Implementing CoAP for Class 1 Devices

draft-kovatsch-lwig-class1-coap-00

Matthias Kovatsch, ETH Zurich
Class 1 Stack Configuration

<table>
<thead>
<tr>
<th>Layer</th>
<th>Protocol</th>
<th>MCUs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Application</td>
<td>CoAP</td>
<td>~ 100 KiB ROM</td>
</tr>
<tr>
<td>Transport</td>
<td>UDP</td>
<td>~ 10 KiB RAM</td>
</tr>
<tr>
<td>Network</td>
<td>IPv6 / RPL</td>
<td></td>
</tr>
<tr>
<td>Adaption</td>
<td>6LoWPAN</td>
<td></td>
</tr>
<tr>
<td>MAC</td>
<td>CSMA / link-layer bursts</td>
<td></td>
</tr>
<tr>
<td>Radio Duty Cycling</td>
<td>IEEE 802.15.4e / ContikiMAC / A-MAC</td>
<td></td>
</tr>
<tr>
<td>Physical</td>
<td>IEEE 802.15.4</td>
<td></td>
</tr>
</tbody>
</table>
Experience from Erbium CoAP for Contiki

COAP IMPLEMENTATION
Memory Management

• Static

• Key parameters
  – Message buffer size (maximum message size)
  – Number of open transmissions

• Typical sizes
  – Messages with 128 or 256 bytes of payload
  – Often not even full MTU of 1280 bytes supported
Message Buffers

• Cooperative multi-threading
  – In-place processing
    – Strings point directly into IP buffer
    – Numerics parsed into variables (message struct)
    – Create reply directly in IP buffer? Good for ACKs
Separate Responses

• Required for long-lasting resource handlers
• Often done in split-phase execution, e.g.:
  – Activate sensor and wait for callback
  – Send UART command and wait for reply
• Provide API to avoid code duplication
  – automatically ACK request
  – store relevant information such as remote address
  – resume later to create response
Retransmissions

• Provide message buffers
  – Store serialized message for retransmissions
    • Requests do not change
    • CON Responses usually from long-lasting handler

<table>
<thead>
<tr>
<th>Header</th>
<th>Payload</th>
</tr>
</thead>
</table>

– Provide payload part as application buffer
e.g., to serialize JSON
– Problem: Maximum header size estimation
Observing

- Manage observe entries per resource
  - Saves space of resource handles in list
  - Store address, port, token, and last MID
- Provide one message buffer per obs. resource
  - Serialize message once and patch address + token
  - Store each retransmission state in observe entry
  - Easy to continue retransmissions with new state
Blockwise Transfers

• Expensive to provide buffer for whole transfer
  – Advantage of Blocks over IP fragments is that applications can make use of partial information
  – On-the-fly processing
  – Ordered blocks required at the receiver

• Main sender problem: `sonprintf()`
  – How to slice a long string into blocks?
  – Resource-specific generator function good for RAM, but bad for ROM
Deduplication

- Generic filter with endpoint list is heavy
  - \((40+2+2 + \text{timestamp})\) times number of clients
- Aim for idempotent requests
- Do optimized filtering in resource handlers
  - Number of clients to manage potentially smaller
  - Can exploit application state for detection
Experience from wireless sensor networks and ubiquitous computing

LOW POWER WIRELESS
Radio Duty Cycling

• Implemented in independent layer
• Virtually always-on
• Trades energy for latency
• Example: ContikiMAC
  – Server-initiated protocol
  – 0.3% idle duty cycle at 4 Hz channel checks
• Well-suited for IEEE 802.15.4
Sleepy Nodes

• Impact on application layer
  – [I-D.vial-core-mirror-proxy]
  – [I-D.fossati-core-publish-option]
  – [I-D.rahman-core-sleepy]

• Useful for other physical layers
  – Single hop
  – Long network association times
  – Example: low-power Wi-Fi

• Useful for energy harvesting