Internet Engineering Task Force
Internet-Draft
Intended status: Standards Track
Expires: September 1, 2012

T. Fossati
KoanLogic
P. Giacomin
Freelance
S. Loreto
Ericsson
M. Rossini
CS Dept. University of Bologna

February 29, 2012

Sleepy Option for CoAP
draft-giacomin-core-sleepy-option-00

Abstract

This memo defines a framework for allowing asynchronous communication between sleepy sensors mediated by a supporting Proxy node. The Proxy acts as a store-and-forward agent that handles requests on behalf of a sleepy client, and buffers responses coming from the target origin until the requesting client wakes up and get the computation results.

A new CoAP option, Sleepy, is defined to initiate and control the asynchronous exchange.

Status of this Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on September 1, 2012.

Copyright Notice

Table of Contents

1. Introduction

The proposal described in this memo covers the following use case:

> a node A, displaying a very short duty-cycle, needs to interact with one or more resources hosted at another sleepy node B. The probability of an empty intersection between their respective wake periods is quite high, making it hard for the two to synchronize.

The proposal is to arm the Proxy with the ability to act as a store-and-forward agent mediating the request/response exchange between A and B.

A declares the will to act onto a given resource hosted at B to the Proxy, and gives a "get back" indication that tells the Proxy the time at which it is going to be on duty again, and willing to retrieve the response from B.

The Proxy is in charge of making the request on behalf of A, using an appropriate poll interval for a time span upper bounded by the "get back" value, and to buffer the response from B until A wakes up again.

This draft defines a new CoAP elective option, Sleepy, targeted specifically at proxies and used to signal a Proxy the will to initiate an asynchronous request/response exchange. The Sleepy option is partitioned in three subfields indicating: the remaining time before sleep, the expected sleep interval, and (optionally) the on-duty interval.

1.1. Terminology

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

Sleepy Device: a sensor/actuator (usually battery operated) that switches off its radio beyond the normal radio duty cycle in order to save energy.

Store-and-Forward Proxy: a CoAP proxy that is able to act as an intermediate agent where CoAP PDUs are received, kept, and sent at a later time to the final destination or to another intermediate station. Its use may be especially helpful in networks with intermittent connectivity, such those hosting a significant amount of sleepy devices.

2. Motivation

This memo focuses on the requirement REQ3 of [I-D.shelby-core-coap-req]:

REQ3: The ability to deal with sleeping nodes. Devices may be powered off at any point in time but periodically "wake up" for brief periods of time.

3. Basic Message Flows

In the most general scenario both A and B are sleepy endpoints showing empty intersection as to their wake intervals, while the Proxy cache is empty.

3.1. Basic Message Flow with CON Semantics

A typical flow of communication involving the Sleepy option using CON messaging is shown in Figure 1.

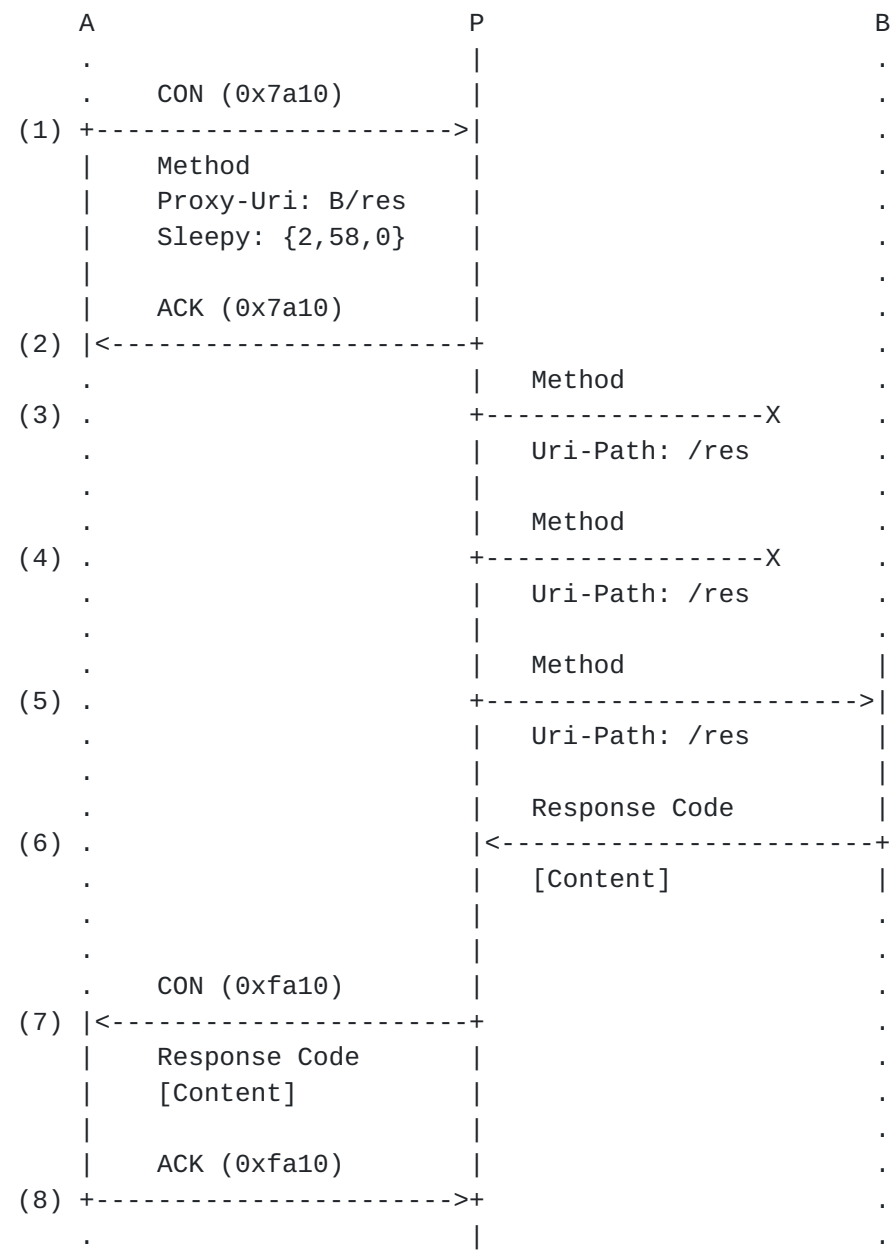

Figure 1

In message (1) a sleepy node, A, asks the Proxy to act upon the resource identified by the Proxy-Uri Option in a possibly asynchronous way by supplying the Sleepy Option indicating a time at which A thinks it may be ready (i.e. awake) to retrieve the response message. The Sleepy Option in the message (1) tells the Proxy that A will go off-duty in 2 milliseconds, and it will be off-duty for 58 milliseconds, but it does not provide any information about the optional on-duty interval.

In case the Proxy understands the Sleepy Option, it replies (2) with

a separate ACK.

From now on A can get back to sleep while the Proxy sends periodically the request to the target node, B -- messages (3-5) -- and eventually gets a response back (6).

The stored response is kept by the Proxy until A is on duty again. (The seeming on-duty time is computed using the quantities previously supplied by A through the Sleepy Option.) The Proxy sends the separate response back, operating with the usual rules of CON retransmission, until an ACK from A is received, or the transmission retries are exhausted.

Please note that, generally speaking, the framework is completely agnostic as to the transported message type and method. Further, the Proxy may rearrange any implied block-wise transfer [I-D.ietf-core-block] or separate acknowledgment in an optimal way.

[[OPEN ISSUE 1: What if message (2) is lost ?]]

3.2. Basic Message Flow with NON Semantics

In case the sleepy sensor uses NON semantics, the resulting exchange is the basically the same as the one depicted in Figure 1 with messages (2) and (8) removed.

4. Optimized Message Flow

The Proxy/Cache, in charge of making the request on behalf of A, MUST try to immediately satisfy a request by searching the Cache.

Figure 2 shows a request from A which can be satisfied from the cache (i.e. cache hit) without interrogating B.

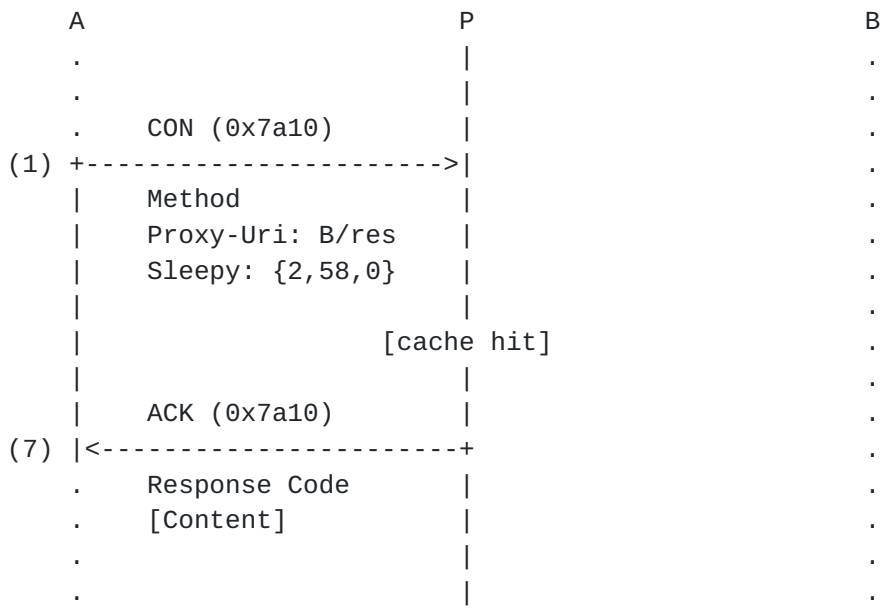

Figure 2

In case a request from A can not be satisfied from the Cache (i.e. cache miss), the Proxy, in charge of making the request on behalf of A, sends periodically the request to the target node and eventually gets a response back.

Figure 3 shows a cache miss scenario where the Proxy, knowing that the target node is awake, forwards the request to B (5) and sends the response to A (7) within the "time left before sleeping" indication supplied by A with the request (1). The latter exchange SHALL concurrently arm a timeout for sending the ACK message to A before it goes to sleep (in case CON is in use).

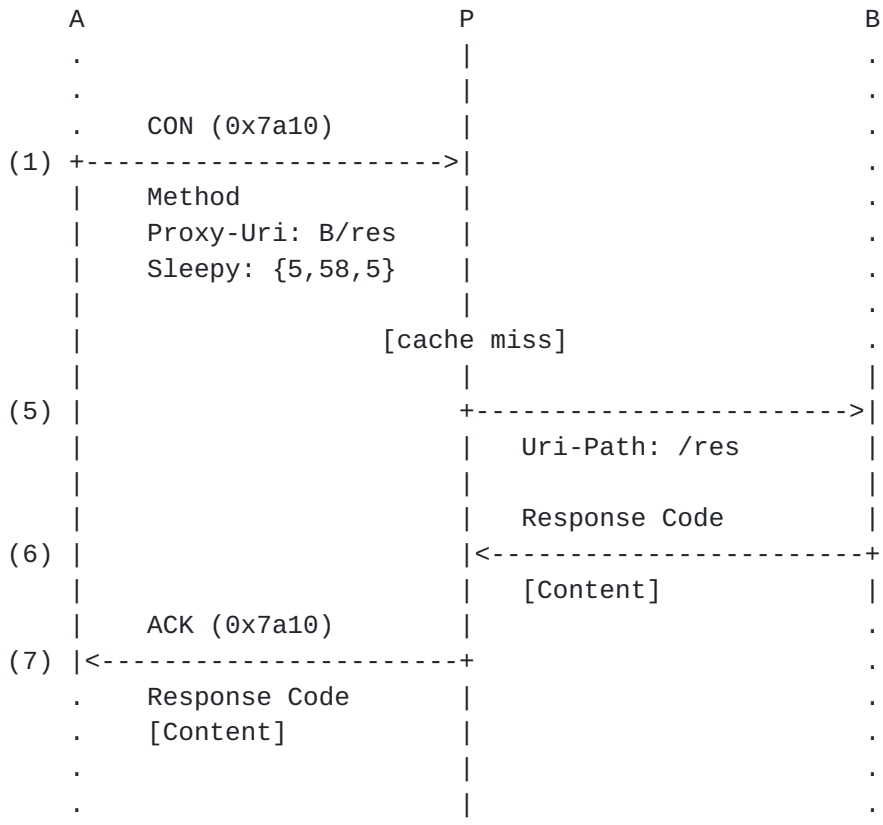

Figure 3

In any case, if the Proxy has previously received an indication from the same target about its on/off-duty behavior via the Sleepy Option (Section 5), or by any other means (e.g. Section 7), it MUST use it to devise the most efficient poll strategy, thus avoiding unnecessary messaging which would just aggravate the constrained network congestion.

5. Sleepy Option

No.	C/E	Name	Format	Length	Default
XX	Elective	Sleepy	uint	8-12 B	(none)

The Sleepy Option in a request is used to signal a Proxy the will to initiate an asynchronous request/response exchange.

The Sleepy option is elective. If the Proxy does not recognize it, it will try to serve a fresh representation of the requested resource, or forward the request to the intended origin; depending on

the availability of the endpoints at the time the Proxy tries to contact them, the usual proxied transaction may succeed, partially fail, or completely fail.

The Sleepy Option MAY be discretionarily piggybacked by a sleepy node on response messages to inform the network about the sleepy pattern in use at the endpoint. This knowledge MAY be used by sleepy-friendly Proxies to reduce the overall network congestion that is implied by resorting to blind polling in order to maximize the chance to get a response from the target.

The value of the Sleepy option is partitioned in three subfields indicating: the remaining time before sleep, the expected sleep interval, and (optionally) the on-duty interval.

Two formats are available, a long format (Figure 4), and a short one (Figure 5) which are easily distinguished from the Length field of the encoded option: 8 and 12 respectively.

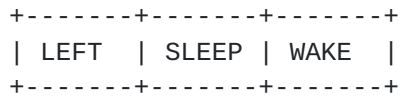

Figure 4

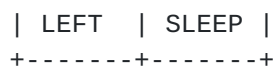

Figure 5

LEFT: 32-bit uint encoding the number of milliseconds that the sending node is left before going off-duty. The maximum value is 0xFFFFFFFF, which allows for 71582 minutes.

SLEEP: 32-bit uint encoding the number of milliseconds that the sending node is off-duty. The maximum value allows for 71582 minutes (i.e. approx. 50 days).

WAKE: optional 32-bit uint encoding the number of milliseconds that the sending node is on-duty.

[[OPEN ISSUE 2: shrink to 24-bit uint LEFT and WAKE (i.e. max ~4 hours) ?]]

[[OPEN ISSUE 3: change milli to seconds ?]]

6. Limiting Network Congestion

The retransmit function at the Proxy is conflicting with the overall requirement of congestion avoidance on the constrained network.

Therefore the proxy SHOULD try to learn as much as possible about the on/off-duty behavior of the nodes that it is trying to reach, and keep the gained knowledge to inform future message exchanges with these endpoints.

Missing an explicit signaling at the network/transport layer, endpoints that have a predictable sleep/awake pattern SHOULD try to inform the other entities in the network by piggybacking, whenever possible, the Sleepy Option in the messages (both requests and responses) they are exchanging with other peers.

A further possibility is to distribute the information regarding the sleep/awake pattern by extending the resource attributes available through the Resource Directory with a link-format [I-D.ietf-core-link-format] version of the Sleepy option (see Section 7).

7. New Link-Format Attributes

This specification defines the following new attributes for use in the CoRE Link Format:

```
link-extension = ( "sleep" "=" 1*DIGIT )
link-extension = ( "wake" "=" 1*DIGIT )
link-extension = ( "start" "=" 1*DIGIT ) ; in seconds since Epoch
```

The sleep and wake attributes have the same semantics and format as the SLEEP and WAKE subfields of the Sleepy Option respectively (Section 5). The start attribute sets the base time from which the offsets indicated by sleep and wake must be computed.

8. Acknowledgements

[TBD]

9. IANA Considerations

The following entries are added to the CoAP Option Numbers registry:

Number	Name	Reference
2k	Sleepy	RFC XXXX

The "start", "wake" and "sleep" attributes need to be registered when a future Web Linking attribute is created.

10. Security Considerations

The same considerations as those highlighted in Section 10.3.2 and 10.3.3 of [I-D.ietf-core-coap] apply, and are somewhat amplified by the possible congestion induced by the tentative setup of communication with the target node (messages 3-5 in Figure 1). The Proxy SHOULD try to send as little messages as possibile in order to contact the requested endpoint and MUST make use of the wake/sleep indication in case they have been previously made available by the target node through the Sleepy Option.

11. References

11.1. Normative References

[I-D.ietf-core-block]
Bormann, C. and Z. Shelby, "Blockwise transfers in CoAP", draft-ietf-core-block-08 (work in progress), February 2012.

[I-D.ietf-core-coap]
Frank, B., Bormann, C., Hartke, K., and Z. Shelby, "Constrained Application Protocol (CoAP)", draft-ietf-core-coap-08 (work in progress), October 2011.

[I-D.ietf-core-link-format]
Shelby, Z., "CoRE Link Format", draft-ietf-core-link-format-11 (work in progress), January 2012.

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

11.2. Informative References

[I-D.shelby-core-coap-req]
Shelby, Z., Stuber, M., Sturek, D., Frank, B., and R.

Kelsey, "CoAP Requirements and Features",
draft-shelby-core-coap-req-02 (work in progress),
October 2010.

Authors' Addresses

Thomas Fossati
KoanLogic
Via di Sabbiuno, 11/5
Bologna 40100
Italy

Email: tho@koanlogic.com

Pierpaolo Giacomin
Freelance

Email: yrz@anche.no

Salvatore Loreto
Ericsson
Hirsalantie 11
Jorvas 02420
Finland

Email: salvatore.loreto@ericsson.com

Mirko Rossini
CS Dept. University of Bologna

Email: mirko.rossini@ymail.com