CoRE Working Group
Internet-Draft
Intended status: Informational
Expires: December 12, 2013

E. Dijk, Ed.
Philips Research
June 10, 2013

Sleepy Devices using CoAP - Requirements
draft-dijk-core-sleepy-reqs-00

Abstract

This document provides requirements for operation of sleepy CoAP devices, based on home control and building control use cases. These requirements can be used to help design, or select, a solution for sleepy CoAP devices in the CoRE WG. In addition, for the existing CoAP, core-block and core-observe functions some notes are made on how these functions could be used in an overall CoRE sleepy devices solution that meets the requirements.

Status of This Memo

Copyright Notice

Table of Contents

1. Terminology

For terminology regarding constrained nodes we use
[I-D.ietf-lwig-terminology]. This document focuses on S0 class
devices ("always-off").

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

Capitalized equirements language is used in this document to indicate the importance of requirements. "MUST" level requirements are those required to be addressed by a solution. "SHOULD" level requirements are about functionality that is preferably provided by a solution. "MAY" level requirements describe optional functionality.

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 above and in RFC 2119 [RFC2119].

2. Introduction

The CoRE WG charter includes the topic of caching resources on behalf of sleepy devices. Already, various proposals have been made in the CoRE WG with solutions to enable operation of sleepy CoAP endpoints.

During and shortly after IETF 83 (March 2012) it was proposed to first clarify the scope and the requirements that a solution should fulfil, before choosing for or developing a solution to support sleepy device operation in CoRE using CoAP.

This document aims to provide input to the mentioned requirement gathering and selection process. The application area that we focus on is Home Control and Building Control. The reader is assumed to be familiar with the Sleepy Devices in CoAP Problem Statement [I-D.rahman-core-sleepy-problem-statement]. Possibly, the content of this document can be input to the Problem Statement I-D.

First, the requirements categories and related use cases are listed in Section 3. From the use cases a set of requirements was extracted, which is detailed in Section 4. Finally, Section 5 provides some ideas how current protocols being defined in the CoRE WG ([I-D.ietf-core-coap], [I-D.ietf-core-block], [I-D.ietf-core-observe]) can be applied to a sleepy CoAP devices solution that meets the requirements.

3. Requirements Categories and Use Cases

From use cases, a number of requirements can be extracted for SEPs and for constrained RESTful systems that contain SEP devices. Section 4 will list the main requirements that we extracted.

This section lists a number of use cases in Home Control and Building Control, in which SEP functionality would be desired. The main driver for SEP functionality in CoAP is to achieve fully wireless, battery-operated or energy-harvesting CoAP devices that require minimal manual maintenance, e.g. having a long battery lifetime.

Note that CoRE application domains like Industrial Sensing, Smart Energy or Smart City are not covered by these use cases. However, it is likely that requirements from these domains overlap to a large extent with those presented here.

The requirements are organized in the categories (Rx) shown below. Per category, some use case examples are provided.

R1. Report SEP->NSEP: SEP reports new information (e.g. events) to a non-SEP, or to a multicast group of non-SEPs.

* A solar-powered daylight sensor periodically measures daylight intensity in a room and reports this information to a designated NSEP or group of endpoints. For example, a local luminaire that adapts its dim level according to daylight level.

* An energy-harvesting light switch is pressed, switches on the radio and sends a request to a light or group of lights (NSEPs) to turn on.

* A battery-powered occupancy sensor detects an event of people present, switches on the radio and sends a request to one or a group of CoAP-enabled lights to turn on.

* Alternative to above: instead of sending directly to the light(s) the sensor sends to an intermediate CoAP node (e.g. Proxy or controller) which then carries out the request to endpoint/group(s) of endpoints.

* A battery-powered temperature sensor reports room temperature to a designated NSEP that controls HVAC devices. The sensor reports periodically and reports extra when the temperature deviates from a predefined range.

* A battery-powered sensor sends an event "battery low" to a designated reporting location (NSEP).

R2. Read SEP->NSEP: SEP reads (or queries) information from a non-SEP

* A sleepy sensor (periodically) updates internal data tables by fetching it from a predetermined remote NSEP, e.g. a backend server.

* A sleepy sensor (periodically) checks for new firmware with a remote (CoAP) endpoint. If new firmware is found, the sensor switches to a non-sleepy operation mode, starts an update procedure and new firmware is downloaded in the device.

R3. Read NSEP->SEP: Non-SEP reads (or queries) information from a SEP, or from a multicast group of SEPs.

* A NSEP (e.g. in the backend) requests the status of a deployed sleepy sensor, e.g. current sensor state and/or firmware version and/or battery status and/or its error log. It expects a response within one, or at most a few, second(s).

* A NSEP (e.g. in the backend) requests the status of a group of deployed sleepy sensors. It expects responses even for the sensors that are sleeping at the time of doing the request.

* A NSEP requests information on when a sleepy sensor was 'last active' (i.e. identified as being awake) in the network.

* A NSEP subscribes itself to sensor events and status reports of multiple sleepy sensors for diagnostic purposes. The subscription relation is only temporary, until the diagnostic operation concludes.

R4. Write NSEP->SEP: Non-SEP writes (or configures, updates, deletes, etc.) information to a SEP

* An authorized NSEP changes the reporting frequency of a deployed sleepy sensor.

* An authorized NSEP adds a new reporting endpoint to an operational sleepy sensor. From that moment on, the new endpoint (NSEP) receives also the sensor events and status updates from the sleepy sensor.

* A remote NSEP instructs a sleepy sensor to upgrade its firmware. The sensor firmware is then upgraded. (In whatever way - the SEP may pull data from a server, or the remote NSEP may push data to the SEP.)

R5. Transfer NSEP->SEP: Large data volume e.g. firmware is transferred into a SEP. The data originates from a non-SEP. Either NSEP or SEP takes initiative to start the transfer.

* A remote NSEP instructs a sleepy sensor to upgrade its firmware. The sensor firmware is then upgraded. (In whatever way - the SEP may pull data from a server, or the remote NSEP may push data to the SEP.)

* A sleepy sensor (SEP) on own initiative switches to a non-sleepy operation mode, starts an update procedure and new firmware is downloaded in the device.

R6. Security

R7. Configuration, commissioning, diagnostics & maintenance

* An installer deploys a new sleepy sensor in a room. The sensor is then configured to control all lights in the room in a commissioning phase. Finally, the sleepy operation is enabled in the sensor right after the commissioning phase (i.e. start of the operational phase).

R8. General

4. Requirements

4.1. R1 - SEP Reports To NSEP

1. Reporting destination type(s). A SEP MUST be able to report directly to an endpoint or group; and to an intermediate node/ Proxy that takes care of communicating to the final endpoint/ group. Multiple reporting destinations MUST be supported. Rationale: direct unicast/multicast reporting is needed for high reliability, low latency, and avoiding single-point-of-failure situations.

2. Reporting destination location. A reporting destination endpoint MUST be able to handle any addressing like: link-local, subnet-local (e.g. 6LoWPAN), site-local (typ. corporate LAN/ Intranet), or global (WAN/Intranet/Internet).

3. Reporting latency. Any report SHOULD be delivered with low latency to the final local destination. Rationale: use cases include e.g. person detection applications, actuators react within 200 ms.

4. Reporting reliability unicast. SEP MUST be able to reliably report events in unicast. For example, a 99.9% reliability may be required in a specific use case.

5. Reporting reliability multicast. SEP MUST be able to report events in multicast to a group of NSEPs, with similar reliability as is achievable by a NSEP doing a multicast.

6. Multicast report via Proxy. It SHOULD be possible to configure a Proxy, to which a SEP reports in unicast, such that it resends the report in multicast. Rationale: to allow simplified SEPs that do not have multicast ability; or devices that do not know a priori how multicast operation needs to be done in target networks.

7. Reporting group topology. For a multicast group it MUST be possible to place group members across multiple subnets, with routers in between. Rationale: allow for flexibility and organic IP network growth.

8. SEP reporting configuration. A CoRE solution MUST leave the freedom to configure a SEP with reporting destination address(es) in the following ways: standard-defined, pre-commissioned, runtime configured, runtime discovered. Rationale: leave it to vendors or other SDOs how device relations are (pre)configured.

9. Configuration of receiver of SEP events. A CoRE solution SHOULD allow a receiver of SEP events to discover group IP address(es) it has to listen to, to receive events from a specific SEP.

10. Direct SEP subscription. A mechanism MUST be provided for NSEPs to receive events (i.e. resource updates) sent by a SEP directly to NSEP, during a given period. Event delivery MUST still work even if a Proxy is not available at the time of the event. Subscribing to events MUST work also if the SEP is sleeping at the time of subscribing.

11. Proxy subscription. A solution SHOULD be provided for clients to receive events sent by a Proxy on behalf of a SEP, during a given period. Rationale: to avoid high load on the SEP due to high number of subscribers.

4.2. R2 - SEP Reads From NSEP

1. Read direct. It SHOULD be possible for a SEP to read from/query a selected CoAP and/or HTTP server.

2. Sleep mode change. It is allowed that the SEP temporarily switches to an always-on mode to perform a Read task.

4.3. R3 - NSEP Reads From SEP

1. Read via Proxy. It MUST be possible for a NSEP to read/query designated SEP information via a Proxy. Rationale: the SEP is unavailable most of the time, making direct contact practically impossible.

2. Multicast read via proxies. It SHOULD be possible for a NSEP to perform a CoAP multicast request to read/query a group of SEPs, or a group of proxies.

3. Reading reported information. A client MUST be able to read the information (CoAP resources) that a SEP is currently reporting (e.g. periodically or event-based).

4. Reading non-reported information. A client MUST be able to read information (CoAP resources) that a SEP is currently not reporting.

   * Note: this may require that the Proxy has a specific relation to the SEP that enables it to detect when the SEP is awake, and at that moment forward the information request to the SEP for processing.

   * Note: some information (manufacturer name) may be sent a priori by the SEP to a Proxy, while other information (error log) may be sent on demand only.

5. Read latency. The response time for a read operation SHOULD NOT differ significantly from a typical NSEP->NSEP request in the same IP network, if the information is available in the Proxy. If the Proxy needs to wait for the next SEP's wakeup, the response time SHOULD NOT significantly exceed the device's current sleep duration.

6. Client location. A requesting client MUST be allowed to be link-local to the SEP, or subnet-local (e.g. 6LoWPAN), or site-local (typ. corporate LAN/Intranet), or global (WAN/Intranet/Internet).

4.4. R4 - NSEP Writes To SEP

1. Write via Proxy. It MUST be possible for an authorized NSEP to write information to or delete information on a SEP via a Proxy.

2. Client location: see R3 client location.

3. Write latency. The writing latency SHOULD be approximately equal to the current sleep interval duration of the SEP, or less.

Rationale: while SEP is sleeping, no writing can possibly take place. Best performance is when write operation occurs at the next SEP wake-up.

4. Write reliability. The write operation MUST be performed reliably. (For example, a 99% success rate may be required in a use case, with notification to the application layer about outcome of the write operation.)

4.5. R5 - Large transfer From NSEP To SEP

1. Sleeping mode. A SEP MAY switch its mode of operation temporarily to always-on to execute a data transfer.

2. Buffering requirement. A Proxy MUST NOT be required to buffer an entire data object in its memory. Rationale: a firmware image will never fit entirely in a Proxy's available memory.

3. Location of data object. See R3 client location.

4. Transfer external trigger. There MUST be a reliable way for a NSEP to trigger a SEP into starting a large data transfer. Rationale: to cover situations where the SEP itself is in best position to initiate the transfer, e.g. acting as a CoAP/HTTP client to fetch data blocks at its own pace.

5. Transfer latency. If a transfer is initiated by a NSEP, It SHOULD be possible for the transfer to start at the next time instant the SEP wakes up.

4.6. R6 - Security

1. Security level. It MUST be possible to secure communication with a SEP at a level similar to NSEP communication. (E.g., DTLS.)

2. Bootstrap security. It SHOULD be possible to secure any communication with a SEP that is needed to bootstrap a (new) SEP into a network.

3. Proxy security. The Proxy MAY be a trusted device. That is, end-to-end CoAP security from SEP to Destination is NOT REQUIRED if a Proxy is used in the communication process.

4. Write Authentication and Authorization. When information/ resources on a SEP are being written or deleted, it MUST be possible for a SEP (or its Proxy) to authenticate the writer and to check that it is authorized to do so.

5. Read/subscribe Authentication and Authorization. When information/resources on a SEP are being requested, it SHOULD be possible for a SEP (or its Proxy) to authenticate the reader/subscriber and to check that it is authorized to read.

4.7. R7 - Configuration, commissioning, diagnostics, maintenance

1. Configurable sleep interval(s). The sleep interval (i.e. heartbeat interval) MUST be fully configurable per sleepy node. A wide range of values (seconds, minutes, hours, days) SHOULD be supported. Variable sleep intervals SHOULD be supported. Multiple sleep intervals (such as a different interval for each sensor resource attached to an endpoint) MUST be supported.

2. Discoverability. There MUST be one or more ways defined for a NSEP to establish the existince of a SEP and to find the IP address to contact the device (either directly or its Proxy). Note: possible ways may be DNS discovery, Resource Directory, IP address(es) derived from hardware identifier, pre-commissioned, standard-defined, etc.

3. Discoverability 2. There SHOULD be a way to discover all proxies in a given network scope (e.g. link-local, subnet-local, site-local) and to find out which SEP they are representing.

4. Portability. A SEP SHOULD be able to be relocated to a new physical position, while keeping its existing association to an IP subnet or PAN, without requiring any reconfiguration of the SEP and/or NSEPs/proxies it communicates with.

5. A Proxy caching information from a SEP MAY be configured to store additional computed information based on past resource values, e.g. an average, standard deviation, history graph, etc.

4.8. R8 - General

1. Optional reliability. It MUST be possible for a SEP to optionally use Confirmable (CON) CoAP messaging.

2. Cached resources freshness. Having a different Max-Age (freshness duration) per resource MUST be supported.

3. Wake-up triggers. Wake-up based on a timer trigger, wake-up based on an (unpredictable) event trigger (e.g. sensor based), and a combination of both all MUST be possible.

4. Local Proxy. A SEP MAY rely on the presence of at least one "parent/Proxy" device in radio range.

5. Reception windows. A SEP SHOULD enable a radio reception interval at least once every interval it is awake.

4.9. R9 - Non-functional requirements

1. Implementation complexity. A solution SHOULD have minimal implementation complexity on the SEPs. Even if this implies shifting additional complexity to the clients/proxies. Rationale: sleepy sensor devices are expected to be more constrained along multiple resource axes (RAM, ROM, MCU, energy).

2. Configuration effort sleepy. A solution SHOULD operate with minimal configuration effort of SEPs.

3. Configuration effort proxies/clients. A solution SHOULD operate with minimal configuration effort of non-SEPs, keeping in mind that both configuration effort and implementation complexity for SEPs should be minimized with higher priority. Rationale: SEPs are typically harder to configure once deployed, due to frequent/ unpredictable sleep periods.

4. Network load. A solution SHOULD introduce minimal extra network load (number of messages, message sizes, etc.)

5. Battery life. A CoRE SEP solution SHOULD aim to provide as long as possible battery life for SEPs. Battery life of non-SEPs is assumed to be of minor importance.

6. Scalability. Number of devices that can subscribe to events from a single SEP SHOULD be as high as possible.

7. Adaptability. SEPs SHOULD be aware, or made aware, of any relevant network configuration changes.

8. NSEP communication effort. The "effort" a NSEP has to spend on communicating with SEPs SHOULD be minimal. Note: "effort" here includes complexity, need for protocols in addition to core-coap, number of transactions/messages needed, waiting time, etc.

5. CoRE Building Blocks in a Sleepy Devices Solution

In the CoRE WG there are various functions, or "building blocks", being developed that could be applied in a CoRE sleepy devices solution.

o [I-D.ietf-core-coap] CoAP proxy: can possibly be re-used to implement the role of Proxy as defined in this document. However, this could require adding some special functions/measures to deal

with SEPs. Currently a CoAP proxy will have the problem that it may not reach the SEP for prolonged periods of time, e.g. to forward a CoAP request or a CoAP+observe request to the SEP.

- [I-D.ietf-core-observe]: the core-observe paradigm and subscription + notification syntax can possibly be re-used to meet the subscription/notification requirements. A problem to solve is that a CoAP client is unable to perform an observe request to a SEP, since it is likely to be sleeping at the time the request is made. In worst case, the SEP can never be reached by the client.

- [I-D.ietf-core-block]: can be used for implementing the requirements R5 (Large transfers, Section 4.5). However, a blockwise POST or PUT initiated by a NSEP can not be immediately used, again due to the reason that the SEP is likely to be asleep at the time(s) the NSEP sends its request.

6. Acknowledgements

Thanks to Peter van der Stok and Sye Loong Keoh for reviewing the text.

7. IANA Considerations

This document includes no request to IANA.

8. Security Considerations

This document contains requirements for solutions, including security requirements (Section 4.6). Refer to [I-D.ietf-core-coap] Section 11.2 for security considerations on (caching) proxies. This is very relevant as the requirements indicate that use of a caching proxy may be necessary.

9. References

9.1. Normative References

[I-D.ietf-core-coap]
Shelby, Z., Hartke, K., and C. Bormann, "Constrained Application Protocol (CoAP)", draft-ietf-core-coap-17 (work in progress), May 2013.

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

9.2. Informative References

[I-D.ietf-core-block]
Bormann, C. and Z. Shelby, "Blockwise transfers in CoAP", draft-ietf-core-block-11 (work in progress), March 2013.

[I-D.ietf-core-observe]
Hartke, K., "Observing Resources in CoAP", draft-ietf-core-observe-08 (work in progress), February 2013.

[I-D.ietf-lwig-terminology]
Bormann, C., Ersue, M., and A. Keranen, "Terminology for Constrained Node Networks", draft-ietf-lwig-terminology-04 (work in progress), April 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.

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