Constrained RESTful Environments WG (core)

Chairs:

Cullen Jennings <fluffy@cisco.com>

Carsten Bormann <cabo@tzi.org>

Mailing List:

core@ietf.org

Jabber:

core@jabber.ietf.org

- **We assume people have read the drafts**
- **Meetings serve to advance difficult issues by making good use of face-to-face communications**
- **Be aware of the IPR principles, according to RFC 3979 and its updates**

✓Blue sheets
✓Scribe(s)

interim 2012-05-16: core WG Agenda (all times are UTC)

14:30	**Introduction, Agenda, Status**	**Chairs (10)**
14:40	**Tickets for -observe**	**KH (49)**
15:29	**Tickets for -coap**	**ZS (59)**
16:28	**Tickets for -block**	**CB (9)**
16:37	**Non-ticket discussion**	**all (38)**
17:15	**planning, wrap-up**	**Chairs (15)**
17:30	***retire for the day***	

-observe 14:40–15:29

- -- check defined resolution and go ahead (2 min)
- #225 Explain why it is not always possible to react to a RST that is in reply to a NON (editorial minor)
- -- discuss (40 min)
- #204 Introduce a minimal version of Pledge (protocol enhancement major)
- #217 how fast must the observe clock be able to go? (protocol enhancement major)
- #220 Should observe support time series data? (protocol enhancement minor)
- #227 Make aborting the previous transaction optional (protocol enhancement minor)
- -- tickets with a clear way forward (optional) (7 min)
- #219 Clarify that observe is about eventual consistency (editorial minor)
- #221 Occasionally sending CON is not just a security consideration (protocol defect minor)
- #223 Fix reordering detection condition description (editorial minor)
- #234 Editorial updates to -observe examples (editorial minor)
- #235 Avoid extending the base standard retransmission rules (other technical minor)
- #236 Clarify the semantics of the "obs" link target attribute (other technical minor)
- #237 Multicast -> reference groupcomm draft (editorial minor)
- -- tickets that need more work on the mailing list
- (none)

Observing Resources in CoAP

draft-ietf-core-observe

CoRE WG Virtual Interim | 2012/05/12 | Klaus Hartke

Ticket #225

Explain why it is not always possible to react to a RST that is in reply to a NON

Section 4.2 says: If the client rejects a non-confirmable notification with a RST message, the server MAY remove the client from the list of observers.

Cullen Jennings thinks this needs to be a MUST.

This is indeed intended to be MAY. We want to make the need to store state for a NON optional. A sender of a NON message may discard the MID state for that message whenever it wants. That may make acting on a RST to that MID impossible. Hence MAY.

Text proposal:

Implementation note: This "MAY" is a relaxation for constrained implementations. The expectation is, where a server still has the state available that is needed to map the RST to an observation relationship, it will indeed remove the client from the list of observers.

Check: Is this what we want to do?

Ticket #204

Introduce a minimal version of Pledge

Various proposals have been made to solve the robust observation relationships problem (#174). #174 was closed, but there is still work to do:

- #174 was closed because the "80 %" were solved and a solution for the "20 %" had not yet come up. We should review the text to make sure the way Max-Age is used now can be made into a default behavior for potential future options (e.g., Pledge).
- Cullen Jennings notes that using Max-Age to indicate when server will send next notification is just wrong. That's not what Max-Age means. We need separate control of how long data is fresh, and how often the client needs to refresh the subscription.

Next slides: Separate concepts for controlling how long data is fresh and determining how long a client is interested in a resource.

Resource State

Background

1. A resource has a state.
   The state can change over time
2. The representation of a resource state is an accurate description of the current state of the resource until the resource changes its state
3. When the state changes, the server sends a notification to each client interested in the resource
   (We cannot send more notifications than the network/client can handle though → eventual consistency)
4. Each notification contains a representation of the new resource state

Theory

5. The representation contained in a notification is fresh until the next notification arrives

Problem

6. A server may go away or erroneously come to the conclusion that a client is no longer interested in the resource

Solution: Soft-state

7. The representation will expire unless it is refreshed.

Implementation

8. Each notification contains an indication of when the server will send the next notification at latest
9. This enables the client to determine if the next notification should have arrived, but also requires the server to send a notification even when the resource state did not change
10. There's a trade-off between detecting failure sooner and sending less unneeded messages

Interest

Background

1. A client has an interest in a resource.
   The interest can change over time

2. A server sends notifications only
   to clients that are interested

Theory

3. When a client becomes interested or stops
   being interested in a resource,
   it sends a message to the server

Problem

4. A client may go away without saying that is no longer interested

Solution: Soft-state

5. The client's interest in a resource
   will expire unless it is refreshed.

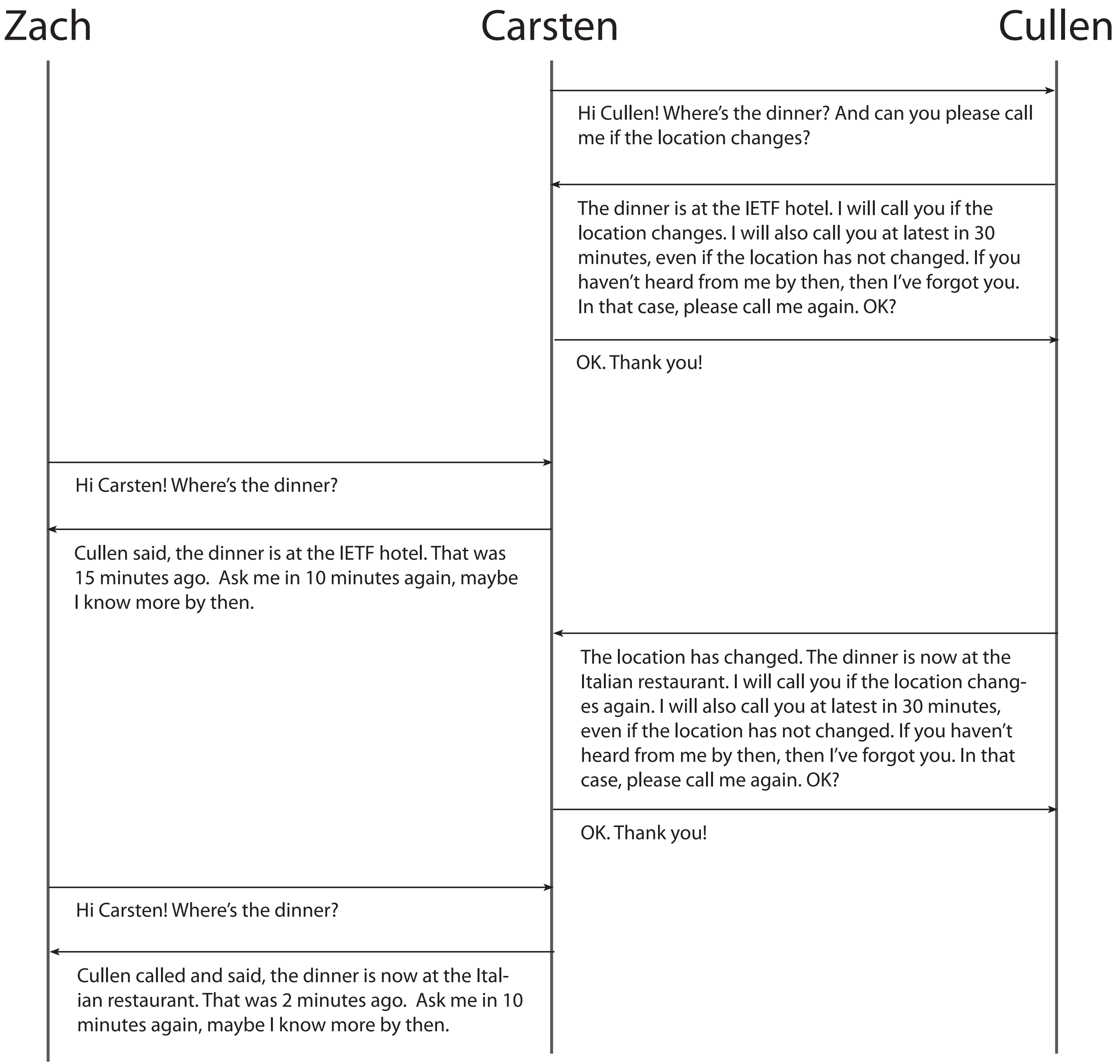
Implementation

6. A confirmable notification asks the client to confirm its interest in the resource

7. If the client confirms the notification, the client's interest in the resource is assumed until the next confirmable notification

8. This enables the server to determine if the client is still there, but also requires the client to send a message even when its interest in the resource did not change

9. There's a trade-off between detecting failure sooner and sending fewer unneeded messages



Zach

Carsten

Cullen

Hi Cullen! Where’s the dinner? And can you please call me if the location changes?

The dinner is at the IETF hotel. I will call you if the location changes. I will also call you at latest in 30 minutes, even if the location has not changed. If you haven’t heard from me by then, then I’ve forgot you. In that case, please call me again. OK?

OK. Thank you!

Hi Carsten! Where’s the dinner?

Cullen said, the dinner is at the IETF hotel. That was 15 minutes ago. Ask me in 10 minutes again, maybe I know more by then.

The location has changed. The dinner is now at the Italian restaurant. I will call you if the location changes again. I will also call you at latest in 30 minutes, even if the location has not changed. If you haven’t heard from me by then, then I’ve forgot you. In that case, please call me again. OK?

OK. Thank you!

Hi Carsten! Where’s the dinner?

Cullen called and said, the dinner is now at the Italian restaurant. That was 2 minutes ago. Ask me in 10 minutes again, maybe I know more by then.

Ticket #217

How fast must the observe clock be able to go?

Section 4.4 mandates that a sequence number must not be reused within 2^{16} seconds. Since there are 2^{16} possible values, this means that a client cannot be notified more than once per second on average.

Cullen Jennings notes that many applications may want way faster updates than this.

The current requirement is very conservative, reflecting a very simple implementation strategy. We could come up with alternative, more elaborate requirements that enable faster updates.

How fast is fast enough?

How much are we willing to assume about reordering and delivery probabilities/distributions?

Should we separate timestamp and sequence number?

Ticket #220

Should observe support time series data?

Observe currently is about eventual consistency.

Jeroen Hoebeke notes that it may be useful to enable a server to inform a client reliably about every state change of a resource.

What kinds of mechanisms would we need to add to support time series data?

Is the resulting set of changes a desirable addition?

Ticket #227

Make aborting the previous transaction optional

Section 4.5 requires a server implementation to stop an old transmission and carry the retransmit count over to the new transaction.

Cullen Jennings notes that this may be hard to implement in some cases and a minor optimization for an edge case.

He proposes that a server implementation can choose if it wants to abort the previous transaction or run two transactions in parallel.

- If it aborts the previous transaction, then it needs to copy over the retransmit state to the new transaction.
- If it doesn't cancel the old transaction, the device still finds out the device is gone.

Who has implemented this MUST? What was your experience?

If not, would this MUST be hard to implement in your structure?

Other Tickets

#219 Clarify that observe is about eventual consistency

#221 Occasionally sending CON is not just a security consideration

#223 Fix reordering detection condition description

#234 Editorial updates to -observe examples

#235 Avoid extending the base standard retransmission rules

#236 Clarify the semantics of the “obs” link target attribute

#237 Multicast — reference the groupcomm draft

-coap 15:29–16:28

- -- check defined resolution and go ahead (22 min)

- #202 Remove the 270 byte artificial limit (protocol defect minor)
- #213 Path/Query options minimum length (protocol defect minor)
- #214 Adopt vendor-defined option into core-coap (protocol enhancement minor)
- #218 Mostly obvious section 5.10.8 fixes (other technical minor)
- #222 RawPublicKey identifier (protocol enhancement minor)
- #228 Proxying of multicast requests (protocol enhancement minor)
- #229 Move sections 10-10.2. out of the "Security Considerations" (editorial minor)
- #232 Clarify inclusion of Location options in a 2.01 (Created) response (editorial minor)
- #233 Response codes with payload inconsistency (editorial trivial
- #239 Always reserve option delta 15 (other technical minor)

- -- discuss (30 min)

- #201 Clarify use of retransmission window for duplicate detection (editorial minor)
- #215 editorial issues around Congestion Control (editorial major)
- #230 Multiple Location options need to be processed as a unit (protocol defect minor)

- -- tickets with a clear way forward (optional) (5 min)

- #207 Add advice on default values for critical options (editorial minor)
- #212 Option numbers 14, 28, 42, ... reserved but usable (editorial minor)
- #224 Clarify the concept of end-point (editorial major)
- #216 IANA: get Multicast addresses (other technical major)
- #226 Clarify which language addresses intermediaries in general vs. forward proxies specifically (other technical major)

-block 16:28–16:37

- -- check defined resolution and go ahead (6)

- #203 Restrict the potential combinations of Block1 and Block2 (protocol defect major)
- #210 Disentangle Block and Token (protocol defect major)
- #211 Signal provisional responses (atomic Block1) in the response code (protocol defect major)

- -- discuss (0)

- -- tickets with a clear way forward (optional) (3)

- #206 Clarify that atomic Block1 transfers match per token *and* endpoint (editorial major)
- #205 Clarify that Size does not modify the request semantics beyond adding the size information (editorial minor)
- #209 Add potential attacks to security considerations (editorial minor)

- -- tickets that need more work on the mailing list

Discussion 16:37–17:15

- (link-format?)

Planning 17:15–17:30

- **Next interim?**