Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: January 2, 2015

M. Koster
ARM Limited
A. Keranen
Ericsson

July 1, 2014

Message Queueing in the Constrained Application Protocol (CoAP)
draft-koster-core-coapmq-00.txt

Abstract

The Constrained Application Protocol, CoAP, and related extensions are intended to support machine-to-machine communication in systems where one or more nodes are resource constrained, in particular for low power wireless sensor networks. This document defines publish-subscribe message queuing functionality for CoAP that extends the capabilities for supporting nodes with long breaks in connectivity and/or up-time.

Status of This Memo

This Internet-Draft will expire on January 2, 2015.

Copyright Notice

Table of Contents

1. Introduction

IETF CoRE supports machine to machine communication across networks of constrained devices. One important class of constrained devices includes devices that are intended to run for years from a small battery, or by scavenging energy from their environment. These devices spend most of their time in a sleeping state with no network connectivity.

Devices may also have limited reachability due to certain middle-boxes, such as Network Address Translators (NATs) or firewalls. Such devices must communicate using a client role, whereby the endpoint is responsible for initiating communication.

This document specifies the means for nodes with limited reachability to communicate using simple extensions to CoAP and the CoRE Resource Directory [I-D.ietf-core-resource-directory-01]. The extensions enable publish-subscribe communication using a Message Queue (MQ) broker node that enables store-and-forward messaging between two or more nodes.

2. Terminology

The key words 'MUST', 'MUST NOT', 'REQUIRED', 'SHALL', 'SHALL NOT', 'SHOULD', 'SHOULD NOT', 'RECOMMENDED', 'MAY', and 'OPTIONAL' in this specification are to be interpreted as described in [1].

This specification requires readers to be familiar with all the terms and concepts that are discussed in [6] and [2]. Readers should also be familiar with the terms and concepts discussed in [5] and [4]. The URI template format, see [3], is used to describe the REST interfaces defined in this specification.

This specification makes use of the following additional terminology:

Publish-Subscribe: CoapMQ supports publish-subscribe pattern interactions, where an endpoint uses the core.mq function server as a broker, sending updates to be buffered and sent to zero or more subscribers. Likewise, an endpoint may register with a core.mq server to receive buffered updates published to the core.mq server by other endpoints. There is a simple binding of operations to pub-sub protocol operations, for example MQTT publish and subscribe operations.

TOpic A Topic, in Publish-Subscribe systems, is a unique identifying string for a particular item or object being published and subscribed to.

The following entities are used in this specification:

CoAP Message Queue (CoAP-MQ) Service A service provided by a node or system where CoAP messages sent by one endpoint to another are queued (stored) by intermediate node(s) and forwarded only when suitable, e.g., when the message recipient endpoint is not sleeping.

CoAP-MQ Broker A server node capable of storing messages to and from other nodes and able to match subscriptions and publications in order to route messages to right destinations.

CoAP-MQ Endpoint An endpoint that implements the MQ function set defined in Section 5. A CoAP-MQ endpoint has two potential roles, CoAP-MQ client and CoAP-MQ server

3. Architecture

3.1. RD Server with MQ function set

Figure 1 shows an example architecture of a CoAP-MQ capable service. A RD service accepts registrations and registration updates from endpoints and hosts a resource discovery service for web application clients. State information is updated from the endpoints to the core.mq function server. Web clients subscribe to the state of the endpoint from the CoAP-MQ broker, and publish updates to the endpoint state through the CoAP-MQ broker. The CoAP-MQ broker performs a store-and-forward function between the web client and the CoAP-MQ capable endpoints.

```
    Endpoints                  Service                   Applications
                              +------+
                              |      |
               +- register -> |  RD  | <- discover -+
    +------+   |              |      |              |   +--------+
    |      | --+              +------+              +-- |  Web   |
    |  EP  |                                            | Client |
    |      | <-+              +------+              +-> |  app   |
    +------+   |              |      |              |   +--------+
     |  EP  |  +-- pub/sub -> |  MQ  | <- pub/sub --+    |  app   |
     +------+                 |      |                   +--------+
                              +------+
```

Figure 1: CoAP MQ Architecture

3.2. Client Endpoint

Client endpoints initiate all interactions with the RD and MQ broker. If the endpoint is an actuator it will need to either use CoAP Observe [I-D.ietf-core-observe] or periodically poll the MQ broker to check for updates. A client endpoint would use CoAP PUT operations to update its state on the MQ broker. An endpoint updates the RD periodically to indicate that it is still alive even if it has no pending data updates. Endpoints can operate in client mode even if not directly reachable from the CoAP-MQ broker or RD.

3.3. Server endpoint

Server endpoint interactions require the CoAP-MQ broker to perform the client role, initiating interaction with the server endpoint. The CoAP-MQ client MAY then use PUT operations to update state at the server endpoint, and MAY use GET or GET+Observe to subscribe to resources at the endpoint. Server mode endpoints are required to be reachable from core.mq function servers. In a network containing both client and server endpoints, client endpoints MAY subscribe to server endpoints directly, in broker-less configurations.

3.4. Publish-Subscribe Topics

Topic are strings used to identify particular resources and objects in publish-subscribe systems. Topics are conventionally formed as a hierarchy, e.g. "/sensors/weather/barometer/pressure". Implementations MAY map topics to resources, reusing existing resource addressing schemes.

4. Registration and discovery - RD server

An endpoint wishing to use the CoapMQ protocol registers with an RD server that advertises the core.mq function set. The endpoint registers topics with the core.pubsub attribute to indicate intention to use the CoapMQ protocol.

4.1. Register PubSub Endpoint

Figure 2 shows the flow of the registration operation. Discovery proceeds as per Resource Directory. When an endpoint wishes to use the CoapMQ protocol, it discovers the "core.mq" function set at the RD service and registers its CoAP-MQ resources with the RD server by creating a RD endpoint and updating it with resources having the rt="core.pubsub" attribute. Topics are created using an initial POST operation to the registered topic or sub-topic. For example, if the registered topic is "/sensors/weather", the sub-topic "/sensors/weather/barometer" is created using POST to "/mq/sensors/weather/barometer". An implementation MAY mix CoAP-MQ resources and CoAP REST resources on the same endpoint. Endpoint registration proceeds as per normal RD registration.

```
   EP                                               MQ        RD
    |                  MQ DISCOVERY                 |         |
    | -------- GET /.well-known/core?rt=core.mq --- | ------> |
    |                                               |         |
    | <-------- 2.05 Content "</mq>; rt=core.mq"--- | ------- |
    |                                               |         |
    |                                               |         |
    |                TOPIC REGISTRATION             |         |
    | ----POST /rd "</mq/0/xx>;rt=core.pubsub" ---- | ------> |
    |                                               |         |
    | <-------- 2.01 Created Location: /rd/1234 --- | ------- |
    |                                               |         |
    |                                               |         |
    |                 FIRST PUBLISH                 |         |
    | ---------------- POST /mq/0/... ------------> |         |
    |                                               |         |
    | <--------------- 2.01 Created---------------- |         |
    |                                               |         |
```

Figure 2: Discovery and Registration Message Exchange

4.2. Unregister Endpoint

CoapMQ endpoints indicate the end of their registration tenure by either explicitly unregistering, as in Figure 3, or allowing the lifetime of the previous registration to expire, as in Figure 3.

```
   EP                                               MQ        RD
   |                                               |         |
   |                    UNREGISTER                 |         |
   | ---------------- DELETE /rd/1234 ------------ | ------> |
   |                                               |         |
   | <-------- 2.02 Deleted Location: /rd/1234 --- | ------- |
   |                                               |         |
   |                                               |         |
```

Figure 3: Unregister Endpoint Message Exchange

5. CoRE MQ Function Set

5.1. Client Endpoint Functions

5.1.1. Endpoint publish to CoAP-MQ broker

Client endpoint uses PUT to publish updates to the state associated with the topic at the CoAP-MQ broker.

```
   EP                                              MQ        RD
   |                                               |         |
   |                                               |         |
   |                    PUBLISH                    |         |
   | ----------------- PUT /0/... ---------------> |         |
   |                                               |         |
   |                                               |         |
   | <--------------- 2.04 Changed---------------- |         |
   |                                               |         |
   |                                               |         |
```

Figure 4: PUBLISH from EP Message Exchange

5.1.2. Endpoint subscribe to CoAP-MQ broker using GET+Observe, CoAP-MQ broker publishes notifications to the endpoint

Client mode endpoint subscribes to the topic at the CoAP-MQ broker using GET+Observe. Published updates to the CoAP-MQ broker are published to the Endpoint using Observe response tokens. Client endpoint MAY update actuator or resource based on received values associated with responses.

```
   EP                                              MQ        RD
   |                                               |         |
   |                                               |         |
   |                 SUBSCRIBE                     |         |
   | ------- GET /0/... Observe: Token:XX -------> |         |
   |                                               |         |
   |                  PUBLISH                      |         |
   | <---------- 2.05 Content Observe:10---------- |         |
   |                                               |         |
   |                  PUBLISH                      |         |
   | <---------- 2.05 Content Observe:12---------- |         |
   |                                               |         |
   |                  PUBLISH                      |         |
   | <---------- 2.05 Content Observe:15---------- |         |
   |                                               |         |
   |                                               |         |
```

Figure 5: Endpoint SUBSCRIBE, PUBLISH to endpoint

5.1.3. Endpoint GET from CoAP-MQ broker

Client mode endpoint MAY issue GET to topic without Observe as needed to obtain last published state from the CoAP-MQ broker.

```
   EP                                              MQ        RD
   |                                               |         |
   |                                               |         |
   |                                               |         |
   | ---------------- GET /0/... ----------------> |         |
   |                                               |         |
   |                                               |         |
   | <--------------- 2.05 Content --------------- |         |
   |                                               |         |
   |                                               |         |
```

Figure 6: GET from CoAP-MQ broker Message Exchange

5.2. Server Endpoint Functions

5.2.1. CoAP-MQ broker subscribes to endpoint using GET+Observe, endpoint publishes notifications to the CoAP-MQ broker

The server mode endpoint enables the core.mq server to act as a client and subscribe to a resource on the endpoint using GET + Observe. Figure 7 shows the flow of core.mq server subscribing to the endpoint.

```
   EP                                               MQ        RD
    |                                               |         |
    |                                               |         |
    |                SUBSCRIBE                      |         |
    | <------- GET /0/... Observe: Token:XX ------- |         |
    |                                               |         |
    |                 PUBLISH                       |         |
    | ---------- 2.05 Content Observe:10----------> |         |
    |                                               |         |
    |                 PUBLISH                       |         |
    | ---------- 2.05 Content Observe:12----------> |         |
    |                                               |         |
    |                 PUBLISH                       |         |
    | ---------- 2.05 Content Observe:15----------> |         |
    |                                               |         |
    |                                               |         |
```

Figure 7: GET+Observe Message Exchange

5.2.2. CoAP-MQ broker publishes to endpoint

CoAP-MQ broker MAY update server mode endpoint using PUT when state updates are published on the associated topic. Endpoint server MAY update actuator or resource.

```
   EP                                               MQ        RD
    |                                               |         |
    |                                               |         |
    |                    PUBLISH                    |         |
    | <---------------- PUT /0/... ---------------- |         |
    |                                               |         |
    |                                               |         |
    | ---------------- 2.04 Changed---------------> |         |
    |                                               |         |
    |                                               |         |
```

Figure 8: PUT by CoAP-MQ broker Message Exchange

5.2.3. core.mq GET from endpoint

CoAP-MQ broker server MAY issue GET without Observe as needed to obtain state update from the server mode endpoint.

```
   EP                                               MQ        RD
    |                                               |         |
    |                                               |         |
    |                                               |         |
    | <--------------- GET /0/... ----------------- |         |
    |                                               |         |
    |                                               |         |
    | ---------------- 2.05 Content --------------> |         |
    |                                               |         |
    |                                               |         |
```

Figure 9: CoAP-MQ broker GET from endpoint Message Exchange

6. Enabling multiple publishers

6.1. Creating a topic

After registration of the EP in the RD and discovering the MQ function, a designated EP acting as publisher for a particular topic is responsible for creating such topic. To do so, it will have to register the new topic in the RD and create it on the MQ function by doing a first publication as shown in Figure 2.

After the topic has been created in the MQ, the MQ will be responsible of hosting this resource and to queue messages published on it as explained in 5.X

6.2. Publishing a topic from multiple publishers

After the topic has been registered in the RD and is created in the MQ, any device with the right access permissions can publish on that topic by using the topic field. For example in the following diagram, both EP1 and EP2 update the same topic that EP3 has previously subscribed to.

After the topic has been created in the MQ, the MQ will be responsible of hosting this resource and to queue messages published on it as explained in 5.X

```
EP1     EP2                                     MQ
|       |            PUBLISH                    |
| ---------------- PUT /0/TOPIC1 ------------> |
|       |                                       |
| <--------------- 2.04 Changed---------------- |
|       |                                       |
|       |            PUBLISH                    |
|       | --------- PUT /0/TOPIC1 ------------> |
|       |                                       |
|       | <------- 2.04 Changed---------------- |
|       |                                       |
```

Figure 10: CoAP-MQ broker GET from endpoint Message Exchange

6.3. Subscribing to a topic with multiple publishers

Subscription to this topic is the same as in 5.X, since it acts as any other resource. Following the previous example, if EP3 is subscribed to the shared topic, it should receive two updates from both EP1 and EP2.

```
   EP3                                              MQ
   |                 SUBSCRIBE                       |
   | ------- GET /0/TOPIC1 Observe --------------> |
   |                                                 |
   |                  PUBLISH                        |
   | <-----------  2.05 Content EP1 -------------- |
   |                                                 |
   |                  PUBLISH                        |
   | <-----------  2.05 Content EP2 -------------- |
   |                                                 |
```

Figure 11: CoAP-MQ broker GET from endpoint Message Exchange

7. Sleep-Wakeup Cycle

7.1. Registration with lifetime > max sleep interval

TBD

7.2. Signal using RD update to indicate wakeup at least once before lifetime

TBD

7.3. ACK not received timeout unregister - CoRE Observe unregister method? timeout linked to max sleep, lifetime? 7.31 with Token?

TBD

7.4. RD update wakeup triggers pending endpoint ops (GET, SUB, PUT), queued ops from core.mq server - can this handshake with update ACK?

TBD

7.5. Simple publish without RD update, either PUT or Observe Notification. Should this also reset the lifetime timer?

TBD

7.6. Example

```
   EP                                               MQ        RD
    |                                               |         |
    |                                               |         |
    |                   MQ DISCOVERY                |         |
    | -------- GET /.well-known/core?rt=core.mq --- | ------> |
    |                                               |         |
    | <-------- 2.05 Content "</mq>; rt=core.mq"--- | ------- |
    |                                               |         |
    |                TOPIC REGISTRATION             |         |
    | ----POST /rd "</mq/0/xx>;rt=core.pubsub" ---- | ------> |
    |                                               |         |
    | <-------- 2.01 Created Location: /rd/1234 --- | ------- |
    |                                               |         |
    |                                               |         |
    |                                               |         |
    |                 WAKEUP SIGNAL                 |         |
    |              UPDATE REGISTRATION              |         |
    | ---------------- PUT /rd/1234 --------------- | ------> |
    |                                               |         |
    |                    PUBLISH                    |         |
    | <---------------- PUT /0/... ---------------- |         |
    |                                               |         |
    | ---------------- 2.04 Changed---------------> |         |
    |                                               |         |
    |                   SUBSCRIBE                   |         |
    | <------- GET /0/... Observe: Token:XX ------- |         |
    |                                               |         |
    |                   PUBLISH                     |         |
    | ---- 2.05 Content Observe:10 Token:XX-------> |         |
    |                                               |         |
    |                  OK TO SLEEP                  |         |
    | <--------------- 2.04 Changed---------------- |         |
    |                                               |         |
    |                                               |         |
    |                   PUBLISH                     |         |
    | ---- 2.05 Content Observe:10 Token:XX-------> |         |
    |                                               |         |
    |                                               |         |
    |                    PUBLISH                    |         |
    | ----------------- PUT /0/... ---------------> |         |
    |                                               |         |
    | <--------------- 2.04 Changed---------------- |         |
    |                                               |         |
    |                                               |         |
```

Figure 12: Example Sleep-Wakeup Message Exchange

8. Security Considerations

TBD

9. IANA Considerations

TBD

10. Acknowledgements

Add your name here.

11. References

11.1. Normative References

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

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

[3] Gregorio, J., Fielding, R., Hadley, M., Nottingham, M., and D. Orchard, "URI Template", RFC 6570, March 2012.

[4] Shelby, Z., Bormann, C., and S. Krco, "CoRE Resource Directory", draft-ietf-core-resource-directory-01 (work in progress), December 2013.

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

11.2. Informative References

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

Authors’ Addresses

Michael Koster
ARM Limited

Email: Michael.Koster@arm.com

Ari Keranen
Ericsson

Email: ari.keranen@ericsson.com