Tiny COAP Sensors

draft-arkko-core-sleepy-sensors

Jari Arkko, Heidi-Maria Rissanen, Salvatore Loreto, Zoltan Turanyi, and Oscar Novo

Ericsson Research
Legacy, Non-IP Technology

Can we do the same on IP?

YES we can!
Motivation

The goal was to create IP(v6) based sensors with

1. Natural support for *sleeping* nodes
2. Build something so simple that it could be re-implemented later with *gates* (not CPUs)
3. Communication models that fit the problem at hand
4. Good design from user perspective
Non-Goals

This is NOT

1. A general purpose implementation of COAP or any other protocol; we only implement what is actually needed in the application context
2. An implementation for general purpose computers
3. RFC compliance exercise. It works. 'nuff said.
Highlights from the Implementation

- Consists of 48 lines of assembler code
- Ethernet, IPv6, UDP, COAP, XML, and app
- Multicast, checksums, msg and device IDs
- Approaches theoretical minimum power usage
- No configuration needed

Look for packets to ff02::fe00:1 in the IETF wired network!
Making Small Implementations: Problem 1 – Sleeping Nodes

The device should ideally sleep as much as possible

The fundamental issue is having to wait for responses

- Asking for an address from DHCP, waiting for a prefix from RA, waiting for DAD responses, waiting for COAP/HTTP requests, or waiting for COAP registrations

The communication model is wrong!

Do this instead:

1. Sensors multicast their readings
2. A cache node collects the messages
3. Other nodes access the cache at any time
Power Savings Comparison

Lets assume periodic messages once per minute. On a 10Mbit/s interface sending one message takes 100 us, i.e., ratio of sleep vs. awake is 600.000x

A node that wakes up for one second every minute to listen has a ratio of only 60x

10.000x difference!!

Even if we assume that it takes ten times more to wake up and process the packet than the actual line speed is, we still get a 1.000x difference
Making Small Implementations: Problem 2 – Address Configuration

How do we get an address without having to stay awake?

The solution:

1. Use IPv6 link-local source addresses
   - No need to wait for RAs or remember prefixes
2. Use MAC-address -based generation of these addresses
3. Do not employ DAD
   - Not quite according to the RFC... but works better
Making Small Implementations: Problem 3 – Zero Configuration

How do we avoid having to configure these tiny devices?

The solution:

1. Sensor IDs are burned into the hardware at factory
2. Sensors use multicast, no need to know any specific destination addresses
3. All configuration that might be needed (e.g., sensor X is at room Y) happens at the gateway/cache node
Draft Schema for HW Implementation
Reflections on COAP

There are areas where additional documentation is needed

- How one should use multicast
- What data to include (URNs, payloads, options)
- How to configure COAP nodes in practical networks

But there are also fundamental concerns

- The lightweight nature of COAP is more about small changes to syntax and behavior (TCP=>UDP) than about eliminating reasons behind complexity and power usage

  *Like re-arranging the deck chairs on Titanic!*

- COAP can (perhaps) be used in sleepy nodes, but it requires great care
Communication Models: 1. Send-Only
Communication Models: 1. Send-Only

User or Intermediary
(Server)

Sensor
(Client)

- NON/POST content
- power-down
- NON/RSP

wake-up
2. Send & Confirm
2. Send & Confirm

User or Intermediary
(Server)

CON/POST
content

Sensor
(Client)

wake-up

ACK/RSP

power-down

IETF | 2011-03-31 | Page 15
3. Server
3. Server

User or
Intermediary
(Client)

CON/GET

ACK/RSP
content

Sensor
(Server)
4. Observer

User or Intermediary (Client)  Sensor (Server)

NON/GET observe registration

power-down

wake-up

power-down

wake-up

power-down
4. Observer

User or Intermediary (Client)  
NON/GET observe registration  
NON/RSP content  

Sensor (Server)  
power-down  
wake-up  
power-down  

power-down  

wake-up
Suggested Changes to COAP Specs

Multicast and Non-Confirmable requests:
- Specify better what the re-transmission rules should be for non-confirmable requests
- Specify what the multicast transmission rules are with respect to congestion (random delays etc)
- Consider standardizing what destination addresses and target URIs to use

Communication models
- Explain the implication of different models
- Change the observer model so that it becomes compatible with sleeping nodes