WoT- websub

F. Viola[1]

[1]ARCES
University of Bologna

May. 2017

Outline

Outline

Disclaimer

This presentation tries to summarize the main concepts contained in the **WebSub W3C Editor's Draft**. It is just a set of key concepts for a productive discussion in the Web of Things working group of the ARCES laboratory.

The referred document is a **Candidate Recommendation**, is under review (that will end on May, 11th).

Introduction

WebSub provides a common mechanism for communication between publishers of any kind of Web content and their subscribers, based on HTTP web hooks. Subscription requests are relayed through hubs, which validate and verify the request. Hubs then distribute new and updated content to subscribers when it becomes available.

WebSub was previously known as **PubSubHubbub** 👎.

Concepts

Topic An HTTP resource URL. It is the unit to which one can subscribe to changes.

Concepts

Topic An HTTP resource URL. It is the unit to which one can subscribe to changes.

Hub The server which implements both sides of this protocol.

Concepts

Topic An HTTP resource URL. It is the unit to which one can subscribe to changes.

Hub The server which implements both sides of this protocol.

Publisher the owner of a topic. Notifies the hub when the topic feed is updated. The publisher is unaware of the subscribers, if any.

Concepts

Topic An HTTP resource URL. It is the unit to which one can subscribe to changes.

Hub The server which implements both sides of this protocol.

Publisher the owner of a topic. Notifies the hub when the topic feed is updated. The publisher is unaware of the subscribers, if any.

Subscriber An entity that wants to be notified of changes on a topic. The subscriber must be directly network-accessible and is identified by its Subscriber Callback URL.

Concepts

Topic An HTTP resource URL. It is the unit to which one can subscribe to changes.

Hub The server which implements both sides of this protocol.

Publisher the owner of a topic. Notifies the hub when the topic feed is updated. The publisher is unaware of the subscribers, if any.

Subscriber An entity that wants to be notified of changes on a topic. The subscriber must be directly network-accessible and is identified by its Subscriber Callback URL.

Subscription It is described by the tuple (Topic URL, Subscriber Callback URL).

Concepts

Topic	An HTTP resource URL. It is the unit to which one can subscribe to changes.
Hub	The server which implements both sides of this protocol.
Publisher	the owner of a topic. Notifies the hub when the topic feed is updated. The publisher is unaware of the subscribers, if any.
Subscriber	An entity that wants to be notified of changes on a topic. The subscriber must be directly network-accessible and is identified by its Subscriber Callback URL.
Subscription	It is described by the tuple (Topic URL, Subscriber Callback URL).
Callback URL	The URL at which a subscriber wishes to receive notifications.

Concepts

Topic	An HTTP resource URL. It is the unit to which one can subscribe to changes.
Hub	The server which implements both sides of this protocol.
Publisher	the owner of a topic. Notifies the hub when the topic feed is updated. The publisher is unaware of the subscribers, if any.
Subscriber	An entity that wants to be notified of changes on a topic. The subscriber must be directly network-accessible and is identified by its Subscriber Callback URL.
Subscription	It is described by the tuple (Topic URL, Subscriber Callback URL).
Callback URL	The URL at which a subscriber wishes to receive notifications.
Event	An event that causes updates to one or more topics. For each event that happens, multiple topics could be affected.

Concepts

Topic	An HTTP resource URL. It is the unit to which one can subscribe to changes.
Hub	The server which implements both sides of this protocol.
Publisher	the owner of a topic. Notifies the hub when the topic feed is updated. The publisher is unaware of the subscribers, if any.
Subscriber	An entity that wants to be notified of changes on a topic. The subscriber must be directly network-accessible and is identified by its Subscriber Callback URL.
Subscription	It is described by the tuple (Topic URL, Subscriber Callback URL).
Callback URL	The URL at which a subscriber wishes to receive notifications.
Event	An event that causes updates to one or more topics. For each event that happens, multiple topics could be affected.
Notification	A payload describing how a topic's contents have changed, or the full updated content. Depending on the topic's content type, the difference (or "delta") may be computed by the hub and sent to all subscribers.

Protocol Flow

As you can imagine, the publish subscribe mechanism works as follows:

- Publishers register a topic URL to a hub

Protocol Flow

As you can imagine, the publish subscribe mechanism works as follows:

- Publishers register a topic URL to a hub
- Publishers notify their hub(s) URLs when their topic(s) change

Protocol Flow

As you can imagine, the publish subscribe mechanism works as follows:

- Publishers register a topic URL to a hub
- Publishers notify their hub(s) URLs when their topic(s) change
- Subscribers POST to one or more of the advertised hubs for a topic they're interested in

Protocol Flow

As you can imagine, the publish subscribe mechanism works as follows:

- Publishers register a topic URL to a hub
- Publishers notify their hub(s) URLs when their topic(s) change
- Subscribers POST to one or more of the advertised hubs for a topic they're interested in
- When the hub identifies a change in the topic, it sends a notification to all registered subscribers.

Outline

The Publisher

The publisher must inform the hubs it previously designated when a topic has been updated. The hub and the publisher can agree on any mechanism, as long as the hub is eventually able send the updated payload to the subscribers.

If the publisher wishes to migrate existing subscriptions to a new topic URL, it can do so using **HTTP redirects**. This is important because allows a seamless transition for any clients **polling** the topic URL.

Outline

The Subscriber

Basically, a subscriber must satisfy requirements about the protocols used to perform the **discovery** and must be able to send a subscription containing the following fields:

hub.callback The subscriber's callback URL where notifications should be delivered. It is uniquer per subscription!

hub.mode The string `subscribe` or `unsubscribe`

hub.topic The topic to subscribe to.

Optional fields are **hub.secret** and **hub.lease_seconds**.
The subscriber must also be able to send a 2xx response to every content notification.

Note: the periodic reconfirm that a subscription is still active is optional!

Outline

The Hub

A hub must be able to accept subscription (and unsubscription) requests conforming to the previous rules. Currently the requested lease duration (optionally requested by a subscriber) may not be respected.

A hub must also allow subscribers to re-request already active subscriptions.

When sending a notification, a hub may send only the delta, or the whole content at the topic url.

If security is requested, the hub must sign every content notification.

Outline

Notifications

A **content distribution** request is sent from the Hub to the Subscriber when new content is available for a topic URL. The request is an **HTTP POST** from the hub to the subscriber's callback URL. The HTTP body of the POST request must include the payload of the notification. The content distribution request must have a Content-Type Header corresponding to the Content-Type of the topic, and must contain the full contents of the topic URL, with an exception allowed as described below.

The request MUST include at least one Link Header with `rel=hub` pointing to a Hub associated with the topic being updated. It must also include one Link Header with `rel=self` set to the canonical URL of the topic being updated.

Each notification must be acknowledged by the subscribers. Hubs should retry notifications up to self-imposed limits on the number of times and the overall time period to retry. When the failing delivery exceeds the hub's limits, the hub terminates the subscription.

Outline

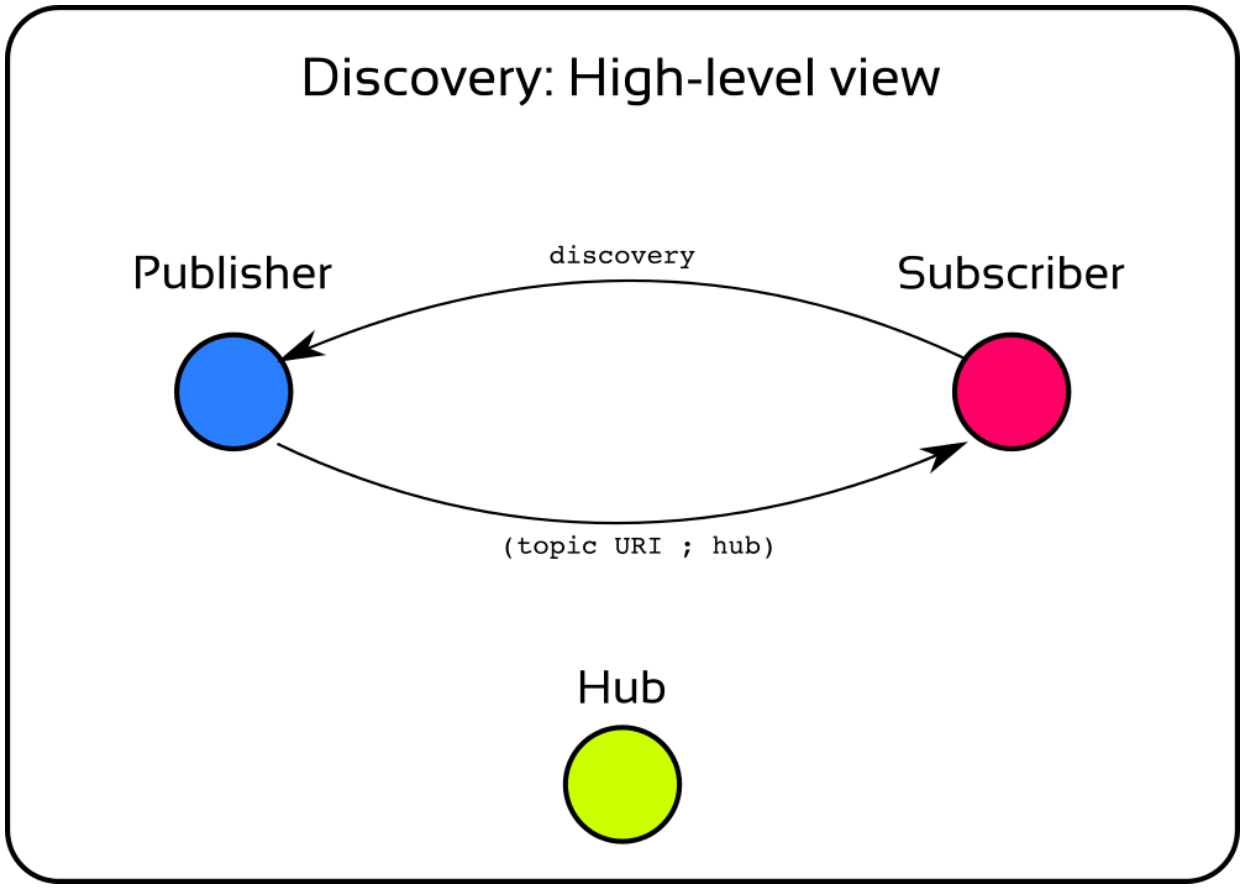
Discovery

The discovery mechanism aims at identifying at least 2 URLs:

- The URL of the hub(s) designated by the publisher.
- The canonical URL for the topic to which subscribers are expected to use for subscriptions.

So. . . the entity that wants to subscribe to a topic should query the publisher about the url of the topic and the hub that will send notifications.

Discovery

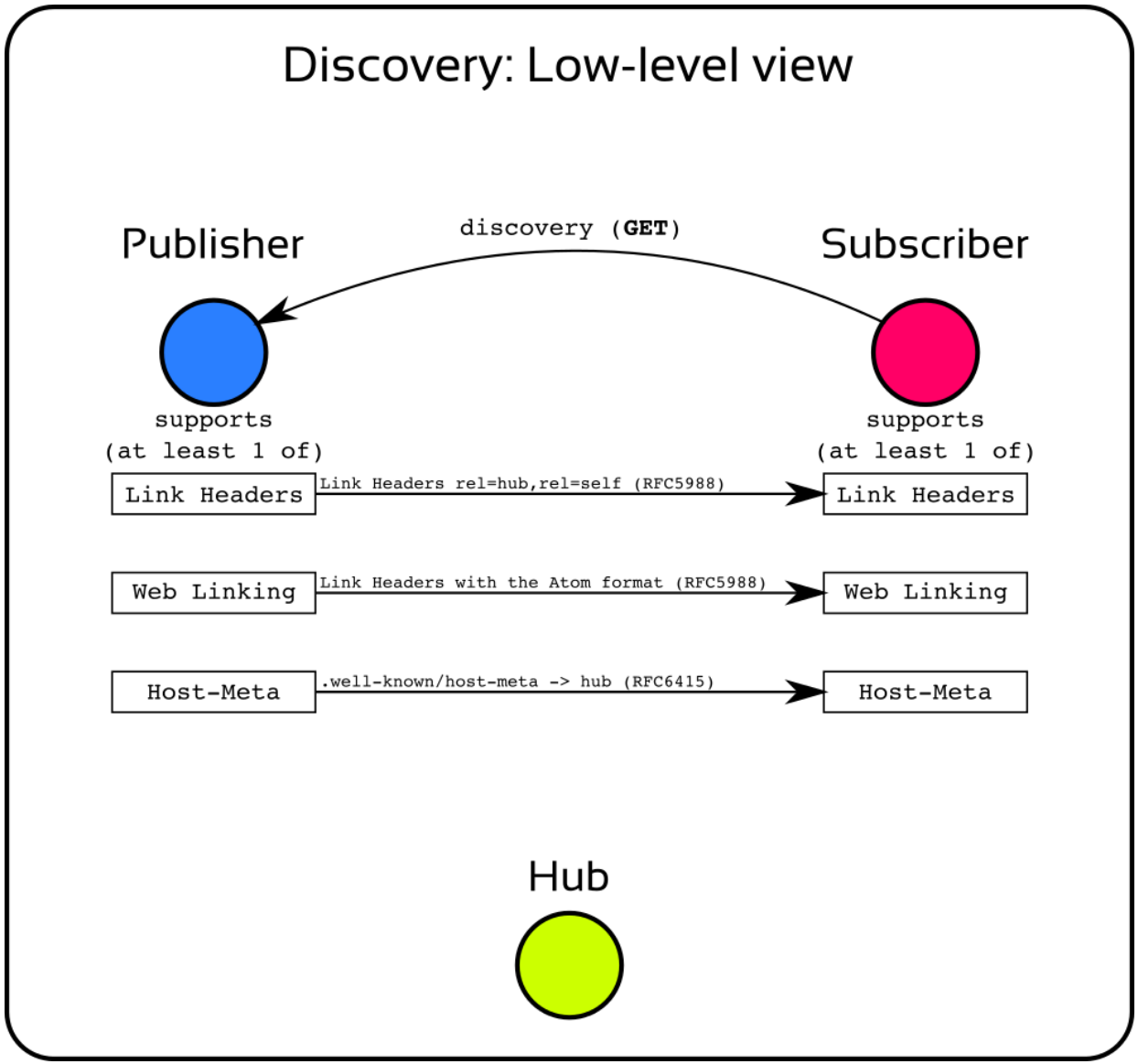

Discovery: High-level view
Publisher
discovery
Subscriber
(topic URI ; hub)
Hub

Discovery

Currently three discovery mechanisms are implemented:

- **Link Headers**: the RFC5988 defines a field for serializing link in HTTP headers.
- **Web Linking**: same as below, but referring to the ATOM format
- **Host-Meta**: (currently at risk, may be deprecated).

Discovery


Discovery: Low-level view
Publisher
discovery (GET)
Subscriber
supports
(at least 1 of)
supports
(at least 1 of)
Link Headers
Link Headers rel=hub,rel=self (RFC5988)
Link Headers
Web Linking
Link Headers with the Atom format (RFC5988)
Web Linking
Host-Meta
.well-known/host-meta -> hub (RFC6415)
Host-Meta
Hub

Discovery

```
GET /feed HTTP/1.1
Host: example.com

HTTP/1.1 200 Ok
Content-type: text/html
Link: <https://hub.example.com/>; rel="hub"
Link: <http://example.com/feed>; rel="self"

<!doctype html>
<html>
  <head>
    <link rel="hub" href="https://hub.example.com/">
    <link rel="self" href="http://example.com/feed">
  </head>
  <body>
    ...
  </body>
</html>
```

Outline

WebSub and Security

Before confirming a subscription...

In order to prevent an attacker from creating unwanted subscriptions on behalf of a subscriber (or unsubscribing desired ones), a hub must ensure that the subscriber did indeed send the subscription request.

The hub verifies a subscription request by sending an HTTP GET request to the subscriber's callback URL as given in the subscription request. This request has the following query string arguments appended:

hub.mode	The literal string "subscribe" or "unsubscribe"
hub.topic	The topic URL given in the corresponding subscription request.
hub.challenge	A hub-generated, random string that MUST be echoed by the subscriber to verify the subscription.
hub.lease_seconds	Must be present for subscription requests, may be present for unsubscriptions and if present must be ignored by unsubscribers.

WebSub and Security

About subscriptions. . .

- subscribers should always favor the HTTPS URL for hubs (even if the URL is advertised as HTTP)
- the subscribers should use unique unguessable capability URLs for the callbacks, as well as make them available via HTTPS
- subscribers should use a hub.secret when subscribing to allow signature of the content distribution
- Hubs should enforce short lived `hub.lease_seconds` (10 days is a good default)

Outline

WebSub vs SEPA - Differences

- WebSub states that a subscription may be referred to any kind of web content. SEPA allows to subscribe to graph patterns.

WebSub vs SEPA - Differences

- WebSub states that a subscription may be referred to any kind of web content. SEPA allows to subscribe to graph patterns.
- WebSub's subscriptions are topic-based, ours are graph-based.

WebSub vs SEPA - Differences

- WebSub states that a subscription may be referred to any kind of web content. SEPA allows to subscribe to graph patterns.
- WebSub's subscriptions are topic-based, ours are graph-based.
- WebSub's Discovery mechanism envisions a direct communication between publisher and subscriber. SEPA allows *Things* to discover other *Things* through the context broker.

WebSub vs SEPA - Differences

- WebSub states that a subscription may be referred to any kind of web content. SEPA allows to subscribe to graph patterns.
- WebSub's subscriptions are topic-based, ours are graph-based.
- WebSub's Discovery mechanism envisions a direct communication between publisher and subscriber. SEPA allows *Things* to discover other *Things* through the context broker.
- With WebSub every subscriber must be provided with a publicly visible URI. SEPA allows to subscribe and receive notification even from a local network. (HTTP vs WebSocket)

WebSub vs SEPA - Differences

- WebSub states that a subscription may be referred to any kind of web content. SEPA allows to subscribe to graph patterns.
- WebSub's subscriptions are topic-based, ours are graph-based.
- WebSub's Discovery mechanism envisions a direct communication between publisher and subscriber. SEPA allows *Things* to discover other *Things* through the context broker.
- With WebSub every subscriber must be provided with a publicly visible URI. SEPA allows to subscribe and receive notification even from a local network. (HTTP vs WebSocket)
- A subscriber must have a callback URI for every subscription. SEPA allows to have a single communication channel for all the subscription of a client.

WebSub vs SEPA - Differences

- WebSub states that a subscription may be referred to any kind of web content. SEPA allows to subscribe to graph patterns.
- WebSub's subscriptions are topic-based, ours are graph-based.
- WebSub's Discovery mechanism envisions a direct communication between publisher and subscriber. SEPA allows *Things* to discover other *Things* through the context broker.
- With WebSub every subscriber must be provided with a publicly visible URI. SEPA allows to subscribe and receive notification even from a local network. (HTTP vs WebSocket)
- A subscriber must have a callback URI for every subscription. SEPA allows to have a single communication channel for all the subscription of a client.