Guidance for Light-Weight Implementations of the Internet Protocol Suite

Carsten Bormann, Ed.
Contributors:
Olaf Bergmann
Tero Kivinen
Carl Williams
Mitsuru Kanda

Outline

- Background
- Drawing the landscape
- Document structure
- Techniques
  - 6LoWPAN
  - CoAP
  - API
  - PANA

Background

- What we have
  - Protocol specs: 6LoWPAN, 6LoWPAN_ND, RPL, CoAP…
- What are in need
  - Techniques to implement these optimized protocols
  - Guidance to make the implementation small and interoperable
- Objectives :
  - Collect experiences from implementers of IP stacks on constrained devices
  - Knowledge of the art of the literature, helpful for future practice
  - Conformance with the relevant specs
  - Not software engineering best practices

"I program, therefore I am...."

Classes of “Constrained” Devices

- Distinguish 2 rough classes of constrained nodes:

	Data Size	Code Size
Class 1	~10 KB	~ 100 KB
Class 2	~50 KB	~ 250 KB

- In each case, make clear which class is being targeted
- (These are a starting point for making sure we discuss from the same requirements, not exact classes.)

Implementation styles

- Single-threaded/giant mainloop
- Event-driven vs. threaded/blocking
- Single/multiple processing elements
  - E.g., separate radio/network processor

- In mind:
  - Some techniques may be applicable only to some of these styles!

Roles of nodes

- Constrained nodes
  - Sleepy nodes
- Nodes talking to constrained nodes
  - To sleepy nodes
  - Normally always alive
- Gateways/Proxies
  - To sleepy nodes
  - Could be always alive

Document Overview

- Data Plane
  - 6LoWPAN
  - CoAP
- Control Plane
  - RPL
- Security
  - PANA

6LoWPAN Route-Over Fragment Forwarding

Contributor: Carsten Bormann
Universität Bremen TZI
cabo@tzi.org

6LoWPAN Implementation Tricks:

Fragment Forwarding Technique

- 6LoWPAN:
  adaptation layer fragmentation can be needed
- Route-Over happens above adaptation layer
- Would have to reassemble at each hop
- Better:
  - Build cache entry on initial fragment
  - Forward initial fragment immediately
  - Forward each non-initial fragment
    based on cached IP header info

Constrained Application Protocol (CoAP)

Contributor : Olaf Bergmann
Universität Bremen TZI
bergmann@tzi.org

Trivia

- Why CoAP matters:
  - M2M communication in *constrained networks*
  - connect smart objects to the Internet
  - Goal: *HTTP equivalent* for WSNs (REST)
- Focus
  - Class 1 devices: ~10 KiB RAM, ~100 KiB Flash
  - Server applications
  - Robustness/latency vs. resources
    (power, dynamic memory needs, static code size)
  - Keep mandatory (to recognize) protocol features

Message Layer Processing

- Avoid fragmentation, retransmission
  - minimize state maintenance and power usage (especially server side)
- Must have send buffer and tick counter (or RTC)
- To generate separate responses, servers must keep client's transport address and Token
- Sleepy nodes
  - fix up clock if interrupts are disabled during sleep
  - No sleep for the first 1 or 2 retransmission cycles

Message Parsing

- The usual parsing strategies
- Propose bit-vector for type-decoding
- Some options are allowed more than once (Uri-Path):
    - Could make last segment unique
      or collect while you parse

(How to) Proceed From Here?

- Feedback from mailing list
  - Clarify which roles are talked about
  - Analyze implementation cost for server and client
  - Hard-coded parameters (e.g. max. payload size?)
  - What about gateways and proxies?
- Security implementation?
  - Proposal: should be covered in the general security section
- Other documents
  - draft-arkko-core-sleepy-sensors
- Is this the right information to put in this document?

General considerations about Application Programming Interfaces (APIs)

Author: Carl Williams

API

- One of the roles of the API can be exactly to hide the detail of the transport protocol
- uIP application interface
  - Event driven API model
  - Standard multi-threaded model not used
- TinyOS
  - Non-blocking API
    - When application interface sends a message the routine would return immediately (before msg is sent)
    - Call-back facility notifies app when sending is done.
    - Benefit: no code runs for long periods of time; otherwise, pkt is dropped.

Work in Progress

- Gathering implementation experiences from IPSO developers
  - Attendance of IPSO late March
  - Work with API implementers in IPSO alliance

Guidance for Lightweight Security Protocol

Author: Mitsuru Kanda

Presenter: Yoshihiro Ohba

Minimal PANA Implementation

- Protocol for Carrying Authentication for Network Access defined between PaC (PANA Client) and PAA (PANA Authentication Agent)
- PaC may be sleeping
    - **Use PaC-initiated session**
        - **Sleeping device can’t process an unsolicited PAA-initiated session message**
    - **PANA ‘Ping’**
        - **Do not use PANA ‘Ping’ for mutual liveness check**
    - **Use PaC Initiated re-authentication**
        - **Sleeping device can’t process an unsolicited PAA-initiated re-authentication message**
- PANA message optimization (reduce number of messages)
    - **Use Piggybacking EAP technique**
    - **Don’t send a PTR message for PANA session lifetime expiration**

Next step

- Integrate more organized text on security and other parts

- Circulate a Questionnaire for implementers to collect information