INTERNET-DRAFT
Intended Status: Standards Track
Expires: September 22, 2016

Luyuan Fang
Microsoft

March 21, 2016

Failure Detection Extensions for Publish-Subscribe in CoAP
draft-fang-core-coap-pubsub-failure-detection-00

Abstract

This document defines extensions to the Constrained Application Protocol Publish/Subscribe function set, to make the protocol suitable to address the use case of failure detection in a hyper-scale system with millions of endpoints. Specifically, this document defines a Last Will mechanism and a scheme to guarantee hot fail-over of the pub/sub broker.

Status of this Memo

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

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

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."

The list of current Internet-Drafts can be accessed at http://www.ietf.org/1id-abstracts.html

The list of Internet-Draft Shadow Directories can be accessed at http://www.ietf.org/shadow.html

Copyright Notice

Table of Contents

1. Introduction

Many new protocols are being specified, and many existing ones are evolving, to meet the scalability, functionality, and footprint requirements of the Internet of Things (IoT), or Web of Things (WoT).

These protocols include Constrained Application Protocol (CoAP) [RFC7252], the Message Queuing Telemetry Transport (MQTT) protocol [MQTT], the Advanced Message Queuing Protocol (AMQP) [AMQP], and the Streaming Text Oriented Messaging Protocol (STOMP) [STOMP], among others. The Extensible Messaging and Presence Protocol (XMPP) [RFC7622] and HTTP/2 [RFC7540] also provide many capabilities that make them very suitable to support IoT use cases.

Although the proliferation of protocols for use in IoT is a clear indication that there is no single "silver bullet" protocol that can optimally address all the emerging IoT use cases, all these protocols are generally designed to provide connectivity to massive numbers of rather simple devices, typically resource constrained in terms of computing power, battery life, bandwidth, and reachability. As such, the design emphasis in these protocols is on scalability, small footprint, efficient use of available bandwidth, ease of parsing and processing, and client independency. To achieve these design objectives, these protocols have introduced several interesting and useful concepts to remove limitations in existing protocol and provide effective solutions to the new requirements.

As these protocols continue to mature, extensions are specified to increase the scope of their use (and, arguably, perhaps have one protocol prevail over others in a sort of "war of protocols" that is ensuing). In these extensions, it is often the case that a protocol is augmented with some desired characteristics or concepts already demonstrated by other protocols. For example, MQTT for Sensor Networks (MQTT-SN) [MQTTSN] is a flavor of MQTT that substitutes TCP with UDP, to achieve better scalability and lower complexity in certain use cases. Directly relevant to this document, recent work in IETF [I-D.draft-koster-core-coap-pubsub] is meant to add the desirable Publish/Subscribe (pub/sub) message paradigm, which distinguishes MQTT, AMQP, and other protocols, to CoAP.

Because of their desirable characteristics, the usefulness of these protocols is not necessarily confined to IoT use cases, but these protocol become strong candidates to address any use case where scalability, simplicity, and responsiveness are paramount. One of such use cases is fault detection in a hyper-scale network with millions or tens of millions of endpoints, such as a Data Center (DC). In a DC, many fault detection, diagnostics, and fault recovery mechanisms are typically deployed. However, as the scale and

complexity of the DC increases, there is an emerging need to devise new light-weight, scalable, device-agnostic, massively distributable, reactive mechanisms to assist and complement existing ones.

The simplicity, efficiency, and scalability of CoAP makes it a frontrunner as an interesting solution for the fault-detection use case. The addition of the pub/sub paradigm and the corresponding introduction of a pub/sub broker for CoAP [I-D.draft-koster-core-coap-pubsub] further provide a convenient, scalable architecture for fault detection, where the broker can rapidly detect the occurrence of faults in the connected clients (e.g., nodes in the DC) and propagate the information to interested listener in a timely fashion.

CoAP with pub/sub mechanism is an important ingredient for solving the use case, but of course it is not the only one. This document further extends the CoAP pub/sub function set with two additional, useful mechanisms for this purpose, thus making CoAP an even stronger candidate as a lightweight protocol solution for fault detection.

First, this document specifies a Last Will and Testament (LWT) mechanism to be added to the CoAP pub/sub function set. The LWT mechanism, which is used in other protocols such as MQTT, is explicitly designed to define the behavior of the broker in case of unexpected loss of connectivity with a client, as it is indeed the case when a fault occurs. The LWT mechanism is most effective when it is used in conjunction with some sort of Keep Alive mechanism, which should also be defined as part of the specification.

A well-known shortcoming of the pub/sub paradigm is the fact that the broker becomes a single point of failure. Clearly, this problem is extremely relevant in the use case at hand, where the broker is a key component of the fault detection architecture. This document further defines extensions to support redundancy among brokers, and achieve hot fail-over in case of failure of the brokers themselves.

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 RFC 2119 [RFC2119].

This document uses terms and concepts that are discussed in [RFC5988], [RFC6690], [RFC7252] and [I-D.ietf-core-resource-directory]. The URI template format [RFC6570] is also used in this specification.

This specification makes use of the following additional terminology,

defined in [I-D.draft-koster-core-coap-pubsub]:

- Publish-Subscribe (pub/sub): A messaging paradigm where a publisher publishes messages to a broker and interested receivers subscribe to the broker to receive messages. The published messages are delivered by the broker to the subscribed receivers.

- CoAP pub/sub function set: A group of REST resources that together provide the CoAP pub/sub service.

- CoAP pub/sub Broker: A server node capable of receiving messages from publishers and sending messages to subscribed receivers.

- CoAP pub/sub Client: A CoAP client that implements the CoAP pub/sub function set.

- Topic: A unique identifier for a particular item being published and/or subscribed to. The broker uses the topics to match subscriptions with publications.

- CoAP pub/sub Function Set: The interface between a CoAP pub/sub Broker and pub/sub Clients.

In addition, this document uses the following terms.

Term	Definition
AMQP	Advanced Message Queuing Protocol
CoAP	Constrained Application Protocol
CSP	Cloud Service Provider
DC	Data Center
IoT	Internet of Things
LWT	Last Will and Testament
MQTT	Message Queuing Telemetry Transport
MQTT-SN	MQTT for Sensor Networks
SDN	Software Defined Network
STOMP	Constrained Application Protocol
SVR	Server
WoT	Web of Things
XMPP	Extensible Messaging and Presence Protocol

2. Pub/Sub Broker for CoAP

The Pub/Sub Broker architecture and pub/sub function set is defined in [I-D.draft-koster-core-coap-pubsub].

The CoAP pub/sub Broker is a CoAP Server that exposes an interface

for CoAP clients to perform publish/subscribe interactions. The Broker typically has resource to buffer messages that are published by the CoAP clients. The Broker matches a published resource/message with the interested listener using Topics. Listeners subscribe to specific topics to receive information published by specific clients.

The CoAP pub/sub function set as defined in [I-D.draft-koster-core-coap-pubsub] provides the following operations.

- o DISCOVER. Used by CoAP clients to discover CoAP pub/sub Brokers
- o CREATE. Used by CoAP clients to create a topic.
- o PUBLISH. Used by CoAP clients to update a specific topic on the broker (i.e., publish a message on a topic).
- o SUBSCRIBE. Used by CoAP clients (listeners) to subscribe to topics.
- o UNSUBSCRIBE. Used by CoAP clients (listeners) to unsubscribe to topics.
- o READ. Used by a CoAP client (listener) to obtain the most recent published value on a topic. Useful when a client first joins or re-joins the pub/sub system.
- o REMOVE. Used by a CoAP client to remove an existing topic.

3. Last Will and Testament

The Last Will and Testament (LWT) mechanism is used to define the behavior of the broker in case of unexpected loss of connectivity with a client. This may be the result of an error detected by the broker, or may be triggered by the client failing to communicate with the broker within a Keep Alive.

In order to implement the LWT mechanism, the CoAP pub/sub function set needs to be extended by adding:

i. a mechanism to create a WILL topic;

ii. a mechanism to specify a WILL message.

The WILL message is the message that the broker MUST post on the WILL topic in case a failure of the corresponding CoAP client is detected.

These two mechanisms are implemented by modifying the CREATE

operation in the CoAP pub/sub function set.

4. Pub/Sub Broker Fail-Over

TBD.

5. Security Considerations

TBD.

6. IANA Considerations

TBD.

7. References

7.1 Normative References

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

[RFC6570] Gregorio, J., Fielding, R., Hadley, M., Nottingham, M., and D. Orchard, "URI Template", RFC 6570, DOI 10.17487/RFC6570, March 2012, <http://www.rfc-editor.org/info/rfc6570>.

[RFC6690] Shelby, Z., "Constrained RESTful Environments (CoRE) Link Format", RFC 6690, DOI 10.17487/RFC6690, August 2012, <http://www.rfc-editor.org/info/rfc6690>.

[RFC7252] Shelby, Z., Hartke, K., and C. Bormann, "The Constrained Application Protocol (CoAP)", RFC 7252, DOI 10.17487/RFC7252, June 2014, <http://www.rfc-editor.org/info/rfc7252>.

[RFC7622] P. Saint-Andre et al., "Extensible Messaging and Presence Protocol (XMPP): Address Format", RFC 6122, September 2015, <https://tools.ietf.org/html/rfc7622>.

[RFC7540] M. Belshe et al., "Hypertext Transfer Protocol Version 2 (HTTP/2)", RFC 7540, May 2015, <https://tools.ietf.org/html/rfc7540>.

7.2 Informative References

[RFC5988] Nottingham, M., "Web Linking", RFC 5988, DOI 10.17487/RFC5988, October 2010, <http://www.rfc-

editor.org/info/rfc5988>.

[I-D.ietf-core-resource-directory] Shelby, Z., Koster, M., Bormann, C., and P. Stok, "CoRE Resource Directory", draft-ietf-core-resource-directory-05 (work in progress), October 2015.

[I-D.draft-koster-core-coap-pubsub] M. Koster et al., "Publish-Subscribe Broker for the Constrained Application Protocol (CoAP)",draft-koster-core-coap-pubsub-04 (work in progress), November 2015.

[AMQP] "OASIS Advanced Message Queuing Protocol (AMQP) Version 1.0", OASIS Standard, October 2012, <http://docs.oasis-open.org/amqp/core/v1.0/os/amqp-core-overview-v1.0-os.html>.

[MQTT] "MQTT Version 3.1.1", OASIS Standard, October 2014, <http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/os/mqtt-v3.1.1-os.html>.

[MQTTSN] "MQTT For Sensor Networks (MQTT-SN) Protocol Specification Version 1.2", November 2013, <http://mqtt.org/new/wp-content/uploads/2009/06/MQTT-SN_spec_v1.2.pdf>.

[STOMP] "STOMP Protocol Specification, Version 1.2", <https://stomp.github.io/stomp-specification-1.2.html>.

Authors' Addresses

Luyuan Fang
Microsoft
15590 NE 31st St
Redmond, WA 98052
Email: lufang@microsoft.com