CoRE Working Group
Internet-Draft
Intended status: Informational
Expires: September 15, 2011

A. Castellani
University of Padova
S. Loreto
Ericsson
March 14, 2011

Best Practice to map HTTP to COAP and viceversa
draft-castellani-core-http-coap-mapping-01.txt

Abstract

This draft aims at being a simple guide to the use of CoAP REST interface, to show how it can be mapped to and from HTTP, and at being a base reference documentation for CoAP/HTTP proxy implementors.

Table of Contents

1. Introduction

Since implementing on constrained devices the full HyperText Transfer Protocol (HTTP) [RFC2616] is believed to be operationally and computationally too complex, especially in an M2M communication environment, resources available on constrained nodes are expected to be served using CoAP [I-D.ietf-core-coap].

> "The interaction model of CoAP is similar to the client/server model of HTTP. However, machine-to-machine interactions typically result in a CoAP implementation acting in both client and server roles (called an end-point). A CoAP request is equivalent to that of HTTP, and is sent by a client to request an action (using a method code) on a resource (identified by a URI) on a server. The server then sends a response with a response code; this response may include a resource representation." Section 2 [I-D.ietf-core-coap]

These days the information is increasingly converging on the Web, thus an easy CoAP interoperability with HTTP is a paramount feature for CoAP. Indeed leveraging on both the easy CoAP/HTTP translation and the common usage of URI(s) to identify resources, it will become extremely simple to integrate constrained nodes in the Web.

The internetworking described in this document between CoAP and HTTP is mainly based on three points:

- the URI does not change between CoAP and HTTP, the scheme identifies the protocol;
- HTTP/CoAP mapping is performed by a proxy, both HTTP/CoAP endpoints can be not aware that a mapping is happening;
- using a named URI authority and DNS can be useful for the mapping.

The proxy itself does not require any particular knowledge about the constrained network topology, devices contained, nor about the content of data exchanged.

2. HTTP-CoAP

HTTP-CoAP mapping spans across several protocol layers:

- HTTP is mapped to CoAP
- TCP is used on the HTTP side, while CoAP uses UDP transport

In addition to this 6LoWPAN adaptation layer addresses a similar networking scenario, thus a convertion between IPv4/IPv6 to 6LoWPAN MAY be present as well.

2.1. URI

Any resource available in CoAP can be accessed using HTTP at the same URI, except for the scheme. The scheme represents the protocol used by the endpoint to access the resource.

The CoAP resource "//node.coap.something.net/foo" can be accessed using CoAP at the URI "coap://node.coap.something.net/foo", and using HTTP at the URI "http://node.coap.something.net/foo". When the resource is accessed using HTTP, the mapping from HTTP to CoAP is performed by a proxy

The usage of the same URI to access a resource, independently if it is accessed by a CoAP client within the same constrained network or by a HTTP client outside the constrained network, reduces the complexity of a proxy performing the mapping.

OPEN ISSUE: discuss the DNS usage resolving the URI.

2.2. Proxy

A device providing cross-protocol HTTP-CoAP mapping is called HTTP-CoAP cross-protocol proxy (HC proxy).

Usually regular HTTP proxies are same-protocol proxies, because can map from HTTP to HTTP. CoAP same-protocol proxies are intermediaries for CoAP to CoAP exchanges, however the discussion about that entities is out-of-scope of this document.

At least two different kinds of HC proxies may exist:

- One-way cross-protocol proxy (1-way proxy): It can translate from a client of a protocol to a server of another protocol but not viceversa.
- Two-way (or bidirectional) cross-protocol proxy (2-way proxy): It can translate from a client of both protocols to a server of the other protocol.

1-way and 2-way HC proxies can be realized using the following general types of proxies:

Forward proxy (F): It is a proxy known by the client (either CoAP or HTTP) used to access a specific cross-protocol server (respectively HTTP or CoAP). Main feature: server(s) do not require to be known in advance by the proxy (ZSC: zero server configuration).

Reverse proxy (R): It is known by the client to be the server, however for a subset of resources it works as a proxy, by knowing the real server(s) serving each resource. When a cross-protocol resource is accessed by a client, the request will be silently forwarded by the reverse proxy to the real server (running a different protocol). If a response is received by the reverse proxy, it will be mapped, if possible, to the original protocol and sent back to the client. Main feature: client(s) do not require to be known in advance by the proxy (ZCC: zero client configuration).

Transparent (or Intercepting) proxy (I): This proxy can intercept any origin protocol request (HTTP or CoAP) and map it the destination protocol, without any kind of knowledge about the client or server involved in the exchange. Main feature: client(s) and server(s) do not require to be known in advance by the proxy (ZCC and ZSC).

The proxy can be placed in the network at three different logical locations:

Server-side proxy (SS): a proxy placed on the same network domain of the server;

Client-side proxy (CS): a proxy placed on the same network domain of the client;

External proxy (E): a proxy placed in a network domain external to both endpoints.

In the most common scenario the HC proxy is expected to be server-side and deployed at the edge of the constrained network. The arguments supporting this assumption are the following:

TCP/UDP: Translation between HTTP and CoAP requires also a TCP to UDP mapping; UDP performance over the Internet may not be adequate, UDP should be dropped as soon as possible to minimize the number of required retransmissions and overall reliability.

Multicast: To enable access to local-multicast in the constrained network, the HC proxy may require a network interface directly attached to the constrained network.

Caching: Efficient caching requires that all the CoAP traffic is intercepted by the same proxy, network edge is a strategical placement for this need.

Security: HTTPS sessions should be terminated as near as possible to the CoAP server.

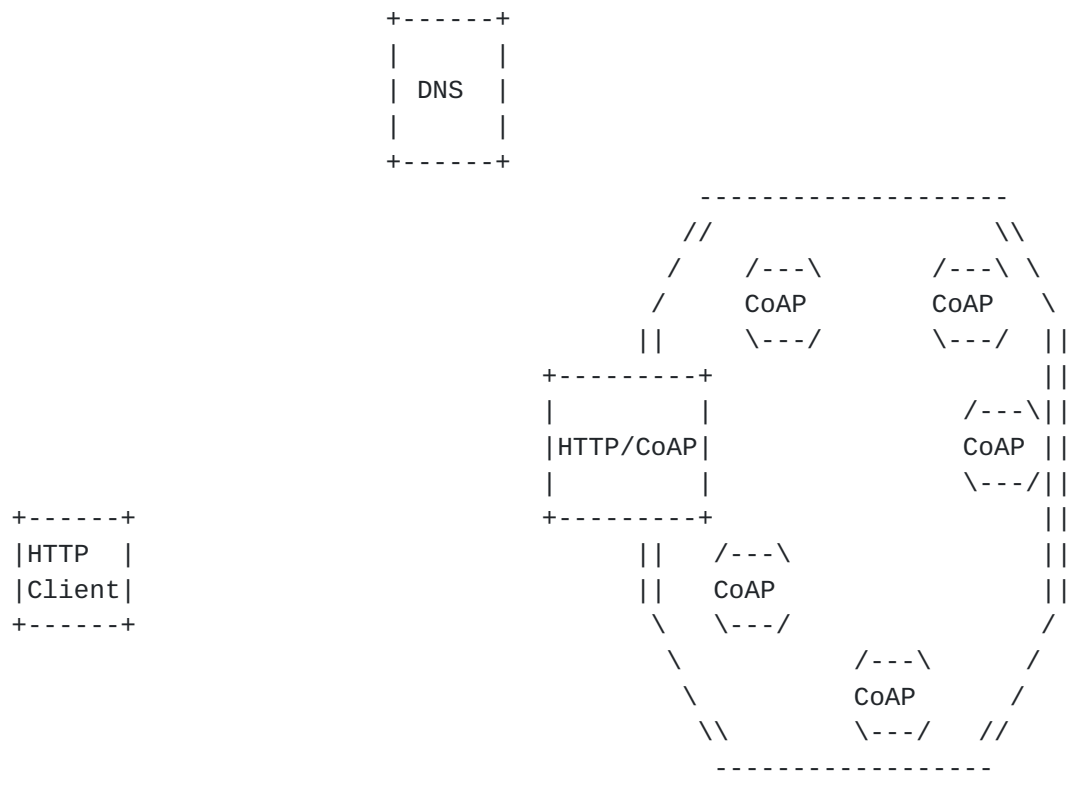

Table 1 shows some interesting HC proxy scenarios, and quickly marks the advantages related to each scenario.

Feature	F CS	R SS	I SS
TCP/UDP	-	+	+
Multicast	-	+	+
Caching	-	+	+
Security	?	+	-
Scalability	+	?	+
Configuration	-	-	+

Table 1: Interesting HC proxy deployments

The following open questions are left open in Table 1:
1. Are CoAP security modes adequate for Internet-wide operation?
2. Are reverse proxy setups scalable?

2.2.1. HC proxy discovery using DNS-SD

DNS-SD can be used by an HTTP client to discover the HC proxy in authority for a specific domain [I-D.jennings-http-srv].

An HTTP client wants access a resource that it knows being identified by the following URI:

//node.coap.something.net/foo

To find the address of the HC proxy, the HTTP client will look up the following SRV record:

_http._tcp.node.coap.something.net

The DNS will contain the following record:

```
    _http._tcp.node.coap.something.net  IN  SRV     0  1  80  hc-
proxy.something.net
      hc-proxy.something.net  IN  A  192.168.0.1  ;  the address of the HC
proxy
```

The client will pass the request to the HC proxy that will translate it in a CoAP request. The CoAP side of the proxy will lookup the DNS in order to find the actual constrained device in authority for that URI.

2.3. Mapping

CoAP offers a subset of HTTP features in terms of methods, statuses and options supported; thus some HTTP request MAY NOT be mappable to CoAP.

In particular CoAP lacks the following methods defined in HTTP: OPTIONS, HEAD, TRACE and CONNECT.

An HC proxy receiving an HTTP request with a method not supported in CoAP MUST immediately drop handling the request and MUST send a response with status "405 Method Not Allowed" to the HTTP client.

The mapping of a CoAP response code to HTTP is not straightforward, this mapping MUST be operated accordingly to Table 4 of [I-D.ietf-core-coap].

The mapping of conditional HTTP requests is defined in Section 8.2 of [I-D.ietf-core-coap].

An HC proxy MUST always try to resolve the URI authority, and SHOULD prefer using the IPv6 resolution if available. The authority section of the URI is thus used internally by the HC proxy and SHOULD not be mapped to CoAP.

If an empty CoAP ACK is received, the actual CoAP response is

deferred. As described in CoAP specification the ACK is transparent to the HTTP client.

No upper bound is defined for a server to provide that response, thus for long delays the HTTP client or any other proxy in between MAY timeout, further considerations are available in Section 7.1.4 of [I-D.ietf-httpbis-p1-messaging].

If the HTTP client times out and drops the HTTP session to the proxy (closing the TCP connection), the HC proxy SHOULD wait for the response and cache it if possible. Further idempotent requests to the same resource can use the result present in cache, or if a response has still to come requests will wait on the open CoAP session.

Safe or non-idempotent requests MAY timeout. How the HC proxy should handle this situation?

The HC proxy MUST define an internal timeout for each CoAP request pending, because the CoAP server MAY silently die before completing the request. This timeout SHOULD be as high as possible.

Figure 2 shows an HTTP client on IPv4 (C) accessing a CoAP server on IPv6 (S) through an HC proxy on IPv4/IPv6 (P). node.coap.something.net has an A record containing the IPv4 address of the HC proxy, and an AAAA record containing the IPv6 of the CoAP server.

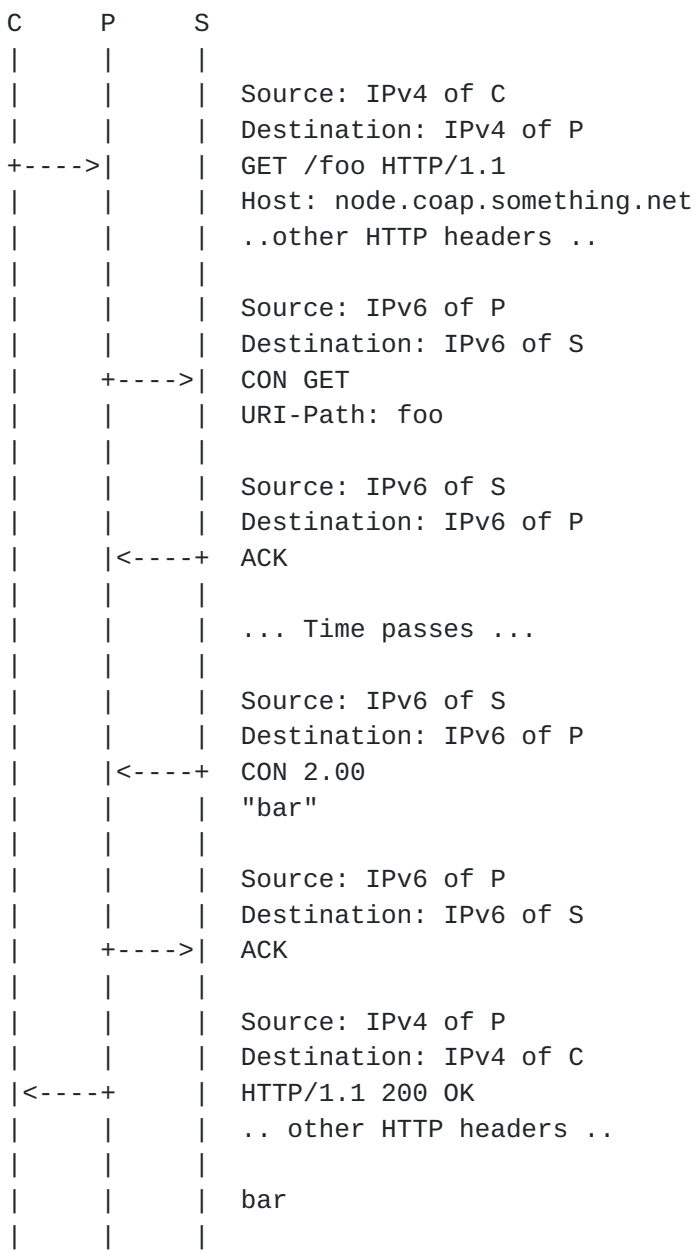

Figure 2: HTTP/IPv4 to CoAP/IPv6 mapping

The proposed example shows the HC proxy operating also the mapping between IPv4 to IPv6 using the authority information available in any HTTP 1.1 request. Thus IPv6 connectivity is not required at the HTTP client when accessing a CoAP server over IPv6 only, which is a typically expected use case.

When P is an intercepting HC proxy, the CoAP request SHOULD have the IPv6 address of C as source (IPv4 can always be mapped into IPv6).

When the HTTP client has native IPv6 support, a convenient deployment choice should be to use an HC intecepting proxy. Thus the proxy MUST be located in the IPv6 network path between the client and the server, thus near to the server itself in order to support any Internet client.

2.4. Multiplexing CoAP responses

Defining the mapping of some advanced CoAP features to HTTP (i.e. multicast, observe) must address the need to asynchronously deliver multiple responses to the same HTTP request.

Some HTTP features are useful to succesfully represent these particular sessions.

Using Multipart media type is a suitable solution to deliver multiple CoAP responses within a single HTTP response.

Each part of a multipart entity SHOULD be represented using "message/http" media type containing the full mapping of a single CoAP response as previously described.

An HC proxy may prefer to transfer each CoAP response immediately after its reception. Responses can be immediately transferred in "chunks" of an HTTP chunked Transfer-Encoding session, without knowing in advance the total number of responses and with arbitrary delay between them.

A detailed discussion on the use of chunked Transfer-Encoding to stream data over HTTP can be found in [I-D.loreto-http-bidirectional]. Large delays between chunks can lead the HTTP session to timeout, more details on this issue can be found in [I-D.thomson-hybi-http-timeout].

When responses are coming from different sources, i.e. multicast, details about the actual source of each CoAP response SHOULD be provided. Source information can be represented in HTTP using a Link option described in [RFC5988] using "via" relation type.

Figure 3 shows an HTTP client (C) requesting the resource "/foo" to a group of CoAP servers (S1/S2/S3) through an HC proxy (P). Discussion related to group communication in CoAP can be found in [I-D.rahman-core-groupcomm].

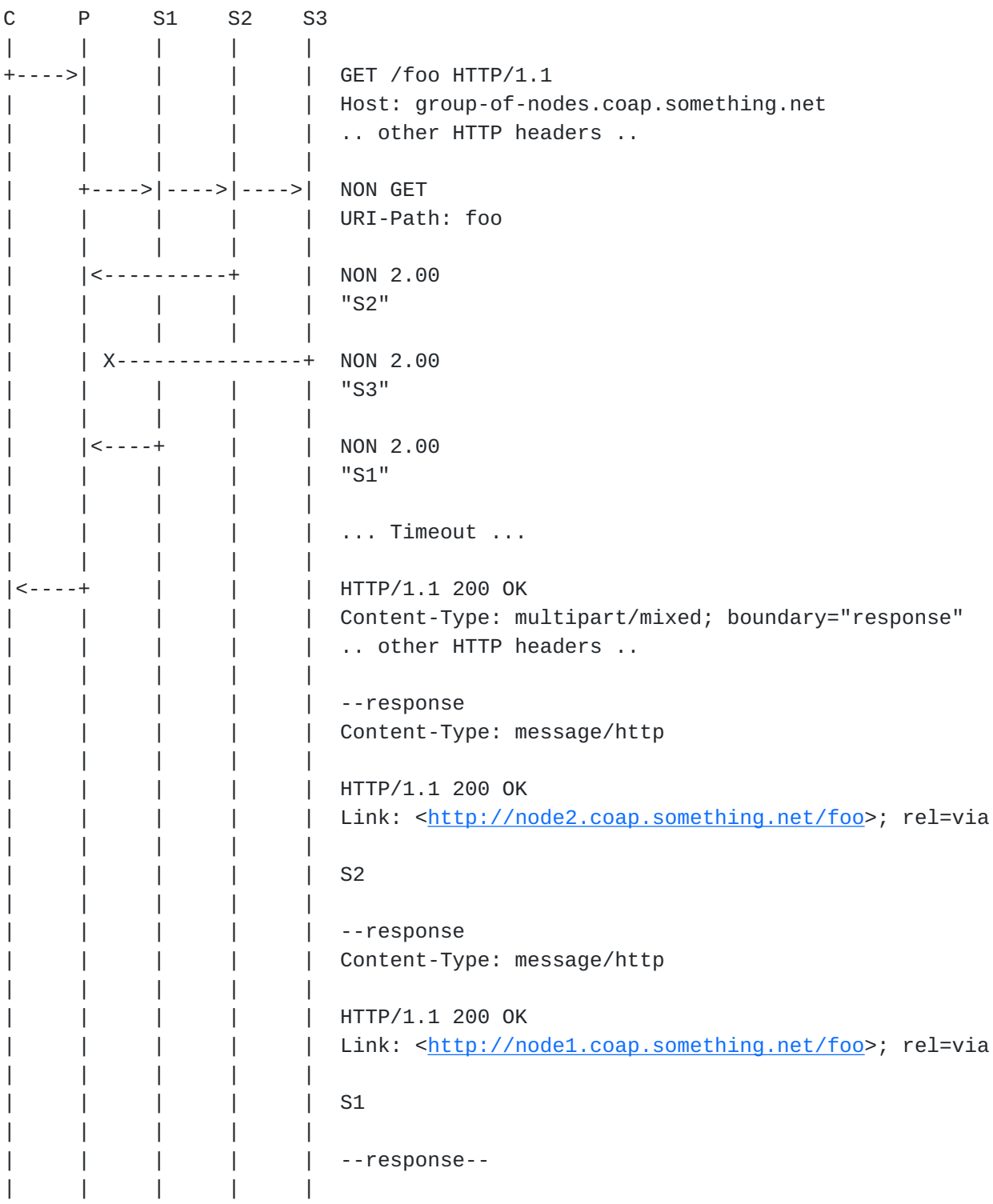

Figure 3: Unicast HTTP to multicast CoAP mapping

The mapping proposed in the above diagram does not make any assumption in how multicasting is done on the constrained network.

If IPv6 multicast support is present in the constrained network, an AAAA record containing the IPv6 multicast group will start multicast operation at the proxy. Otherwise the authority part of the URI is used by the HC proxy to match with a locally defined group of nodes.

In order to minimize the delay in delivering the responses (e.g. HTTP client can incrementally process the responses, HC proxy can reduce internal buffering), each CoAP response can be immediately streamed using HTTP chunked Transfer-Encoding. This encoding was not shown in order to simplify Figure 3, an example showing immediate delivery of CoAP responses is provided in Figure 4 (observe session).

2.4.1. Establishing a CoAP subscription

Using an exchange similar to the one shown in Figure 3, a CoAP observe session can be directly established by a willing HTTP client. Observe mechanism is specified in [I-D.ietf-core-observe].

An HTTP client willing to establish a subscription to the "/temperature" resource of a CoAP server SHOULD send an HTTP request with Expect header set to "206" and Accept header set to "multipart/mixed".

The Lifetime of the subscription itself SHALL be sent defining the subscription interval using "Date:" header as starting time and "If-Modified-Since:" as ending time. The HC proxy can compute Lifetime option by using that HTTP headers.

Due to the asynchronous nature of this exchange, the HC proxy willing to accept establishing a subscription SHOULD send an HTTP response with status "206 Partial Content", Content-Type "multipart/mixed" and Transfer-Encoding "chunked".

Each CoAP response will be delivered in a different HTTP chunk until the subscription lifetime expires, when the subscription has expired the HTTP session MUST be closed.

If the HC proxy does not support this exchange or is not willing to establish this session, it SHOULD fail with status "417 Expectation failed".

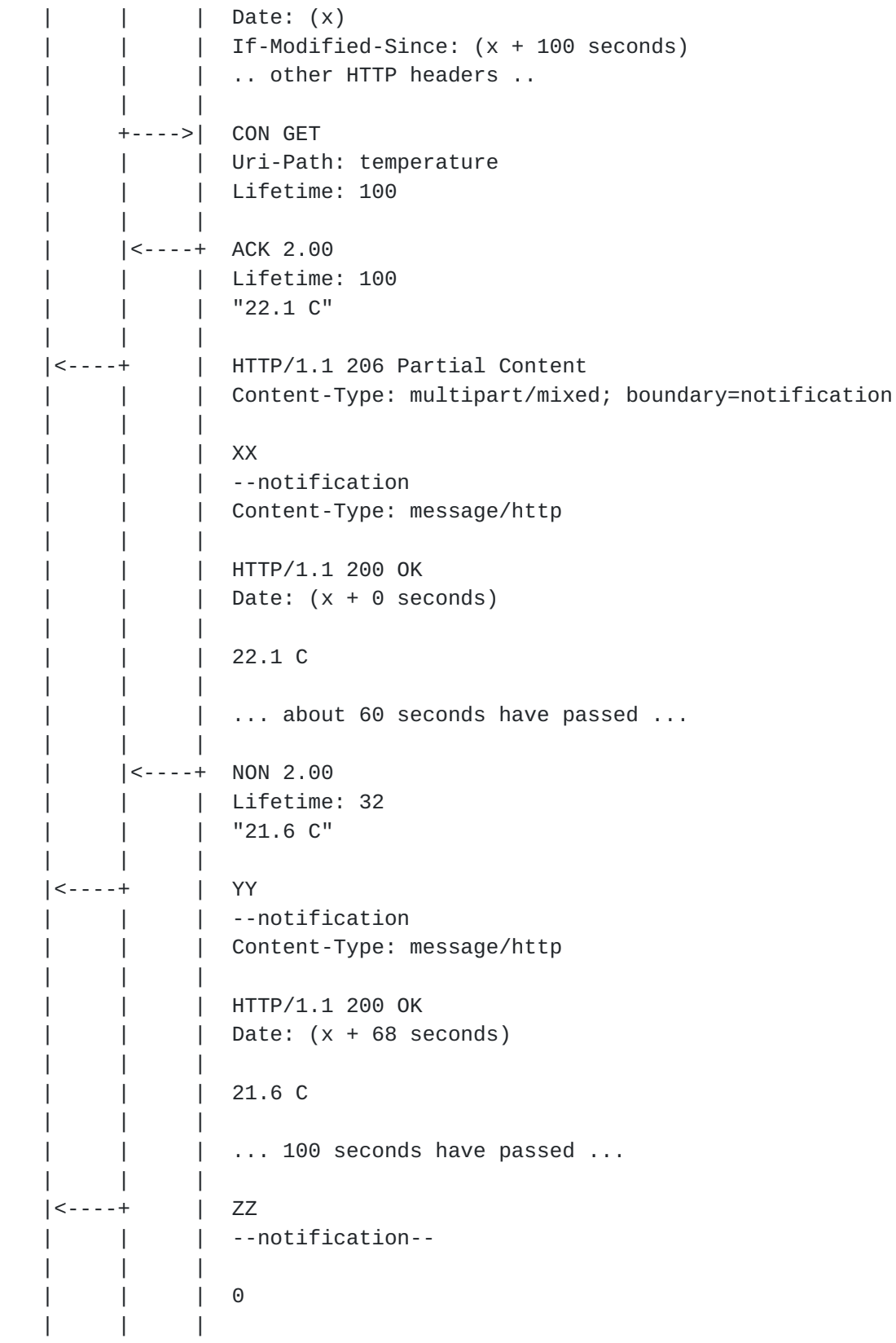
Date: (x)
If-Modified-Since: (x + 100 seconds)
.. other HTTP headers ..
CON GET
Uri-Path: temperature
Lifetime: 100
ACK 2.00
Lifetime: 100
"22.1 C"
HTTP/1.1 206 Partial Content
Content-Type: multipart/mixed; boundary=notification
XX
--notification
Content-Type: message/http
HTTP/1.1 200 OK
Date: (x + 0 seconds)
22.1 C
... about 60 seconds have passed ...
NON 2.00
Lifetime: 32
"21.6 C"
YY
--notification
Content-Type: message/http
HTTP/1.1 200 OK
Date: (x + 68 seconds)
21.6 C
... 100 seconds have passed ...
ZZ
--notification--
0

Figure 4: HTTP subscription to a CoAP resource

When an HTTP client performs direct subscriptions to CoAP servers using this method, the HC proxy has to keep for a possibly long time state information about the observe session and an open HTTP/TCP session to the client.

Soft state required by the various involved protocols (HTTP/TCP, CoAP/UDP) leads to scalability issues when an high number of direct subscriptions are established using the same HC proxy.

Moreover the HC proxy has an active role in the subscription process, thus if crashed or rebooted the subscription to the CoAP node will be lost.

HTTP clients in the real world usually implement notification mechanisms over HTTP using a technique called "Long Polling", an extensive description of this technique is available in Section 2 of [I-D.loreto-http-bidirectional]. A mapping using a "Long Polling" may be identified and can be preferred for longer sessions of observe.

3. CoAP-HTTP

TBD

4. Security Considerations

TBD

5. IANA Considerations

This document does not require any actions by the IANA.

6. Acknowledgements

Special thanks to Nicola Bui and Michele Zorzi for their support and for the various contributions to this document.

Thanks to Kerry Lynn, Akbar Rahman, Peter van der Stok and Anders Brandt for the extensive fruitful discussion about URI mapping, DNS and multicast group communication useful for writing various sections of this document.

Thanks to Brian Frank for its support and its feedback about the content.

7. References

7.1. Normative References

[RFC2616] Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., and T. Berners-Lee, "Hypertext Transfer Protocol -- HTTP/1.1", RFC 2616, June 1999.

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

[I-D.ietf-httpbis-p1-messaging] Fielding, R., Gettys, J., Mogul, J., Nielsen, H., Masinter, L., Leach, P., Berners-Lee, T., and J. Reschke, "HTTP/1.1, part 1: URIs, Connections, and Message Parsing", draft-ietf-httpbis-p1-messaging-12 (work in progress), October 2010.

[I-D.ietf-core-coap] Shelby, Z., Hartke, K., Bormann, C., and B. Frank, "Constrained Application Protocol (CoAP)", draft-ietf-core-coap-04 (work in progress), January 2011.

[I-D.ietf-core-observe] Hartke, K. and Z. Shelby, "Observing Resources in CoAP", draft-ietf-core-observe-01 (work in progress), February 2011.

7.2. Informative References

[I-D.loreto-http-bidirectional] Loreto, S., Saint-Andre, P., Salsano, S., and G. Wilkins, "Known Issues and Best Practices for the Use of Long Polling and Streaming in Bidirectional HTTP", draft-loreto-http-bidirectional-07 (work in progress), January 2011.

[I-D.thomson-hybi-http-timeout] Thomson, M., Loreto, S., and G. Wilkins, "Hypertext Transfer Protocol (HTTP) Timeouts", draft-thomson-hybi-http-timeout-00 (work in progress), March 2011.

[I-D.jennings-http-srv] Jennings, C., "DNS SRV Records for HTTP",

draft-jennings-http-srv-05 (work in progress), March 2009.

[I-D.rahman-core-groupcomm]
Rahman, A., "Group Communication for CoAP",
draft-rahman-core-groupcomm-04 (work in progress),
March 2011.

Authors' Addresses

Angelo P. Castellani
University of Padova
Via Gradenigo 6/B
Padova 35131
Italy

Email: angelo@castellani.net

Salvatore Loreto
Ericsson
Hirsalantie 11
Jorvas 02420
Finland

Email: salvatore.loreto@ericsson.com