CoRE Working Group
Internet-Draft
Intended status: Informational
Expires: May 12, 2014

E. Dijk, Ed.
Philips Research

November 8, 2013

Sleepy Devices using CoAP - Possible Solutions
draft-dijk-core-sleepy-solutions-02

Abstract

This document describes possible solutions for sleepy devices support for the CoAP protocol. The solutions aim to meet the requirements for CoAP sleepy devices in home and building control use cases. The purpose of this document is to guide and stimulate the discussion on sleepy devices support for CoAP in the CoRE WG.

Table of Contents

1. Terminology

1.1. Abbreviations

CoRE: Constrained RESTful Environments

SEP: Sleepy Endpoint

NSEP: Non-Sleepy Endpoint

1.2. Definitions

Sleepy Endpoint (SEP) : A CoAP endpoint hosted on a networked computing device, which sets its network link to a disconnected state during long periods of time to save energy. "Long" means here that the period is of such duration that most messages sent to a SEP are lost despite use of standard "reliable transmission" techniques. The device is S0 class and any of E0/E1/E2 class according to [I-D.ietf-lwig-terminology]. See also the similar definition of SEP in [I-D.rahman-core-sleepy-problem-statement].

Non-Sleepy Endpoint (NSEP) : A CoAP endpoint hosted on a networked computing device, which has its network interface in an always-connected state or operates its network interface such that the endpoint(s) on it appear always-connected. The device is S1 or S2 class and any of E1/E2/E3 class as in [I-D.ietf-lwig-terminology].

Sleeping/Asleep : A SEP being in a "sleeping state" i.e. its network interface is disconnected and a SEP is not able to send or receive messages.

Awake/Not Sleeping : A SEP being in an "awake state" i.e. its network interface is connected and the SEP is able to send or receive messages.

Destination : a NSEP to which event messages are sent by a SEP, or by a Proxy on behalf of a SEP.

Heartbeat : a type of message (event), which is sent periodically to indicate to a Destination that the sender is still operational and able to communicate to the Destination. A heartbeat message may contain data about the current status of the sender. Typically sent by a SEP.

Proxy : a NSEP which is communicating directly with a SEP; able to cache information/CoAP resources on behalf of SEP for the purpose of further distribution or making it accessible to interested endpoints. It acts as an intermediary between a SEP and a NSEP.

The Proxy provides immediate/reliable connectivity, to enable NSEPs to operate on SEP resources even while the SEP is sleeping.

In addition to these definitions, readers should also be familiar with the terms and concepts discussed in [I-D.ietf-core-coap].

1.3. Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

2. Introduction

The CoRE WG charter includes the topic of caching resources on behalf of sleepy devices. This document describes an overall architecture proposal on how support for sleepy CoAP devices can be added to Constrained RESTful systems, to support caching but also other functions. Possible solutions for the various identified functions are proposed. The motivation for sleepy CoAP devices is described in [I-D.rahman-core-sleepy-problem-statement] and [I-D.dijk-core-sleepy-reqs].

The aim of this document is to guide and stimulate the discussion on sleepy devices support in the CoRE WG. The use cases and requirements documented in [I-D.dijk-core-sleepy-reqs] are taken as the reference.

3. Architecture

Based on the use cases, requirements and existing CoRE building blocks (such as the CoAP protocol, CoAP proxying, core-observe, etc.) a solution architecture is described in this section. First we identify the components, then the interfaces between components, and finally possible solutions to realize these interfaces.

3.1. Components

From the use cases and requirements the following components (i.e. devices, or functions of devices) can be identified:

1. Sleepy Endpoint (SEP)

2. Proxy: NSEP that maintains a relation with SEP and caches resources on behalf of the SEP.

3. Destinations(s): NSEPs, other than Proxy, where SEP directly reports events to. Events are typically a change of resource. A destination endpoint may consist of a multicast group.

4. External server: a CoAP server to which a SEP can make requests e.g. for parameter updates, firmware or external information.

5. Configuring NSEP: a CoAP endpoint that changes/writes data on the SEP

6. Reading NSEP: a NSEP that needs to read a resource from the SEP. This may include resources that the SEP regularly reports as events already, or resources that a SEP did not send before.

7. Discovery Service: an optional service that enables discovery of SEPs, their resources and their associated Proxies. For example, a Resource Directory ([I-D.ietf-core-resource-directory]).

3.2. Interfaces

Below diagram shows the components and the interfaces (Ixx) that need to be defined between components to meet the various requirements for CoAP sleepy devices. The arrowheads indicate the direction of taking initiative; e.g. the arrow from SEP to Destination NSEP(s) shows that the SEP upon an event takes the initiative to send to one or more Destination(s). This "taking initiative" could be implemented either via a CoAP request or a CoAP response (e.g. a core-observe response) so an arrow does not indicate which component is acting in the role of CoAP client or CoAP server.

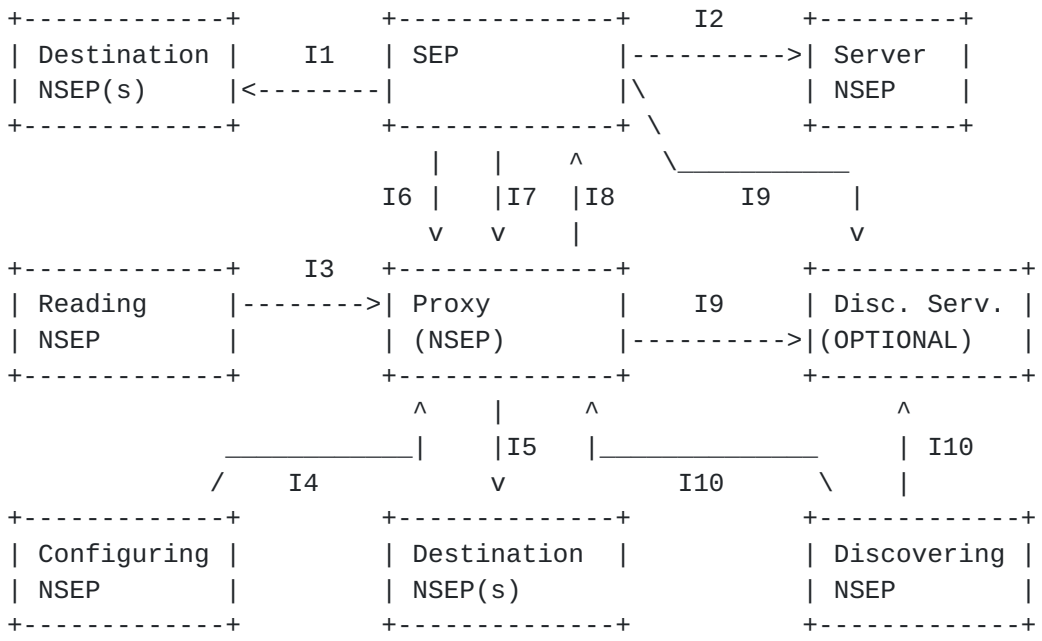

Figure 1: Architecture Components and Interfaces (numbered)

3.3. Implementation of Interfaces

This section gives possible solutions how each interface (Ixx, identified in the previous section) could be implemented. The "Rxx items" (R1, R2, etc.) between brackets show which requirements from [I-D.dijk-core-sleepy-reqs] are addressed by the interface.

3.3.1. I1: SEP Reporting to Destinations (R1)

A SEP can report events to one or more destinations using CoAP POST requests, acting as a CoAP client. For each request either NON or CON may be used. The endpoint(s) to report to can be hardcoded in the software and/or determined by the configuration applied through I4 (Section 3.3.4). In addition, a SEP may be programmed to fetch such configuration from a server through I2 (Section 3.3.2). To which resource (URI) the POST request is sent, plus the Content-Format used and the contents of this content, is determined by the implementers and/or SDOs that define application profiles on top of CoAP.

3.3.2. I2: SEP Reading from External Server (R2,R5)

A SEP can read from an NS server using CoAP (GET) requests, acting as a CoAP client. While waiting for a response, a SEP is in an awake state. Similar to I1, the selection of endpoint(s), URI and content is up to implementers and/or SDOs.

3.3.3. I3: Reading Device Reads SEP Resource(s) Via Proxy (R3)

A Reading Device sends CoAP GET requests to the Proxy (acting as a CoAP client) to read SEP resources. The Proxy serves these requests from cache. If a requested resource is not cached, various behaviors could be defined: e.g. return an error, or the Proxy returns a 5.03 Service Unavailable first, and then starts a process to GET the resource directly from the SEP using interface I8.

3.3.4. I4: Configuring Device Writes SEP Resource(s) Via Proxy (R4,R5,R7)

A Configuring Device sends CoAP PUT/DELETE requests to the Proxy in order to write/delete resources on the SEP. The resources on the Proxy that are written to are the resources that the SEP has delegated towards the Proxy.

The Proxy itself then uses I8 to update the SEP with the new resource(s).

3.3.5. I5: Proxy Notifies Destination (R1)

A Destination can use core-observe to register to resource updates on the Proxy. The Proxy sends core-observe notifications whenever the resource is updated. The resources here are the resources that the SEP has delegated to the Proxy.

3.3.6. I6: SEP Notifies Events to Proxy (R1)

The SEP sends CoAP requests (acting as a CoAP client) to the Proxy to communicate any events i.e. SEP resource updates. The format of request and response should be partly standardized in CoRE, e.g. as in Mirror Server [I-D.vial-core-mirror-server]].

3.3.7. I7: SEP Checks Proxy For Resource Updates (R4,R5)

The SEP may send a single CoAP GET request to the Proxy to check if any changes to its writeable delegated resources are available. If so, it could use multiple GET requests (one per changed resource) to get the new content from the Proxy, or perhaps a single GET to retrieve multiple resource values as a single composite representation.

Text TBD; some initial thoughts: a single message to post a sensor value (+ heartbeat) and to ask for any updates is more efficient. The proxy could use its response code to signal if any updates are available. For example, for a POST, a 2.04 Changed response may indicate a resource is changed. A code 2.00 (which is to be defined)

may indicate success but no change to resource. Or a small payload attached to the response to the POST could indicate which updates to resources are available at the Proxy, as was discussed on the CoRE WG list around March 29, 2013.

Alternative: technique that once a Proxy receives a POST from the SEP with a sensor value, it first sends a CoAP PUT request to change a resource on the SEP, and only when that is successful it provides a separate response to to the original POST request done by the SEP.

3.3.8. I8: Proxy Requests Resources from SEP (R3)

For I8 to work there needs to be a mechanism defined in I6 or I7, so that the Proxy can be notified when a SEP is awake. Then, a Proxy that needs to request resource(s) from a SEP can make the requests via interface I8 as soon as the SEP has woken up.

A solution is that a SEP acts as a regular CoAP server during the time it is awake.

3.3.9. I9: Proxy Registers Resources at Discovery Service (R7)

This interface is OPTIONAL i.e. only required if the Discovery Service is available. A SEP could communicate its available resources described in CoRE Link Format [RFC6690] to a Proxy. The Proxy is then responsible for registering these resources/ descriptions in a further Discovery Service which may be implemented as a Resource Directory [I-D.ietf-core-resource-directory].

A SEP SHOULD describe its own resources in CoRE Link Format in its "/.well-known/core" resource, such that a Proxy is able to read this resource description and to do further registration of SEP resources in a Discovery Service.

3.3.10. I10: CoAP Endpoint Discovers SEP(s) (R7)

There are two ways in which a CoAP endpoint can discover a SEP and its resources. The first way which should be supported in a system, requires that a Proxy includes in its "/.well-known/core" Link Format description the descriptions of the individual SEPs as detailed in [I-D.vial-core-mirror-server]. The CoAP endpoint can then use CoRE resource discovery ([I-D.ietf-core-coap]) using either unicast or multicast CoAP requests to discover the SEP.

The second way which is OPTIONALLY supported in a system, is that a Proxy registers the SEP into a Discovery Service (such as RD [I-D.ietf-core-resource-directory]), and the CoAP endpoint uses the specific interface of the Discovery Service to discover a SEP.

3.4. Resources

This section provides some information on the CoAP resources that need to be allocated on the different components in the architecture.

3.4.1. SEP Resources

On a SEP, a clear distinction has to be made between types of resources.

- Read-only resources: resource can be modified by the SEP, but SHOULD NOT be modifiable by any external device. This includes static information resource (e.g. manufacturer name, type, firmware version, etc.) but also volatile resources (e.g. latest sensor value, error log entries, etc.) that the SEP internally updates.

- Read/Write resources: resource can be modified by an (authorized) external device. This is used for configuration information (e.g. sensor thresholds, which Destination(s) to use, event frequency, etc.). Authorized CoAP clients can write such resource at the Proxy, which will communicate the updated resource to the SEP next time it wakes up and contacts the Proxy. To avoid write/write conflicts, such a resource SHOULD NOT be modified autonomously by a SEP.

3.4.2. Proxy Resources

TBD

3.4.3. Destination Resources

TBD. One draft proposal is: a single reporting resource for receiving events "/event". This can be changed to anything SDO/ vendor specific, but above can be a default.

3.4.4. Other Resources

TBD

4. Acknowledgements

TBD

5. IANA Considerations

This document includes no request to IANA.

6. Security Considerations

TBD: per interface the security needs and solution need to be described. Anywhere CoAP unicast is used, DTLS may apply as a transport security solution. DTLS key update on a sleepy device may pose a problem.

7. References

7.1. Normative References

[I-D.ietf-core-coap]
Shelby, Z., Hartke, K., and C. Bormann, "Constrained Application Protocol (CoAP)", draft-ietf-core-coap-18 (work in progress), June 2013.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC6690] Shelby, Z., "Constrained RESTful Environments (CoRE) Link Format", RFC 6690, August 2012.

7.2. Informative References

[I-D.dijk-core-sleepy-reqs]
Dijk, E., "Sleepy Devices using CoAP - Requirements", draft-dijk-core-sleepy-reqs-00 (work in progress), June 2013.

[I-D.ietf-core-block]
Bormann, C. and Z. Shelby, "Blockwise transfers in CoAP", draft-ietf-core-block-14 (work in progress), October 2013.

[I-D.ietf-core-observe]
Hartke, K., "Observing Resources in CoAP", draft-ietf-core-observe-11 (work in progress), October 2013.

[I-D.ietf-core-resource-directory]
Shelby, Z., Krco, S., and C. Bormann, "CoRE Resource Directory", draft-ietf-core-resource-directory-00 (work in progress), June 2013.

[I-D.ietf-lwig-terminology]
Bormann, C., Ersue, M., and A. Keranen, "Terminology for Constrained Node Networks", draft-ietf-lwig-terminology-05 (work in progress), July 2013.

[I-D.rahman-core-sleepy-problem-statement]
Rahman, A., Fossati, T., Loreto, S., and M. Vial, "Sleepy Devices in CoAP - Problem Statement", draft-rahman-core-sleepy-problem-statement-01 (work in progress), October 2012.

[I-D.vial-core-mirror-server]
Vial, M., "CoRE Mirror Server", draft-vial-core-mirror-server-01 (work in progress), April 2013.

Author's Address

Esko Dijk (editor)
Philips Research
High Tech Campus 34
Eindhoven, 5656 AE
NL

Phone: +31 40 2747947
Email: esko.dijk@philips.com