application/octet-stream
  
  
  
  
  ```

- **Response:**
  
  1. **Status-Line:**
     
     HTTP/1.0 200 OK
     
  2. **Acknowledgment**
  3. **Response:**
     
     2

- **Three-way handshake (Termination)**
  
  with Acknowledgment for response
CoAP vs HTTP (Message Exchange CoAP)

- **Request:**
  - Header: CON GET
  - Options: \1 & Content-Type

- **Response:**
  - Header: ACK 200
  - Options: Content-Type
  - Payload: 2

**Figure: CoAP**

**Figure: HTTP/UDP**