CoRE   M. Vial
Internet-Draft   Schneider-Electric
Intended status: Standards Track   March 2, 2012
Expires: September 3, 2012

CoRE Mirror Proxy
draft-vial-core-mirror-proxy-00

Abstract

This document introduces the concept of Mirror Proxy that enables sleeping devices to participate in a REST architecture despite the fact that they are not web servers. Most constrained devices may sleep during long periods preventing them from acting as traditional web servers. However as client-only endpoints they can rely on a Mirror Proxy to cache and serve the content they provide.

Status of this Memo

Copyright Notice

Table of Contents

1. Introduction 3
2. Requirements Language 3
3. Architecture 3
3.1. Resource mapping in a Mirror Proxy 4
3.2. Cache refresh strategies 5
3.3. Placing a Mirror Proxy 7
4. Mirror Proxy interfaces 7
4.1. Discovery 7
4.2. Registration 7
4.3. Update 8
4.4. Validation 8
4.5. Removal 8
4.6. Lookup 8
5. Examples 8
5.1. Discovery 9
5.2. Registration 9
5.3. Content refresh 10
5.4. Polling 11
6. TODO 12
6.1. Extensions 12
7. Acknowledgements 13
8. IANA Considerations 13
9. Security Considerations 13
10. References 14
10.1. Normative References 14
10.2. Informative References 14
Author's Address 14

1. Introduction

The Constrained RESTful Environments (CoRE) working group aims at realizing the REST architecture in a suitable form for the most constrained nodes (e.g. 8-bit microcontrollers with limited RAM and ROM) and networks (e.g. 6LoWPAN). As pointed out by [I-D.arkko-core-sleepy-sensors], the server model is far from being optimum for devices with high energy constraints. Since the client model seems to be the most efficient energy mode for sleeping device, this document proposes to define a new intermediary called Mirror Proxy whose role is to make a sleeping device appears like any other web server in the network. On that point, a Mirror Proxy is similar to a caching reverse proxy except that there is no origin server but rather an "origin" client. So the Mirror Proxy serves content and also advertises resources on behalf of its registered endpoints.

This document defines the REST interface required to support the Mirror Proxy function on a constrained web server.

2. Requirements Language

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 RFC 2119 [RFC2119].

This specification requires readers to be familiar with all the terms and concepts that are discussed in [RFC5988] and [I-D.shelby-core-resource-directory]. Readers should also be familiar with the terms and concepts discussed in [I-D.ietf-core-coap] and [I-D.ietf-core-link-format]. This specification makes use of the following additional terminology:

Sleeping endpoint (SEP): A sleeping network node that participates in a constrained REST environment but can only act as a CoAP client endpoint due to limited energy budget.

Mirror Proxy (MP): A web entity that caches, serves and advertises content on behalf of a sleeping endpoint.

3. Architecture

The Mirror Proxy architecture is shown in Figure 1. A Mirror Proxy (MP) hosts resources in its own resource tree on behalf of other web entities which are called sleeping endpoint (SEP). For energy efficiency a SEP is a client-only CoAP endpoint and hence is not able to serve content by itself. The MP implements REST interfaces

allowing a SEP to maintain a set of cached resources that will be served in turn by the MP. A CoAP client discovers resources from the SEP hosted on the MP using traditional mechanisms such as /.well-known/core [I-D.ietf-core-link-format] or Resource Directory [I-D.shelby-core-resource-directory].

SEP are assumed to proactively register and maintain mirror entries on the MP, which are soft state and need to be periodically refreshed. A SEP is provided with interfaces to register, update and remove a mirror entry with associated resources. Furthermore, a mechanism to discover a MP using the CoRE Link Format is defined.

```
                  Registration          Mapping
        +-----+          |                 |
        | SEP |----      |                 |     +--------+
        +-----+    ----  |             ....|.....|   RD   |
                       --|-    +------+.   |     +--------+
        +-----+          | ----|      |    |     +--------+
        | SEP | ---------|-----|  MP  |----|-----| Client |
        +-----+          | ----|      |    |     +--------+
                       --|-    +------+    |
        +-----+    ----  |                 |
        | SEP |----      |                 |
        +-----+
```

Figure 1: Mirror Proxy architecture

3.1. Resource mapping in a Mirror Proxy

The resources that a SEP wishes to be served are described using link-format [I-D.ietf-core-link-format]. The description is identical to the /.well-known/core resource found on a typical CoRE web server. Upon successful registration a MP allocates a mirror entry resource for a SEP. The resources specified by the SEP during registration are created as sub-resources of the mirror entry on the MP. The MP updates its own /.well-known/core resource to reflect the changes in its resource tree. When the web server of the MP registers its resources in a Resource Directory (RD), the MP must also register the resources of the SEP with the RD. A MP MUST register the resources of a SEP in a separate resource directory entry. Once a mirror entry has expired, the MP deletes the resources associated to that entry, clears the content cache accordingly and finally updates its /.well-known/core resource. The RD and MP entries are supposed to have the same lifetime so the MP don't need to explicitly delete the RD entry except when the SEP uses the Removal interface.

Once the resources have been created on the MP, the SEP can refresh

the content of its resources at its own pace. The SEP updates the cached content on the MP using PUT requests. The SEP may also poll its writable resources using GET requests. When a SEP registers with a MP, the REST interface defined with the Interface Description (if) attribute is only valid from a client point of view. A Mirror Proxy MUST accept PUT requests coming from a SEP even if the Interface Description attribute doesn't allow this method.

The MP may accept to establish an observation relation between a mapped resource from a SEP and a client using [I-D.ietf-core-observe].

3.2. Cache refresh strategies

For non periodic cache refresh, confirmable PUT requests are preferable for reliability. But most of the SEP can't stay awake while waiting for a response. This is especially true when the MP is not in the direct neighborhood of the SEP and the latency is higher. A link layer (L2) like 802.15.4 supports low-level buffering that allows a SEP to poll its parent router for incoming traffic and sleep between poll requests. Figure 2 is an example of reliable cache refresh with L2 buffering.

```
   SEP                                 Router          MP
    | [sleep]                             |            |
    |                                     |            |
    | --- (CON) PUT /mp/0/val  -------->  | ---PUT---> |
    | [sleep]                             |            |
    |                                     |            |
    | --- L2 poll request ------------->  |            |
    | [sleep]                             |            |
    |                                     |            |
    | [CoAP retransmissions if necessary] |            |
    |                                     |            |
    | --- L2 poll request ------------->  |            |
    | [sleep]             [L2 buffering]  | <--2.04--- |
    |                                     |            |
    | --- L2 poll request ------------->  |            |
    | <-- (ACK) 2.04 Changed -----------  |            |
    | [sleep]                             |            |
    |                                     |            |
```

Figure 2: Confirmable cache refresh with L2 buffering

When no L2 buffering is available another solution is to send a burst of non confirmable PUT requests and come back to sleep mode without waiting for a response. Unfortunately there is currently no option in CoAP to indicate that the client endpoint is not interested in the

response to the request. Figure 3 shows that the MP generates "2.04 Changed" even if the client endpoint is not able to process them because it is sleeping.

```
   SEP                                                   MP
    | [long sleep]                                        |
    |                                                     |
    | --- (NON) PUT /mp/0/val ---------------------->     |
    | [short sleep]                                       |
    |                  X-- (NON) 2.04 Changed -----       |
    | --- (NON) PUT /mp/0/val ---------------------->     |
    | [short sleep]                                       |
    |                  X-- (NON) 2.04 Changed -----       |
    | --- (NON) PUT /mp/0/val ---------------------->     |
    | [long sleep]                                        |
    |                  X-- (NON) 2.04 Changed -----       |
    |                                                     |
```

Figure 3: Non-confirmable cache refresh without response suppression

Figure 4 depicts how a new suppression option in CoAP could improve network efficiency. This mechanism is also valuable for periodic refresh but in that case there is no compelling need to repeat the request at each round.

```
   SEP                                                   MP
    | [long sleep]                                        |
    |                                                     |
    | --- (NON) PUT /mp/0/val (Suppr-Rsp: All) ---->      |
    | [short sleep]                                       |
    | --- (NON) PUT /mp/0/val (Suppr-Rsp: All) ---->      |
    | [short sleep]                                       |
    | --- (NON) PUT /mp/0/val (Suppr-Rsp: All) ---->      |
    | [long sleep]                                        |
    |                                                     |
```

Figure 4: Non-confirmable cache refresh with response suppression

Note: The registration and update procedures between the SEP and the MP must also be reliable. Moreover POST requests used for registration are not idempotent so it is not possible to repeat them as non-confirmable requests. In that regard, a CoAP option indicating the sleeping constraints of a SEP might help to synchronize with a MP. Also the response suppression mechanism may delay the time needed for a SEP to detect that its mirrored resources are unreachable.

3.3. Placing a Mirror Proxy

The Mirror Proxy functionality can be distributed over multiple server endpoints in the network or centralized on a more powerful web server. The closest the Mirror Proxy is from the sleeping endpoint, the shortest is the round-trip time (RTT). A shorter RTT gives better energy efficiency for CoAP transactions. And a Mirror Proxy in the direct neighborhood of a sleeping endpoint may even avoid having to configure global connectivity for the latter. However in a wireless sensor network relying on local connectivity may result in fragility due to device mobility or radio fluctuations. This could lead a constrained endpoint to frequently try to discover another suitable Mirror Proxy. In that regard, a centralized Mirror Proxy gives more stability but usually at the expense of energy performance. A centralized Mirror Proxy also concentrates network traffic on a central point and may cause network congestion in a wireless sensor network. However data flow of a sleeping endpoint is expected to be low hence mitigating the risk of network congestion.

A sleeping endpoint MAY register with more than one Mirror Proxy but in that case the resources of a sleeping endpoint appear duplicated during resource discovery. Since there is currently no way to de-duplicate the resources, multiple registrations are discouraged.

4. Mirror Proxy interfaces

The interface is mostly identical to that of a Resource Directory [I-D.shelby-core-resource-directory] so this document only points out the differences.

4.1. Discovery

The discovery procedure is identical except that the resource type is replaced with "core-mp".

4.2. Registration

The registration interface is identical and the following additional actions are required.

The MP MUST check it has enough memory to create the resources for the new SEP before accepting the registration. If the MP is out of memory it MUST reply with a status code 5.03 "Service Unavailable". In that case the SEP SHOULD try to find another MP.

Upon successful registration, a Mirror Proxy MUST update its /.well-known/core resource to reflect the changes in its resource tree. If

the web server of a Mirror Proxy is publishing its own resources in a Resource Directory, it MUST also register the resources of the sleeping endpoint.

Since each SEP may register resources with different lifetimes, the MP MUST register each SEP as a separate resource directory entry in the RD. The MP creates different RD entries by reusing the SEP name provided during MP registration. If no name was provided, the MP can use its own name and a new Instance identifier.

4.3. Update

The update interface is identical.

Upon successful update, /.well-known/core and the Resource Directory (if applicable) MUST be updated accordingly.

4.4. Validation

The validation interface is not supported on a Mirror Proxy since the sleeping endpoint is not a server endpoint.

4.5. Removal

The removal interface is identical.

Upon successful removal, /.well-known/core and the Resource Directory (if applicable) MUST be updated accordingly.

4.6. Lookup

The lookup interface is not supported. An endpoint can discover the resources associated to a sleeping endpoint by getting the /.well-known/core resource of the Mirror Proxy or using the lookup interface of the Resource Directory if any is available.

5. Examples

The following example details the typical message flow between a SEP and a MP.

The SEP is here a light switch providing the content below:

```
</dev/>;rt="ipso:dev",
</dev/mfg >;rt="ipso:dev-mfg",
</dev/mdl>;rt="ipso:dev-mdl",
</lt/>;rt="ipso:lt",
```

```
   </lt/ctr>;rt="ipso:lt-ctr"
```

5.1. Discovery

```
   SEP                                                 MP
    |                                                  |
    | ----- GET /.well-known/core?rt=core-mp ------>   |
    |                                                  |
    |                                                  |
    | <---- 2.05 Content "</mp>; rt="core-mp" ------   |
    |                                                  |

   Req: GET coap://[ff02::1]/.well-known/core?rt=core-mp
   Res: 2.05 Content
   </mp>;rt="core-mp"
```

5.2. Registration

```
   SEP                                                 MP
    |                                                  |
    | --- POST /mp "</dev..." -------------------->    |
    |                                                  |
    |                                                  |
    | <-- 2.01 Created Location: /mp/0 -------------   |
    |                                                  |

   Req: POST coap://mp.example.org/mp?h=switch&lt=1024
   Etag: 0x3f
   Payload:
   </dev/>;rt="ipso:dev",
   </dev/mfg >;rt="ipso:dev-mfg",
   </dev/mdl>;rt="ipso:dev-mdl",
   </dev/n>;rt="ipso:dev-name",
   </lt/>;rt="ipso:lt",
   </lt/ctr>;rt="ipso:lt-ctr"

   Res: 2.01 Created
   Location: /mp/0
```

The resources below have been created on the MP.

```
   </mp/0/dev/>;rt="ipso:dev",
   </mp/0/dev/mfg >;rt="ipso:dev-mfg",
   </mp/0/dev/mdl>;rt="ipso:dev-mdl",
   </mp/0/dev/n>;rt="ipso:dev-name",
   </mp/0/lt/>;rt="ipso:lt",
   </mp/0/lt/ctr>;rt="ipso:lt-ctr"
```

Then the MP registers those new resources in the RD.

```
   MP                                                   RD
    |                                                   |
    | --- POST /rd "</mp/0..." -------------------->    |
    |                                                   |
    |                                                   |
    | <-- 2.01 Created Location: /rd/6534 ----------    |
    |                                                   |
```

```
   Req: POST coap://rd.example.org/rd?h=switch&lt=1024
   Etag: 0x6a
   Payload:
   </mp/0/dev/>;rt="ipso:dev",
   </mp/0/dev/mfg >;rt="ipso:dev-mfg",
   </mp/0/dev/mdl>;rt="ipso:dev-mdl",
   </mp/0/dev/n>;rt="ipso:dev-name",
   </mp/0/lt/>;rt="ipso:lt",
   </mp/0/lt/ctr>;rt="ipso:lt-ctr"

   Res: 2.01 Created
   Location: /rd/6534
```

5.3. **Content refresh**

Just after registration, the SEP refreshes the content of static resources such as the manufacturer name (/dev/mfg) and model name (/dev/mdl). These resources are updated with almost infinite max-age so there is no need to send more PUT requests afterward as long as the mirror proxy entry is refreshed on the MP.

```
   SEP                                                  MP
    |                                                   |
    | --- PUT /mp/0/dev/mfg "Exa..." ------------->     |
    |                                                   |
    |                                                   |
    | <-- 2.04 Changed -----------------------------    |
    |                                                   |
```

```
   Req: PUT coap://mp.example.org/mp/0/dev/mfg (Max-Age: 0xFFFFFFFF)
   Payload: Example.Com
   Res: 2.04 Changed
```

When the switch is activated the SEP wakes up and sends a request to refresh the light control resource. The Max-Age option in the request indicates the maximum time between two PUT requests even if the switch keeps the same state.

```
   SEP                                                MP
    |                                                  |
    | --- PUT /mp/0/lt/ctr "1" -------------------->   |
    |                                                  |
    |                                                  |
    | <-- 2.04 Changed ------------------------------  |
    |                                                  |
```

```
   Req: PUT coap://mp.example.org/mp/0/lt/ctr (Max-Age: 1 hour)
   Payload: 1
   Res: 2.04 Changed
```

A client may have subscribed to change of values for the resource /mp/0/lt/ctr on the MP. In that case, the MP would have to notify the CoAP client right after a PUT request from the SEP. A typical exchange would be:

```
   SEP                          MP                        Client
    |                            |                          |
    |                            | <-- GET /mp/0/lt/ctr --  |
    |                            |          (observe)       |
    |                            |                          |
    |                            | -- 2.05 Content "0" -->  |
    |                            |                          |
    | - PUT /mp/0/lt/ctr "1" ->  |                          |
    |                            |                          |
    | <- 2.04 Changed ---------  |                          |
    |                            |                          |
    |                            | -- 2.05 Content "1" -->  |
```

5.4. **Polling**

A SEP may ask a MP to mirror writable resources. In that case the SEP needs to poll the MP to detect whether the resources have been updated. The example below shows a SEP polling a writable resource once a day to update its name.

```
   SEP                          MP                        Client
    |                            |                          |
    | - GET /mp/0/dev/n ------>  |                          |
    |                            |                          |
    | <- 2.03 Valid -----------  |                          |
    |                            |                          |
    |                            | <-- PUT /mp/0/dev/n ---  |
    | [1 day]                    |                          |
    |                            | -- 2.04 Changed ------>  |
    |                            |                          |
    | - GET /mp/0/dev/n ------>  |                          |
    |                            |                          |
    | <- 2.05 Content ---------  |                          |
    |                            |                          |
```

6. TODO

We need a way to identify a SEP registered in multiple proxies as a single entity. Otherwise clients may have a wrong view of the available services during resource discovery. The duplicate resources are indeed seen as new devices.

We need a simple mechanism to define allowed methods on mirrored resources that is independant of the resource profile. A second Interface Description attribute with the CRUD letters might work.

We may add a CoAP option that would permit a MP to indicate in a response that a client has updated another writable resource in the cache for the requesting SEP. This would gives better response time than polling.

When a SEP registers with multiple MP and each MP reuses the same SEP's name to register with the RD, the resource directory entry might be overwritten. We need to figure out what piece of information is the handler for a resource directory entry in a RD database.

6.1. Extensions

Implicit registration could be useful for highly constrained SEP that don't have enough energy to maintain soft state in a MP. Like the proposition in [I-D.arkko-core-sleepy-sensors], we could use a multicast address to infer a resource type. Alternatively we could allow explicit registration from a third-party endpoint.

7. Acknowledgements

Thanks to Zach Shelby who is the author of the Resource Directory interface.

8. IANA Considerations

"core-mp" resource type needs to be registered if an appropriate registry is created.

This document suggest the creation of a new CoAP option to suppress an undesired response to a request. But this new option should be detailed in a separate draft.

9. Security Considerations

This document needs the same security considerations as described in Section 7 of [RFC5988] and Section 6 of [I-D.ietf-core-link-format].

Unrestricted access to mirrored resources may allow a malicious web client to poison the cache on a MP. A MP SHOULD authenticate the SEP and restrict allowed methods according to the interface description supplied during resource registration.

A malicious client could start a denial of service attack by trying to mirror a large resource on a MP. Memory exhaustion would prevent the other sleeping endpoints from mirroring their resources.

A malicious client could trigger the removal or the update interface on a MP to delete mirrored resources. This would cause an interruption of service for the targeted SEP until it registers again its resources. A MP SHOULD authenticate client endpoints using the interfaces that can modify a SEP description.

A MP could loose or delete the mirror proxy entry associated to a SEP without sending an explicit notification (e.g. after reboot). A SEP SHOULD be able to detect this situation by processing the response code while sending requests to the proxy. Especially an error code "4.04 Not Found" SHOULD cause the SEP to register again. A SEP MAY also register with multiple proxies to alleviate the risk of interruption of service.

10. References

10.1. Normative References

[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.

[I-D.ietf-core-observe]
Hartke, K., "Observing Resources in CoAP", draft-ietf-core-observe-04 (work in progress), February 2012.

[I-D.shelby-core-resource-directory]
Krco, S. and Z. Shelby, "CoRE Resource Directory", draft-shelby-core-resource-directory-02 (work in progress), October 2011.

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

[RFC5988] Nottingham, M., "Web Linking", RFC 5988, October 2010.

10.2. Informative References

[I-D.arkko-core-sleepy-sensors]
Arkko, J., Rissanen, H., Loreto, S., Turanyi, Z., and O. Novo, "Implementing Tiny COAP Sensors", draft-arkko-core-sleepy-sensors-01 (work in progress), July 2011.

Author's Address

Matthieu Vial
Schneider-Electric
Grenoble,
FRANCE

Phone: +33 (0)47657 6522
Email: matthieu.vial@schneider-electric.com