Constrained RESTful Environments WG (core)

Chairs:

Andrew McGregor <andrewmcgr@gmail.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**
- **Note Well: Be aware of the IPR principles, according to RFC 3979 and its updates**

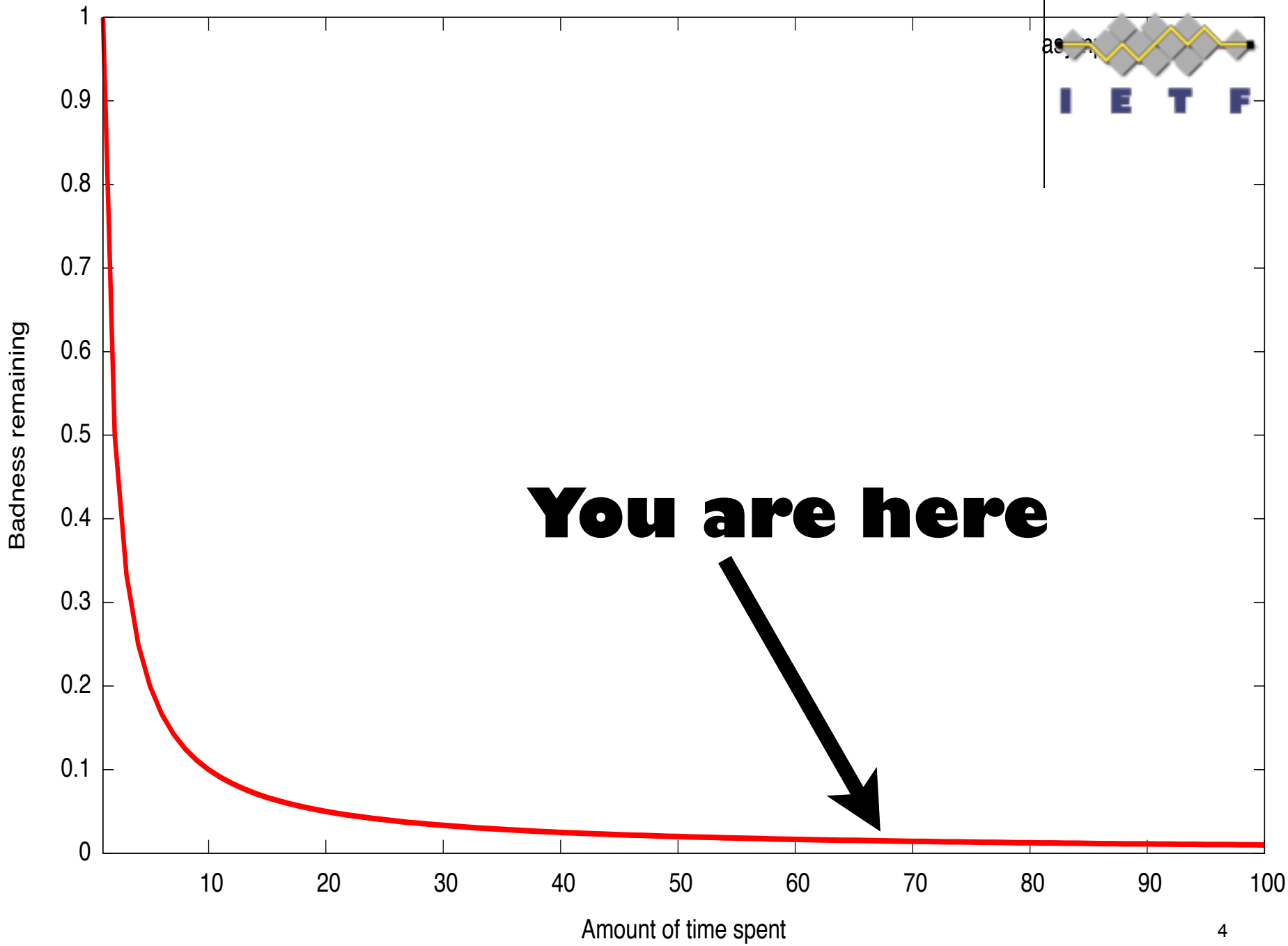
✓Blue sheets
✓Scribe(s):
http://tools.ietf.org/wg/core/minutes

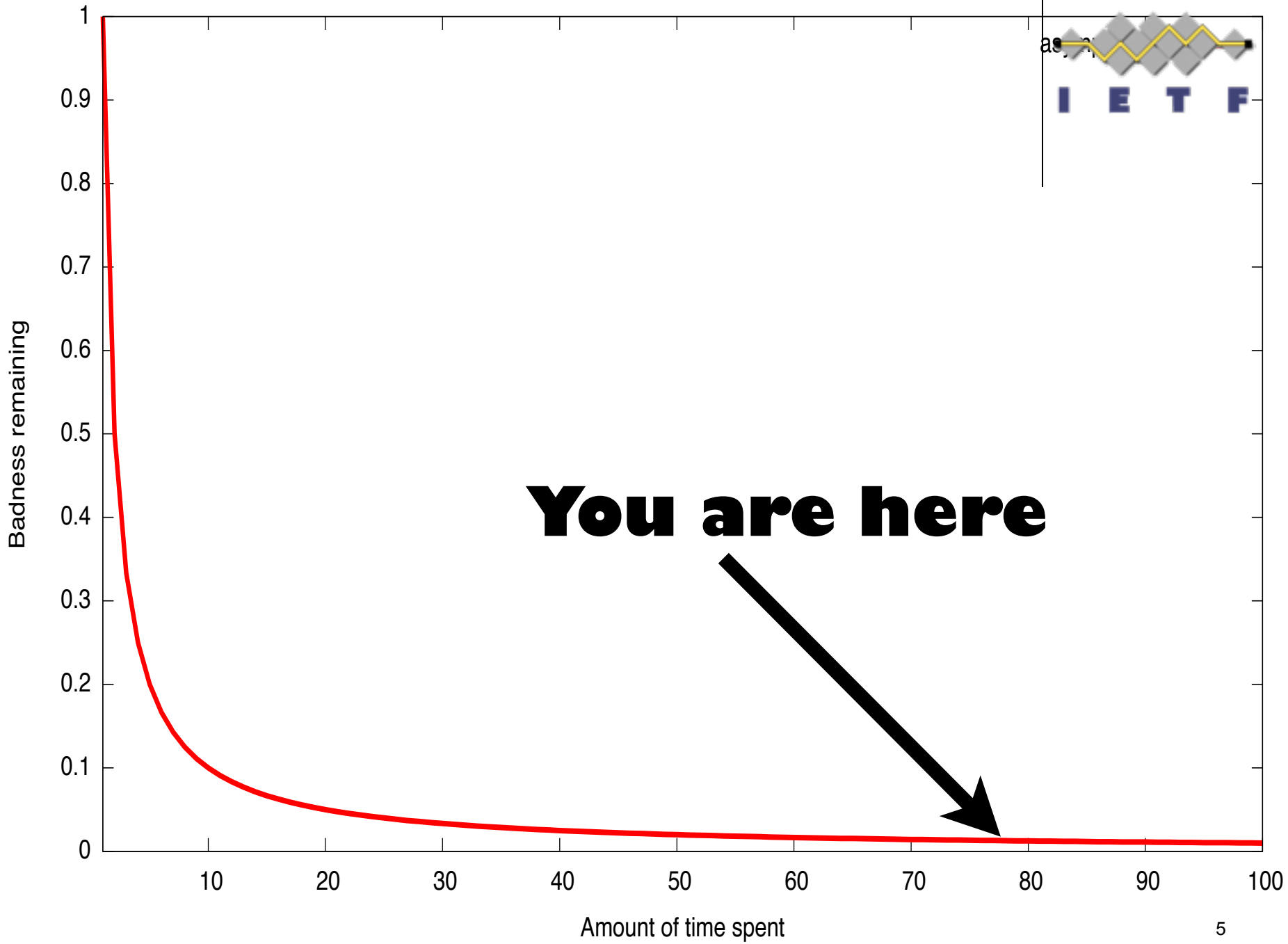
Milestones (from WG charter page)

http://datatracker.ietf.org/wg/core/charter/

Document submissions to IESG:

- **Dec 2012 CoAP protocol specification with mapping to HTTP Rest API to IESG**
- **Feb 2013 Blockwise transfers in CoAP to IESG**
- **Feb 2013 Observing Resources in CoAP to IESG**
- **Apr 2013 Group Communication for CoAP to IESG**
- **Dec 2099 HOLD (date TBD) Constrained security bootstrapping specification to IESG**


Badness remaining
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
10
20
30
40
50
60
70
80
90
100
Amount of time spent
You are here


Badness remaining
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
10
20
30
40
50
60
70
80
90
100
Amount of time spent
You are here

Group 1: 2nd WGLC

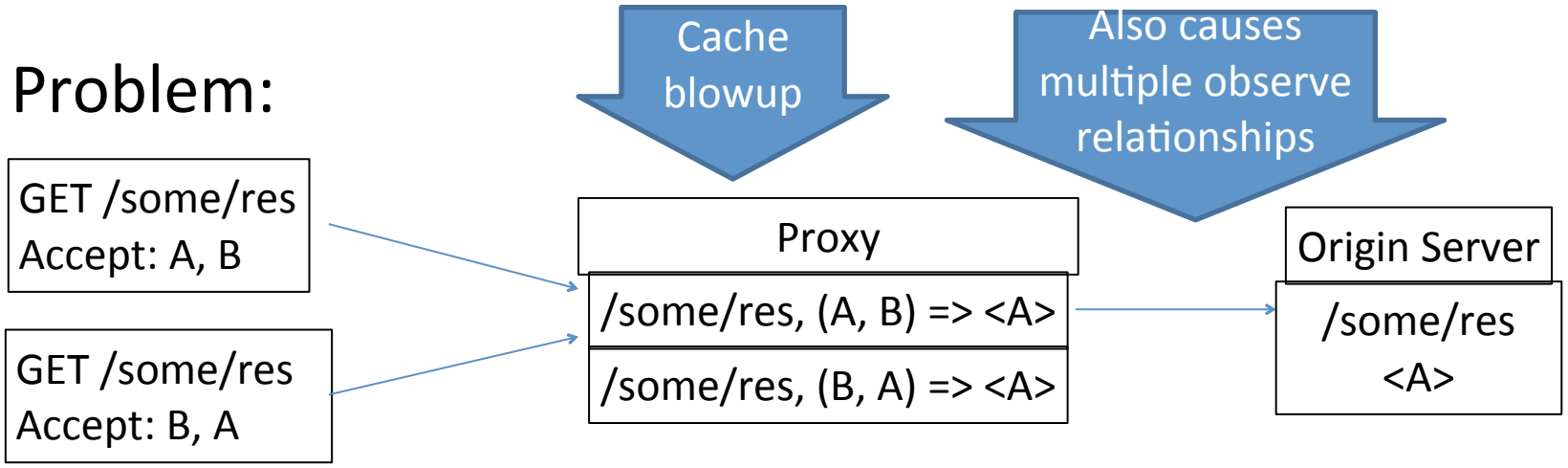
coap-13 → coap-14

draft-ietf-core-coap-14

- Will be published a couple of hours from now
  - please do read and comment again during IETF last call!
- Changes from –13:
  - Clarify that payload sniffing is acceptable only if no Content-Format was supplied.
  - Clarify that safe-to-forward options in a 2.03 Valid response update the cache.
  - Clarify URI examples (Appendix B).
  - Numerous editorial improvements and clarifications.
  - Made Accept option non-repeatable.

Repeatable Accept Option

- Problem:



- Solution A: “Just do not do it” and keep it general
- Solution B: Change to non-repeatable
  - + Predictable behavior when going through different caches
  - + Makes Accept and proxies easier to understand
  - + We have out-of-band content negotiation (ct=“<A> <B>”)
  - + Can be added later as extension, if really needed

Group 1a: WGLC processing

observe, block

draft-ietf-core-observe-08

- **Changes from –07 to –08:**
  - **Expanded text on transmitting notification while a previous transmission is pending (#242).**
  - **Changed reordering detection to use a fixed time span of 128 seconds instead of EXCHANGE_LIFETIME (#276).**
  - **Removed the use of the freshness model to determine if the client is still on the list of observers.**

draft-ietf-core-block-10

- **Still awaits grand editorial rewrite**
  - **right after coap-14 is shipped**

Tickets

- … are our way to make the steps forward
- are at:
- http:// **tools.ietf.org/wg/core**
- Updates are sent to the mailing list
  - Please review!
- When we close a ticket, please review once more!

Group 2: groupcomm

Group Communication for CoAP

Akbar Rahman
Esko Dijk

IETF 86, March 2013

http://www.ietf.org/id/draft-ietf-core-groupcomm-05.txt

Summary of Changes (1/4)

- I-D had two updates (Rev. 04 & 05) after IETF-85 (Atlanta). Many of the updates were due to the detailed review by Peter van der Stok.
  - Thank you, Peter, for your valuable comments!

- Changes from ietf-03 to ietf-04:
  - Section 2.3 (Potential Solutions for Group Communications) ; moved to draft-dijk-core-groupcomm-misc (#266).
  - Added reference to draft-keoh-tls-multicast-security to section 6 (Security Considerations).
  - Appendix B (CoAP-Observe Alternative to Group Communications) moved to draft-dijk-core-groupcomm-misc (#267).
  - Deleted section 8 (Conclusions) as it is redundant (#268).
  - Simplified light switch use case (#269) by splitting into basic operations and additional functions (#269).

Summary of Changes (2/4)

- Changes from ietf-03 to ietf-04 (Continued):
  - Moved section 3.7 (CoAP Multicast and HTTP Unicast Interworking) to draft-dijk-core-groupcomm-misc (#270).
  - Moved section 3.3.1 (DNS-SD) and 3.3.2 (CoRE Resource Directory) to draft-dijk-core-groupcomm-misc;
    Clarified that DNS based features are optional (#272).
  - Focus section 3.5 (Configuring Group Membership) on a single proposed solution.
  - Scope of section 5.3 (Use of MLD) widened to multicast destination advertisement methods in general.
  - Rewrote section 2.2 (Scope) for improved readibility.
  - Moved use cases that are not adressed to draft-dijk-core-groupcomm-misc.
  - Various editorial updates for improved readibility.

Summary of Changes (3/4)

- Changes from ietf-04 to ietf-05:
  - Added a new section 3.9 (Exceptions) that highlights that IP multicast (and hence group communications) is not always available (#187).
  - Included guidelines on when (not) to use CoAP responses to multicast requests and when (not) to accept multicast requests (#273).
  - Added guideline on use of core-block for minimizing response size (#275).
  - Restructured section 6 (Security Considerations) to more fully describe threats and threat mitigation (#277).
  - Clearly indicated that DNS resolution and reverse DNS lookup are optional.
  - Removed confusing text about a single group having multiple IP addresses. If multiple IP addresses are required then multiple groups (with the same members) should be created.
  - Removed repetitive text about the fact that group communications is not guaranteed.

Summary of Changes (4/4)

- Changes from ietf-04 to ietf-05 (Continued):
  - Merged previous section 5.2 (Multicast Routing) into 3.1 (IP Multicast Routing Background) and added link to section 5.2 (Advertising Membership of Multicast Groups).
  - Clarified text in section 3.8 (Congestion Control) regarding precedence of use of IP multicast domains (i.e. first try to use link-local scope, then site-local scope, and only use global IP Communication for CoAP multicast as a last resort).
  - Extended group resource manipulation guidelines with use of preconfigured ports/paths for the multicast group.
  - Consolidated all text relating to ports in a new section 3.3 (Port Configuration).
  - Clarified that all methods (GET/PUT/POST) for configuring group membership in endpoints should be unicast (and not multicast) in section 3.7 (Configuring Group Membership In Endpoints).
  - Various editorial updates for improved readability.

Discussion item (1/1)

- **Solution for Configuring Group Membership (no ticket#)**
  - Example of (unicast) configuring an endpoint to be part of one multicast group:

```
Req: POST /gp (Content-Format: application/json)
   { "n": "floor1.west.bldg6.example.com",
     "ip": "ff15::4200:f7fe:ed37:14cb" }
Res: 2.04 Changed
```

  - Where the “gp” resource has resource type (rt) “core.gp” which is defined for this purpose.
  - Question: Do we want specify such a solution (or for example leave it completely up to implementation)?

Open Issues(1/5)

- **Review Groupcomm requirements language (#271)**
  - Decision made to keep requirements language for informational draft
  - Each use of MAY/MUST/SHOULD etc. is to be reviewed.

Open Issues(2/5)

- **Add section on multicast via Proxy and its limitations (#274)**
  - Add a section to explain the issues & limitations of doing CoAP multicast requests via a CoAP Proxy.
  - Goal is to warn and guide implementers in this subject.
  - Also such a section would provide a placeholder/reminder of the problems that might be addressed further in the future.
  - Based on IETF 85 CoRE WG meeting discussion on Multicast CoAP requests via a Proxy. Discussion summary, for reference:
    - Aggregation of responses in a proxy is difficult, since it doesn't know when to stop aggregating responses.
    - SIP tried to deal with this problem but didn't solve it.
    - There may be an expectancy of CoAP client to get single response back from proxy, not multiple. The client in this case may not even know what it could expect. It may not even know the Proxy-URI contains a multicast target.
    - Presented via-proxy use case exposes gap in the core-coap specification regarding multicast via proxy. (Note: Expectation is that core-coap won’t address it in the first planned RFC release.)

Open Issues(3/5)

- **Issues with two/multiple Resource Directories in use case (#280)**
  - Some potential issues with RD came out of review Peter van der Stok and WG discussion IETF85.
  - There are two RD servers; use case shows that only one is found despite the fact that site-local multicast is used - should be two RDs found.
  - Proposal: simplify use case to a single RD server on the backbone to avoid such RD-specific issues like synchronization of RD information between servers.

Open Issues(4/5)

- **Add single-subnet configuration to use cases (#278)**
  - Suggested by review Peter van der Stok and IETF85 WG discussion, that it's best to start explaining the simple/basic cases and then gradually build up complexity.
  - Simplest case not shown yet would be a single subnet network configuration, which is currently not present in the I-D; only the 2-subnet configuration of Fig 1.
  - Question: Do we need to add such case, or doesn't it add much?
  - (Note that core-coap Figure 22 already shows the simplest case of link-local multicast.)

Open Issues(5/5)

- **Add use case with controller (client) located on the backbone (#279)**
  - Suggested by discussion following review Peter van der Stok:
  - Case missing where a controller (client) is located on the backbone.
  - Question: Do we need to add such case, or does it add much?

Group 2a: other old friends

Best Practices for HTTP-CoAP Mapping Implementation

Angelo Castellani, Salvatore Loreto, Akbar Rahman, Thomas Fossati, Esko Dijk

IETF 86, March 2013

http://www.ietf.org/id/draft-castellani-core-http-mapping-07.txt

Summary of Changes (from -06 rev)

- Based on discussion at last IETF-85 (Atlanta):
  - Clarification of HTTP-CoAP Proxies
    - Definition
    - Placement
    - Benefits
  - Clarification of HTTP-CoAP URI mapping options

Goals of I-D

- For Reverse HTTP-CoAP Cross Protocol Proxy:
  - Provide more detailed information to proxy designers (beyond Section 10 of [I-D.ietf-core-coap]), to help implement proxies that correctly inter-work with other CoAP and HTTP client/server implementations that adhere to the specifications
  - Define a consistent set of guidelines that a HTTP-to-CoAP proxy implementation MAY adhere to. The main reason of adhering to such guidelines is to reduce variation between proxy implementations, thereby increasing interoperability.
    - As an example use case, a proxy conforming to these guidelines made by vendor A can be easily replaced by a proxy from vendor B that also conforms to the guidelines

I-D Outline

- Guidance on HTTP to CoAP URI mapping
- HTTP-CoAP Reverse cross-protocol proxy implementation
  - Placement
  - Response code & media type translations
  - Caching and congestion control
  - Cache refresh via Observe
  - Use of CoAP blockwise transfer
  - Security translation
- Security Considerations
  - Traffic overflow
  - Handling secured exchanges

Definitions (1/2)

- Cross-Protocol Proxy (or Cross Proxy): is a proxy performing a cross-protocol mapping, in the context of this document a HTTP-CoAP (HC) mapping. A Cross-Protocol Proxy can behave as a Forward Proxy, Reverse Proxy or Interception Proxy.
  - Note: In this document we focus on the Reverse Proxy mode of the Cross-Protocol Proxy.

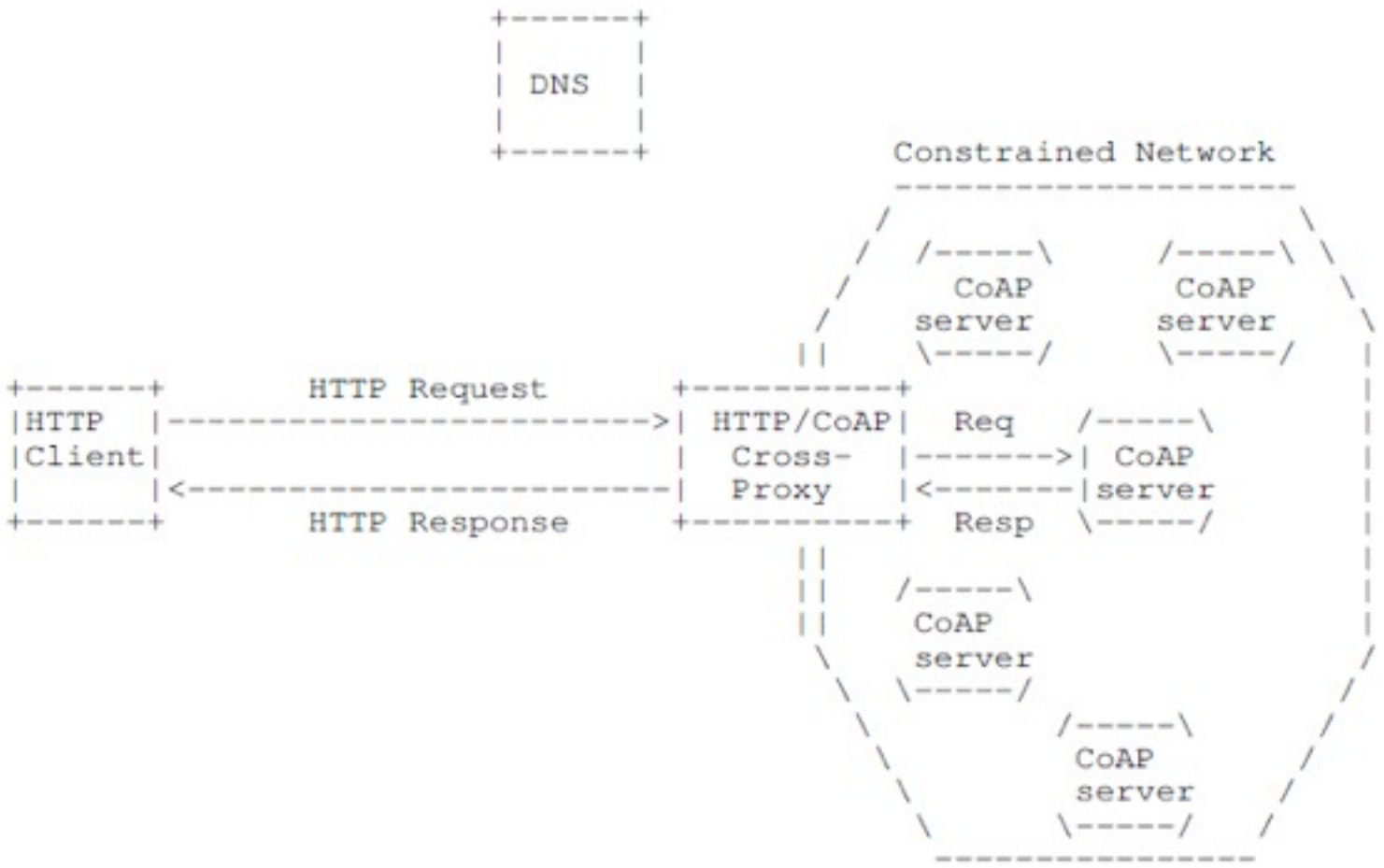
- Forward Proxy: a message forwarding agent that is selected by the client, usually via local configuration rules, to receive requests for some type(s) of absolute URI and to attempt to satisfy those requests via translation to the protocol indicated by the absolute URI. The user decides (is willing to) use the proxy as the forwarding/dereferencing agent for a predefined subset of the URI space.

Definitions (2/2)

- Reverse Proxy: a receiving agent that acts as a layer above some other server(s) and translates the received requests to the underlying server's protocol. It behaves as an origin (HTTP) server on its connection towards the (HTTP) client and as a (CoAP) client on its connection towards the (CoAP) origin server. The (HTTP) client uses the "origin-form" [I-D.ietf-httpbis-p1-messaging] as a request- target URI.

- Reverse and Forward proxies are technically very similar, with main differences being that the former appears to a client as an origin server while the latter does not, and that clients may be unaware they are communicating with a proxy.

Reverse Cross-Protocol Proxy Deployment Scenario



HTTP-CoAP Response Code Mapping

CoAP Response Code	HTTP Status Code	Notes
2.01 Created	201 Created	1
2.02 Deleted	200 OK	2
	204 No Content	2
2.03 Valid	304 Not Modified	3
	200 OK	4
2.04 Changed	200 OK	2
	204 No Content	2
2.05 Content	200 OK	
4.00 Bad Request	400 Bad Request	
4.01 Unauthorized	400 Bad Request	5
4.02 Bad Option	400 Bad Request	6
4.03 Forbidden	403 Forbidden	
4.04 Not Found	404 Not Found	
4.05 Method Not Allowed	400 Bad Request	7
4.06 Not Acceptable	406 Not Acceptable	
4.12 Precondition Failed	412 Precondition Failed	
4.13 Request Entity Too Large	413 Request Repr. Too Large	
4.15 Unsupported Media Type	415 Unsupported Media Type	
5.00 Internal Server Error	500 Internal Server Error	
5.01 Not Implemented	501 Not Implemented	
5.02 Bad Gateway	502 Bad Gateway	
5.03 Service Unavailable	503 Service Unavailable	8
5.04 Gateway Timeout	504 Gateway Timeout	
5.05 Proxying Not Supported	502 Bad Gateway	9

Implementation Experience

- Direct experience from the draft authors:
  - Squid HTTP-CoAP mapping module
    - University of Padova
    - http://telecom.dei.unipd.it/iot
    - Both Forward and Interception operation supported
  - HTTP-CoAP proxy based on EvCoAP
    - KoanLogic, University of Bologna and Salvatore Loreto (as individual)
    - https://github.com/koanlogic/webthings/tree/master/bridge/sw/lib/evcoap
- The document is open to input from other implementations

Next Steps

- Does the WG recommend adoption?
  - Intended status: Informational Best Practice
  - Purpose: Reduce arbitrary variation between proxy implementations, thereby increasing interoperability

Backup

HTTP to CoAP URI Mapping Options (1/2)

- Embedded Mapping
  - In an embedded mapping approach, the HTTP URI has embedded inside it the authority and path part of the CoAP URI.
  - Example: The CoAP resource "//node.coap.something.net/foo" can be accessed by an HTTP client by inserting in the request "http://hc-proxy.something.net/coap/node.coap.something.net/foo". The Cross-Protocol Proxy then maps the URI to "coap:// node.coap.something.net/foo"

HTTP to CoAP URI Mapping Options (2/2)

- Homogeneous Mapping
  - In a homogeneous mapping approach, only the scheme portion of the URI needs to be mapped. The rest of the URI (i.e. authority, path, etc.) remains unchanged.
  - Example: The CoAP resource "coap://node.coap.something.net/foo" can be accessed by an HTTP client by requesting "http://node.coap.something.net/foo". The Cross-Protocol Proxy receiving the request is responsible to map the URI to "coap://node.coap.something.net/foo"
  - Background info:
    - The assumption in this case is that the HTTP client would be able to successfully resolve "node.coap.something.net" using DNS infrastructure to return the IP address of the HC proxy. Most likely this would be through a two step DNS lookup where the first DNS lookup would resolve "something.net" using public DNS infrastructure.
    - Then the second DNS lookup on the subdomain "coap" and the host "node" would typically be resolved by a DNS server operated by the owner of domain "something.net". So this domain owner can manage its own internal node names and subdomain allocation which would correspond to the CoAP namespace

Group 3:“new work”

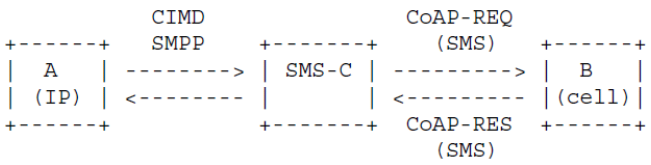
Transport of CoAP over SMS, USSD and GPRS draft-becker-core-coap-sms-gprs-03

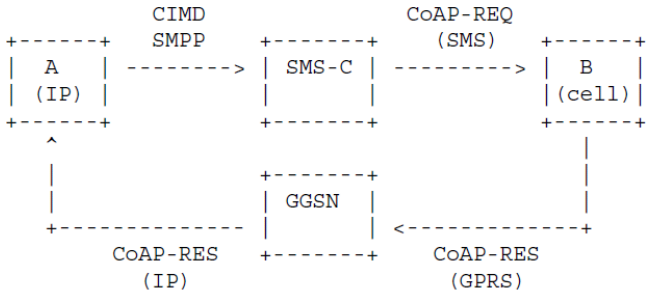
Markus Becker, Kepeng Li,
Koojana Kuladinithi, Thomas Pötsch

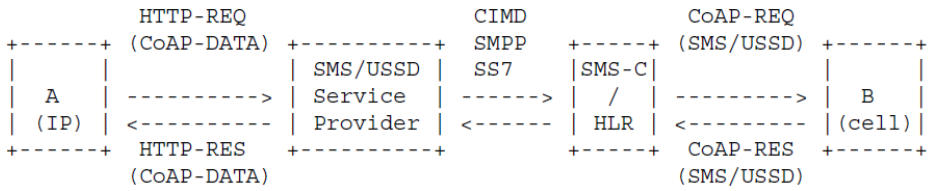
CoRE WG, IETF-86, Orlando

Scenarios

- In M2M communication, IP connectivity is not **always** supported by the constrained end-points
  - Power saving
  - Coverage (GPRS, 3G, LTE)
- SMS and USSD based communication is almost **always** supported







Changes from draft-01 to draft-03 (1)

- Added possible CON/NON/ACK interactions. Section 5
- Option name changed from Reply-To-* to Response-To-*.
  Section 16 and Section 10.1
- Added URI scheme.
  E.g.: coap+tel://+15105550101/.well-known/core. Section 11
- Added possible M2M proxy scenarios. Section 14

Changes from draft-01 to draft-03 (2)

- Added reference to bormann-coap-misc for other SMS encoding. Section 7.1
- Updated requirements on Uri-Host and Uri-Port for coap+tel://. Section 10
- Added security considerations: Transport and Object Security. Section 15
- Chose CoAP option numbers and updated the option number table to meet draft-ietf-core-coap-10. Table 1
- Added an IANA registration for the URI scheme coap+tel. Section 16.2

Feedback: Split coap+tel:// into coap+sms:// and coap+ussd://

- How else to differ the various transports in the URI?
- Or is the transport to use decided by the implementation?
- See also draft-silverajan-core-coap-alternative-transports-01

Feedback: Response-To-Scheme option

- Should there be a Response-To-Scheme option additionally to Response-To-Uri-Host and Response-To-Uri-Path?
  - So that a CoAP end-point which receives a request by CoAP over non-IP transport, can be instructed to also use the non-IP transport for the response?
  - This would imply to add a Response-To-Telephone-Subscriber and more options for other transports that have other URI schemes than coap:// and coaps://.

Feedback: Selection of Transport by Client

- When a server is reachable by various transports, how does a client decide which transport to use?
  - Client-side application configuration?
  - Leave it to a proxy, which uses cellular network lookup functions?

Feedback: Potential Threats

- Trigger expensive short messages
- Redirect traffic to other addresses
- More threats?

- Are the security measures in the draft adequate?
  - Whitelisting on server
  - Relying on cellular network security (dedicated M2M APNs)
  - Object security on CoAP payload

Feedback: Message Exchanges

- Figures 11-18 show possible message exchanges when client and server have two addresses.
  - With NON: Not using CoAP retransmission capability.
  - ACK on different transport.
  - Additional (costly) message on Transport A.
  - Request with Content.
- Which ones are allowed?
- Which ones are meaningful?

Next steps

- Refine Uri scheme
- Response-To-Scheme option
- Selection of Message Exchange
- Refine Proxying

CoAP Communication with Alternative Transports

Bill Silverajan and Teemu Savolainen

Core WG, IETF 86, Orlando

Objectives (In Scope)

- Unambiguous representation of transport to be used by CoAP
- Details how the transport can carry CoAP packet
- Focus is on what CoAP communications requirements are
- Allow CoAP mechanisms to exploit cross-layer optimisations

Non Goals (Not in scope)

- Application-level QoS requirements
- Real-time constraints
- Ordering, reliability, congestion control
- Application and network adaptation
- Mobility and readdressing support
- Network protocol (eg IPsec) configuration

Use Cases

- NAT and Firewall traversal with TCP
- Delay Tolerant Networking
- Non-IP Low Energy Protocols

Related Work

- OMA Lightweight M2M Protocol
- CoAP over SMS/GPRS/USSD

What does the work entail?

- The ability to specify end point identifiers and URIs to reflect which transport CoAP can use
- The ability to register to resource directories resource types reflecting types of transport available
- Ability for multi-transport nodes to select transport to use without significant alteration to CoAP packet structure

Points to consider

- Requirements for transport layer protocol to carry CoAP packets
- Representation of URI scheme for generic transport protocols
- Option extensions for CoAP to express available transports
- Mapping and message size
- Security considerations

URI Representations for CoAP

Means of expressing transport types

URI = scheme ":" "//" authority path-abempty ["?"query]

- Within the scheme name
- In the URI path
- As a query component

URI Representations for CoAP

Means of expressing transport types

- Within the scheme name
  - coap+tel://+15105550101/.well-known/core
  - coap+tcp://example.com:5683/temperature
  - coap+ble.l2cap://[12:34:56:78:90:AB]:4/pulse
- Easy URI parsing for transport type and identifier
- Each new scheme name requires IANA registration

URI Representations for CoAP

Means of expressing transport types

- In the URI path
  - coap://host.example.com;transport=tcp/.well-known/core?rt=core-rd
- Easy adoption of new (and experimental) CoAP transports without IANA registration
- Caveat: Uri-Path option is a Critical CoAP option, see section 5.4.1 of I-D.ietf-core-coap

URI Representations for CoAP

Means of expressing transport types

- As a URI query component
  - coap://host.example.com/.well-known/core?rt=core-rd?tt=tcp
- Also easy to adopt new (and experimental) CoAP transports without IANA registration
- Caveat: Uri-Query option is a Critical CoAP option, see section 5.4.1 of I-D.ietf-core-coap

URI Representations for CoAP

Non standard ways

- Transport in the URI authority component
  - coap://host[:port][transport]/ instead of coap://host:[port]/
- Transport as a service: URL
  - service:coap:tcp://host.example.com/.well-known/core?rt=core-rd
- Above two ways break existing compatibility (although both are based on valid IANA URI registrations)

URI Representations for CoAP

Miscellaneous Considerations

- Ensure that transport is defined unambiguously (eg L2CAP used by both BLE and standard Bluetooth, so avoid ”coap+l2cap” or ”;transport=l2cap” or ”?tt=l2cap”
  - Can be done as WG consensus
  - Namespace approach, eg ”coap+ble.l2cap” or ”?tt=ble.l2cap”
- Allow end-points to use CoAP Resource Directory to register alternative transports and identifiers and provide periodic updates
  - Register ”core-transport” resource type

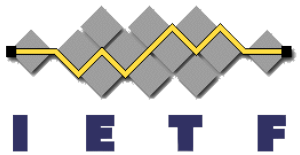
Transport Considerations

- Uniqueness of end-point identification
- Communication support: Unidirectional, Bidirectional, 1:N (broadcast, multicast, anycast) messaging
- Binary encoding, Network byte ordering
- MTU correlation with CoAP PDU size
- Transport latency

Security Considerations

- Draft does not introduce new security requirements by itself
- There may be privacy issues if nodes use telephone numbers or MAC addresses as public end point identifiers

Enhanced Sleepy Node Support for CoAP

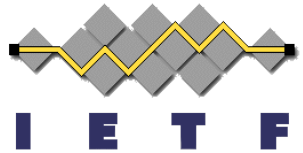


Akbar Rahman

IETF 86, March 2013

http://www.ietf.org/id/draft-rahman-core-sleepy-02.txt

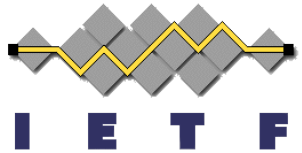
Introduction



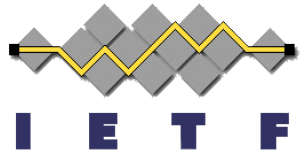
- It is expected that in CoAP networks there will be a certain portion of devices that are "sleepy" and which may occasionally go into a sleep mode (i.e. go into a low power state to conserve power) and subsequently have reduced CoAP protocol communication ability

- This I-D proposes a minimal and efficient mechanism building on the Resource Directory concept (which will be integrated into a CoAP Proxy) to enhance sleepy node support in CoAP networks

Main Question from IETF-85 (Atlanta)

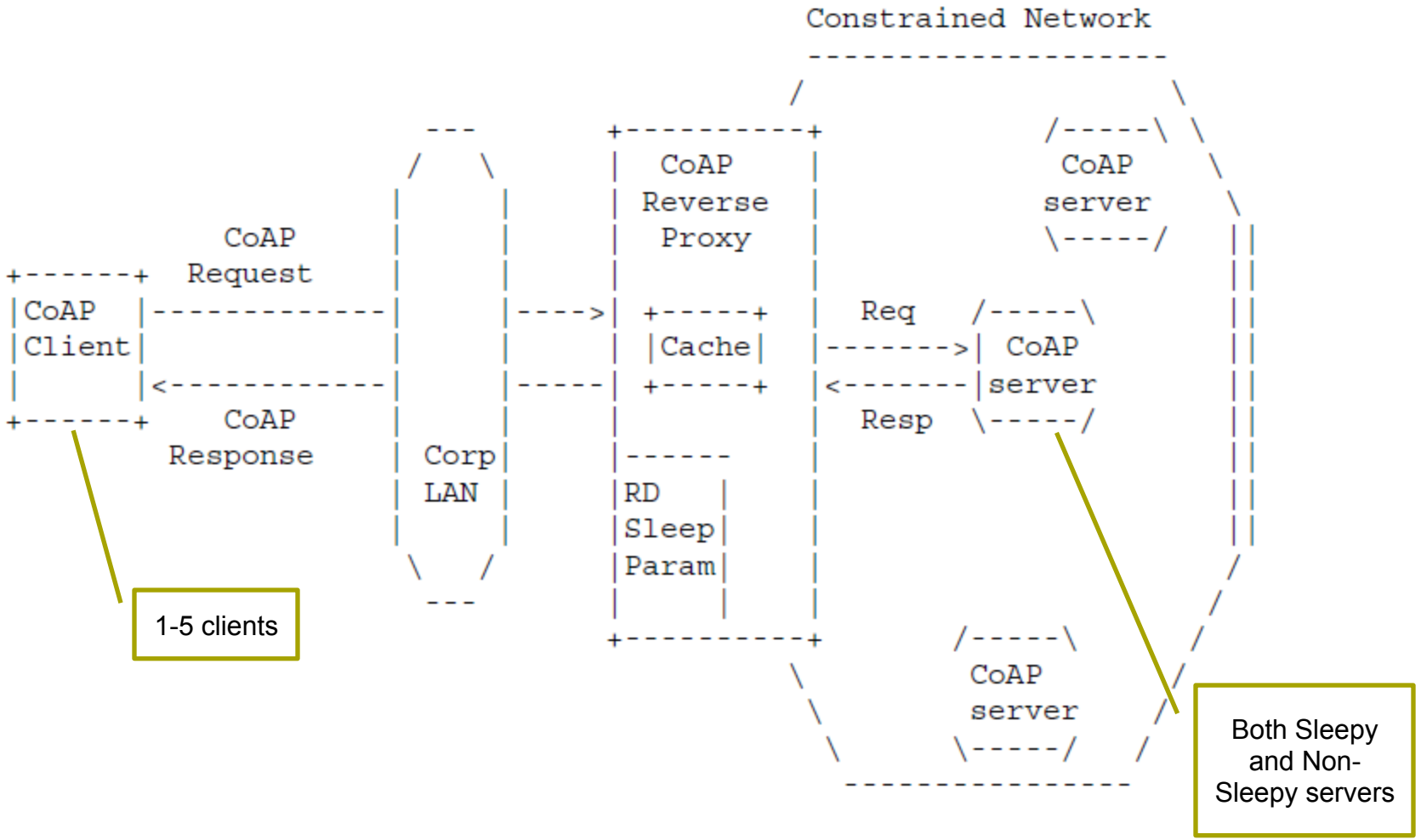


- What performance benefit does the proposed Sleepy Node support solution give in a network that already has standard CoAP caching enabled?



Sleepy Node Performance Analysis

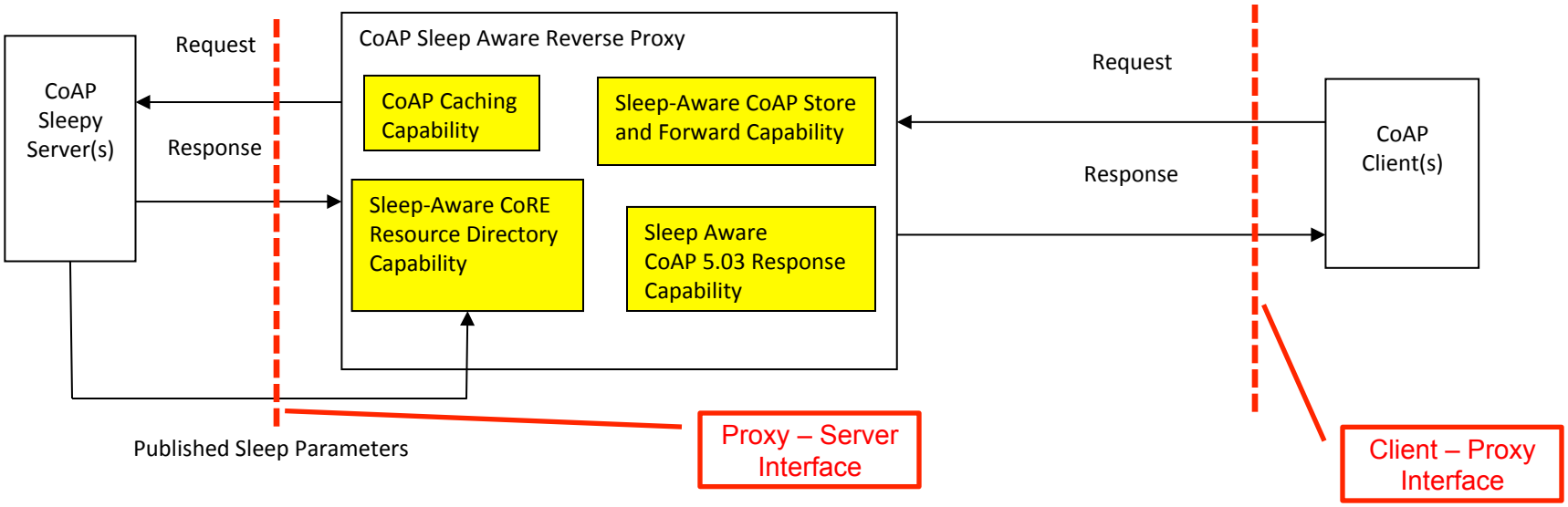
Experimental Network



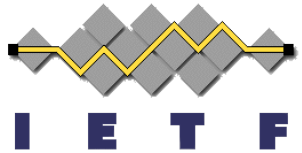
Sleepy Node Performance Analysis Network Setup

- Server supports publishing sleep parameters to proxy
  - E.g. I'm going to be asleep for *X* seconds
- Proxy supports sleep-awareness capabilities
- Proxy also supports caching capability based off of maxAge
- Protocol flow similar to approach of Figure 1 (Synchronous RD Based Sleep Tracking) of I-D

** Proxy capabilities can be selectively enabled/disabled*

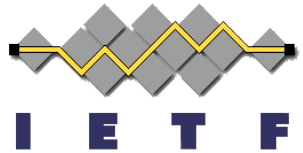


CoAP Sleep-Aware Reverse Proxy Features



- CoRE Sleep-aware Resource Directory
  - Support storing published sleep parameters from CoAP servers
    - sleepState (AWAKE, ASLEEP)
    - sleepDuration
- Sleep-aware CoAP 5.03 Response Capability
  - If CoAP request to a sleeping server is received, proxy returns a '5.03 Retry-After' response to client.
    - 5.03 contains a timestamp indicating when the sever will wake back up (timestamp delivered in CoAP maxAge option)
- Sleep-aware CoAP Store-and-Forward Capability
  - If CoAP request to a sleeping server is received, proxy stores request until server wakes up and then forwards it
- Caching capability
  - Cache GET responses from server if maxAge option is present (this is not a sleep aware feature)

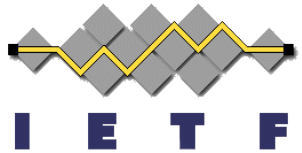
Goals of Performance Analysis



- For networks having sleepy servers, provide measurements to quantify the impact that CoAP sleep-awareness capabilities can have

- See if sleep-awareness capabilities can provide additional benefits even when CoAP caching is used

Overview of Performance Analysis Performed

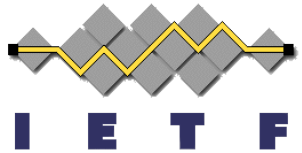


- The analysis was broken up into a set of test scenarios
- Each test scenario used a different combination of the following variables
  - Client issued GET requests
  - Server was configured as a sleepy or non-sleepy server
  - Proxy caching of GET responses was enabled or disabled
  - Proxy sleep-aware 5.03 response capability was enabled or disabled
  - Proxy store-and-forward capability was enabled or disabled
  - maxAge in GET responses – Two values were tested 60sec and 5sec
  - # of Clients – Two values were used – 1 client and 5 clients
- For comparison purposes, the following were held constant across all test scenarios
  - Client issued a fixed number of requests (100) across all test scenarios
  - Client used a delay between each GET request (35sec) to better exercise proxy's caching and sleep-awareness capabilities
  - For sleepy server test scenarios, the server slept 60 sec / awake for 3 sec

Format of Results

The following results were collected for each test scenario:

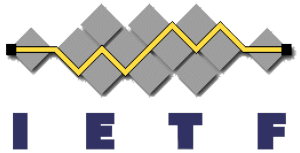
- Breakdown of the # and types of transactions occurring between client/proxy/server



“GET” – Test Scenario Results

GET – Description of Test Scenarios

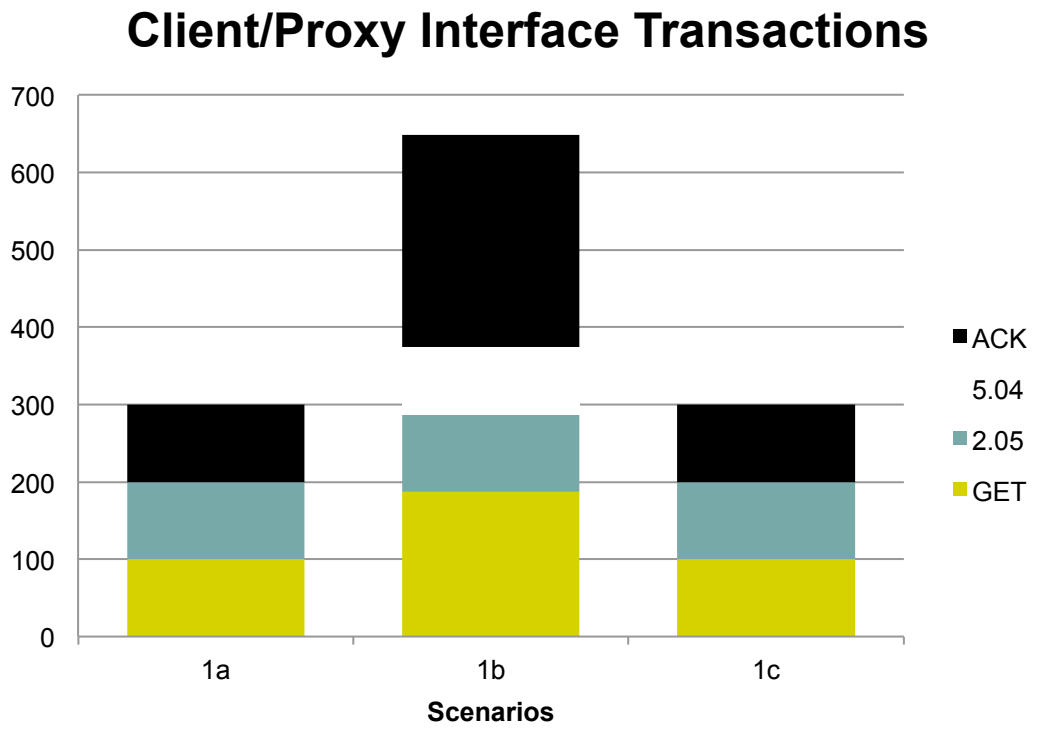
Test Scenarios - GET								
Scenario	CoAP Sleepy Server Settings			CoAP Sleep Aware Reverse Proxy Settings		CoAP Client Settings		
	Sleep Duration (sec)	Awake Duration (sec)	maxAge (sec)	Caching	Sleep Aware SAF + 5.03	Number of Clients	Number of GETs	Delay Between Client Requests (sec)
1a	0	Always	60	Enabled	Disabled	1	100	35
1b	60	3	60	Enabled	Disabled	1	100	35
1c	60	3	60	Enabled	Enabled	1	100	35
2a	0	Always	60	Enabled	Disabled	5	100	35
2b	60	3	60	Enabled	Disabled	5	100	35
2c	60	3	60	Enabled	Enabled	5	100	35
3a	0	Always	5	Enabled	Disabled	5	100	35
3b	60	3	5	Enabled	Disabled	5	100	35
3c	60	3	5	Enabled	Enabled	5	100	35



“GET” – Test Scenario 1 Results

→ maxAge = 60, and 1 Client

GET – Scenario 1 Client/Proxy Interface Transaction Mix

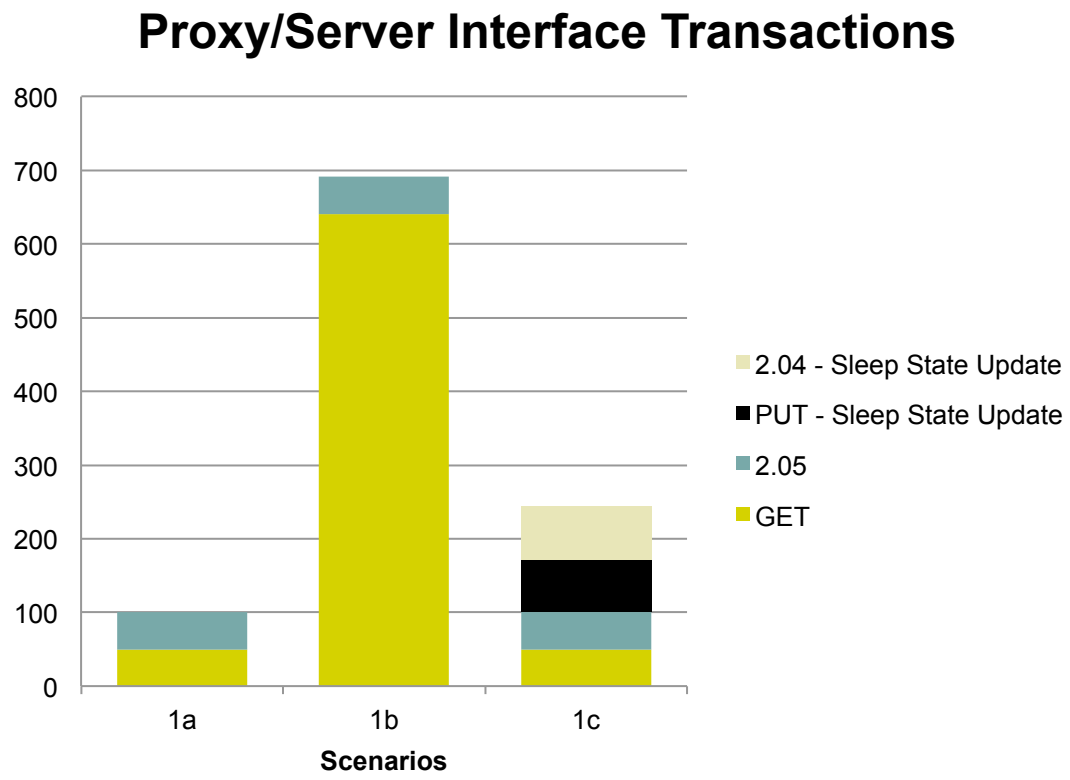


Scenarios:

1a - Non-Sleepy Server, Caching Proxy, maxAge = 60sec
1b- Sleepy Server, Caching Proxy, maxAge = 60sec
1c - Sleepy Server, Sleep Aware Proxy, maxAge = 60sec

→ Sleep Aware Proxy has better performance for sleepy servers

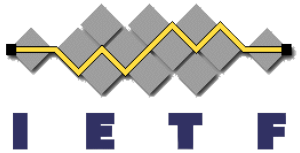
GET – Scenario 1 Proxy/Server Interface Transaction Mix



Scenarios:

1a - Non-Sleepy Server, Caching Proxy, maxAge = 60sec
1b- Sleepy Server, Caching Proxy, maxAge = 60sec
1c - Sleepy Server, Sleep Aware Proxy, maxAge = 60sec

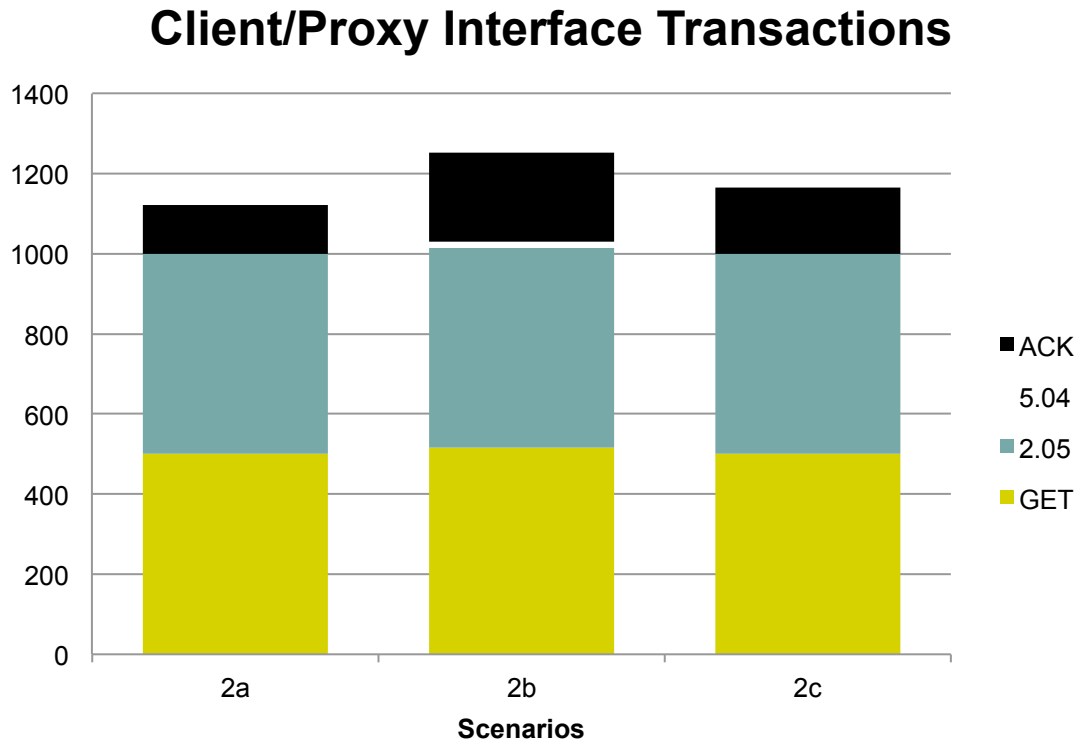
→ Sleep Aware Proxy has better performance for sleepy servers



“GET” – Test Scenario 2 Results

→ maxAge = 60, and 5 Clients

GET – Scenario 2 Client/Proxy Interface Transaction Mix



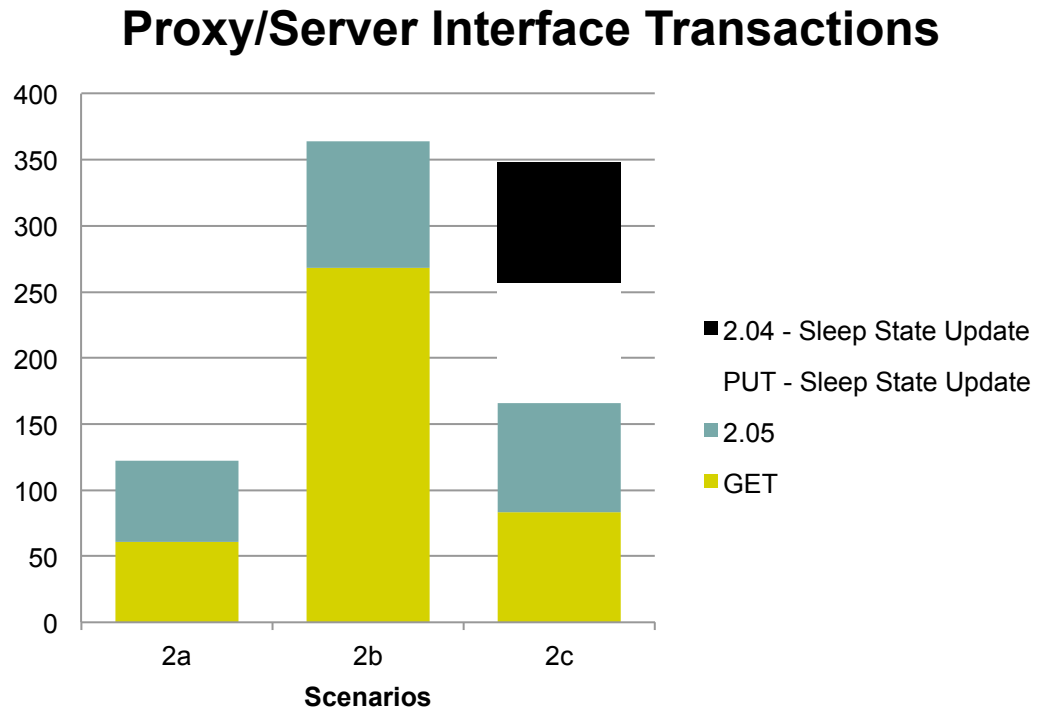
Scenarios:

2a - Non-Sleepy Server, Caching Proxy, maxAge = 60sec, 5 Clients
2b- Sleepy Server, Caching Proxy, maxAge = 60sec , 5 Clients
2c - Sleepy Server, Sleep Aware Proxy, maxAge = 60sec , 5 Clients

→ Sleep Aware Proxy has slightly better performance for sleepy servers

See "Experimental Artifacts" Note 1

GET – Scenario 2 Proxy/Server Interface Transaction Mix



Scenarios:

2a - Non-Sleepy Server, Caching Proxy, maxAge = 60sec, 5 Clients
2b- Sleepy Server, Caching Proxy, maxAge = 60sec , 5 Clients
2c - Sleepy Server, Sleep Aware Proxy, maxAge = 60sec , 5 Clients

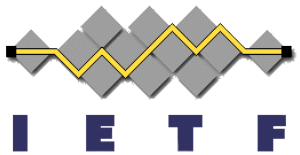
→ Sleep Aware Proxy has slightly better performance for sleepy servers

See “Experimental Artifacts” Note 2

Experimental Artifacts

- Note 1:
  - Currently in the experiment, the sleep aware proxy checks the cached state of a resource before putting a request into its Store-and-forward queue while a server is sleeping. Once in the queue, the proxy unconditionally forwards the queued requests to the server once it wakes up. This is not the most efficient setup and can easily be improved. If the proxy instead checks its cache again before forwarding each queued request it may not need to forward the request to the server. For example, if there are two queued GET requests in the proxy's queue that address the same resource. Only the first request needs to be sent to the server, the second request can use the cached response from the first request.

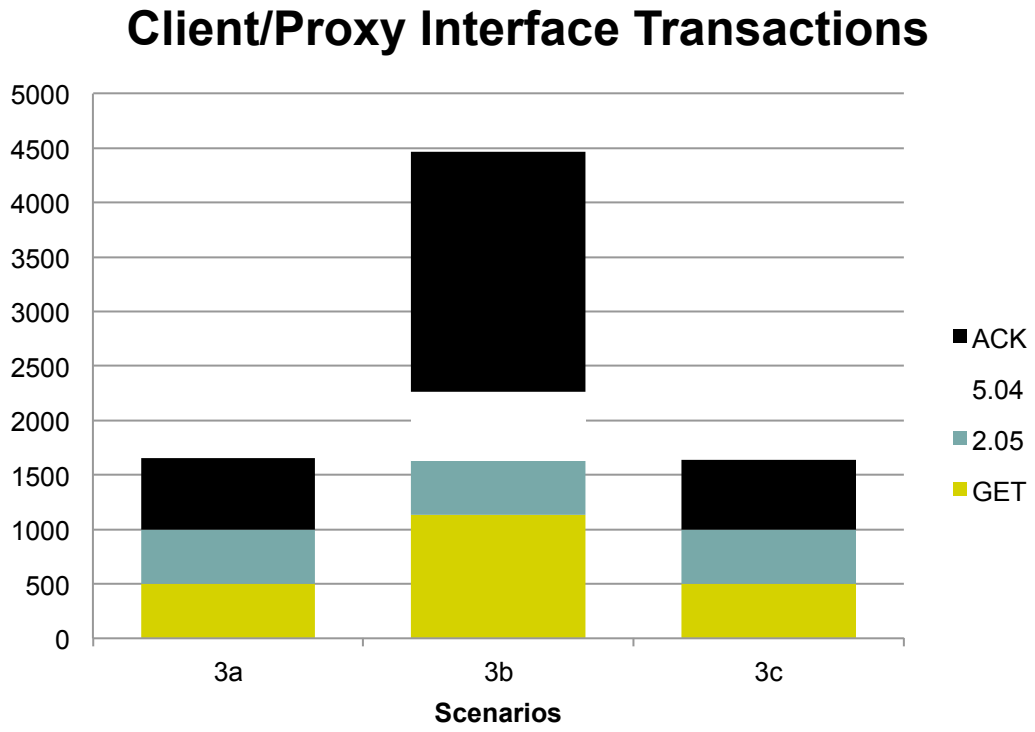
- Note 2:
  - Currently in the experiment, the sleepy server sends a sleepState PUT request to the proxy each time it goes to sleep. This is not the most efficient setup and can easily be improved. For example, if the server assumes that the proxy can keep track of its ON/OFF cycles with its internal timers (e.g. as per Fig. 1 of I-D) then the number of messages from the sleepy server dramatically goes down.



“GET” – Test Scenario 3 Results

→ maxAge = 5, and 5 Clients

GET – Scenario 3 Client/Proxy Interface Transaction Mix

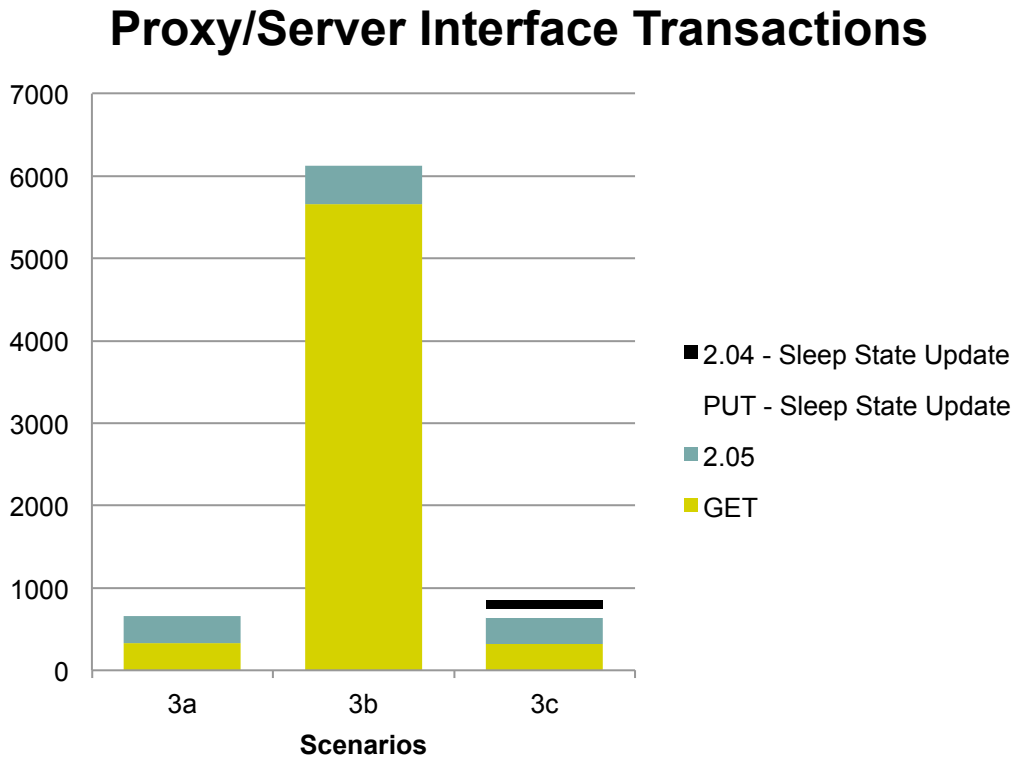


Scenarios:

3a - Non-Sleepy Server, Caching Proxy, maxAge = 5sec, 5 Clients
3b- Sleepy Server, Caching Proxy, maxAge = 5sec, 5 Clients
3c - Sleepy Server, Sleep Aware Proxy, maxAge = 5sec, 5 Clients

→ Sleep Aware Proxy has better performance for sleepy servers

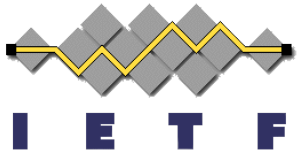
GET – Scenario 3 Proxy/Server Interface Transaction Mix



Scenarios:

3a - Non-Sleepy Server, Caching Proxy, maxAge = 5sec, 5 Clients
3b- Sleepy Server, Caching Proxy, maxAge = 5sec, 5 Clients
3c - Sleepy Server, Sleep Aware Proxy, maxAge = 5sec, 5 Clients

→ Sleep Aware Proxy has better performance for sleepy servers



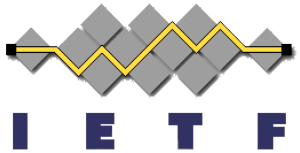
Conclusions

Conclusions

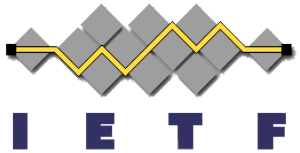
- These results show that sleep-aware CoAP proxy features can significantly optimize communication with sleepy servers in most scenarios

- These results also show that sleep-awareness capabilities can provide additional benefits above and beyond CoAP caching in most scenarios

Backup



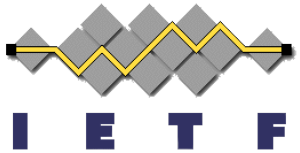
Current CoAP Support of Sleepy Node (1/2)



- CoAP proxies can use a previously cached response to service a new GET request for a sleepy origin server (as in HTTP)
  - But if no valid cache then proxy has to attempt to retrieve and may fail if origin server is sleeping
  - [I-D.ietf-core-coap]

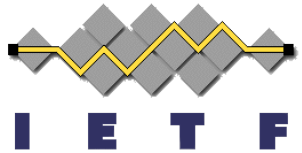
- Clients can discover list of resources from RD (GET /rd-lookup/…) for sleepy servers
  - But attempt to GET resource from sleepy origin server may fail if origin server is sleeping
  - [I.D.ietf-core-link-format & I.D.shelby-core-resource-directory]

Current CoAP Support of Sleepy Node (2/2)



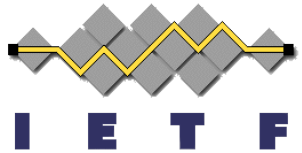
- Lower layer support for sleepy nodes in most wireless technologies (e.g. WiFi, ZigBee).
  - But limited to MAC packet scheduling for sleepy nodes and not aware of specific needs of IP applications (like CoAP)

Proposal – RD Based Sleep Tracking (1/4)



- The current CoAP approach to support sleepy nodes can be significantly improved by introducing RD based mechanisms for a CoAP client to determine whether:
  - A targeted resource is located on a sleepy server
  - A sleepy server is currently in sleep mode or not
  - There is any associated caching Proxy (possibly the RD itself) for a sleepy server

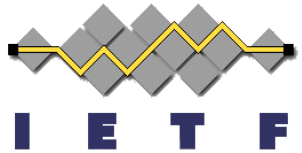
Proposal – RD Based Sleep Tracking (2/4)



- We define the following new RD attributes to characterize the properties of a sleepy node:
  - SleepState - Indicates whether the node is currently in sleep mode or not (i.e. Sleeping or Awake)
  - SleepDuration - Indicates the maximum duration of time that the node stays in sleep mode
  - TimeSleeping - Indicates the length of time the node has been sleeping (i.e. if Sleep State = Sleeping)
  - NextSleep - Indicates the next time the node will go to sleep (i.e. if Sleep State = Awake)

  - CachingProxy – Indicates the caching proxy of the sleepy node (i.e. the RD itself or another node)

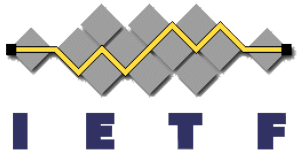
Proposal – RD Based Sleep Tracking (3/4)



- These attributes are all server (node) level and are new parameters added to the RD URI Template Variables

- Finally, we also define a new lookup-type ("ss") for the RD lookup interface specified in [I-D.shelby-core-resource-directory].
  - This new lookup-type supports looking up the “SleepState” (ss) of a specified end-point

Proposal – RD Based Sleep Tracking (4/4)



- The three time based parameters (SleepDuration, TimeSleeping, NextSleep) can be based on either an absolute network time (for a time synchronized network) or a relative local time (measured at the local node)

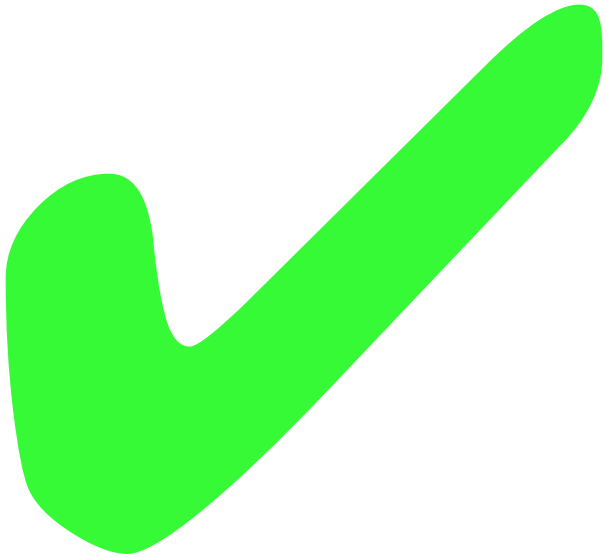
- Following the approach of [I-D.ietf-core-link-format] and [I-D.shelby-core-resource-directory], sleep parameters for sleepy servers can be stored by the server in the RD and accessed by all interested clients

- Examples of using these parameters in a synchronous or asynchronous manner are shown in the I-D

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

✓Blue sheets
✓Scribe(s)

draft-ietf-core-coap submitted to IESG



2013-03-13

OMA Lightweight M2M Overview

(A new standard using lots of IETF specs including DTLS, CoAP, Block, Observe, Resource Directory, SenML…)

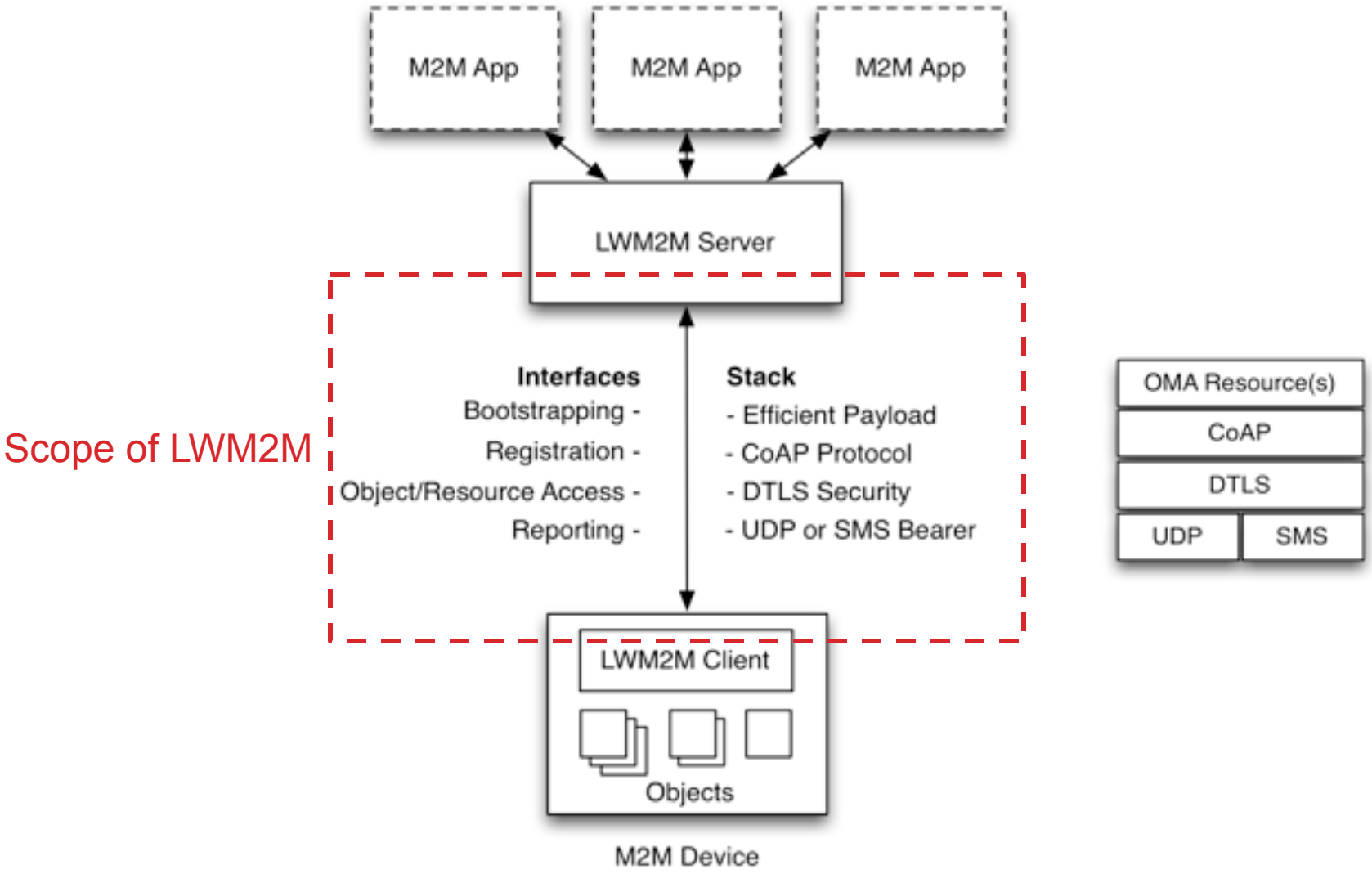
Zach Shelby

CoRE WG @ IETF-86 Orlando

OMA Lightweight M2M

- Open Mobile Alliance is well known for Device Management (DM)
- OMA Lightweight M2M is a new standard from the alliance
  - Focused on constrained Cellular and sensor network M2M devices
  - Driven by leading operators and vendors
- Scope
  - Interfaces, protocol & security between Device and Server
  - Object and Resource model
  - Bootstrap, Device, Access Control, Connectivity and Firmware Objects
- Draft Specifications are available
  - http://member.openmobilealliance.org/ftp/Public_documents/DM/LightweightM2M/Permanent_documents/OMA-TS-LightweightM2M-V1_0_0-20130301-D.zip
- Timeline
  - Requirements and Architecture completed 3Q/2012
  - Spec ready for Consistency Review 2Q/2013
  - Candidate Approval expected 3Q/2013

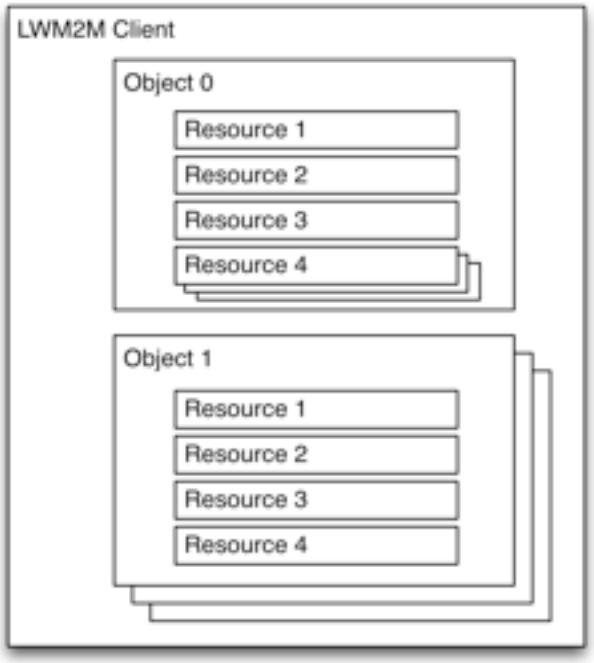
Architecture



Object Model

- A Client has one or more Object Instances
- An Object is a collection of Resources
- A Resource is an atomic piece of information
  - Read, Written or Executed
- Resources can have multiple instances
- Objects and Resources are identified by a 16-bit Integer, Instances by an 8-bit Integer
- Objects/Resources are accessed with simple URIs:

```
/{Object ID}/{Object Instance}/{Resource ID}
e.g. /12/1/3
```



CoRE Resource Directory

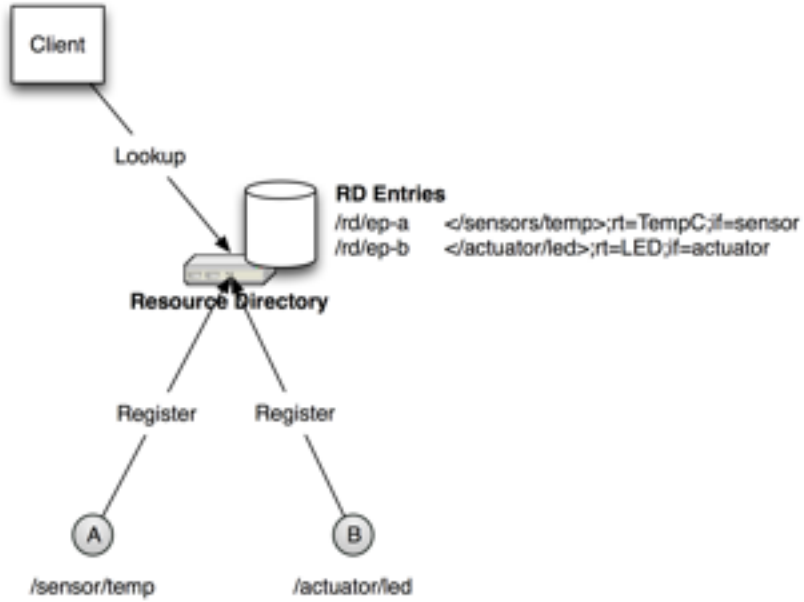
draft-shelby-core-resource-directory-05

Z. Shelby, C. Bormann, S. Krco

CoRE WG, IETF-86 Orlando

Background

- Not a new concept
  - think web search engine or any link directory
- Defines the interfaces to a Resource Directory
- Based on Web Linking framework and the CoRE Link Format
- Generic REST design for use over HTTP and CoAP
- Used by OMA Lightweight M2M
- Has already been deployed
  - In traffic monitoring systems
  - In street lighting systems
  - For vehicular asset tracking
  - By a major Cellular M2M operator



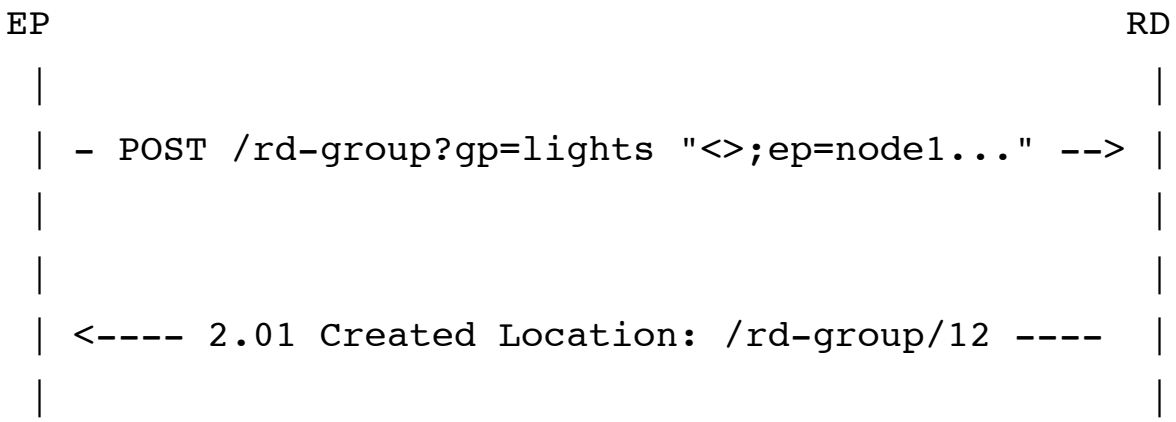
Changes since -04

- Restricted Update to parameter updates
- Added pagination support for the Lookup interface
- Changed rt= to et= for the registration & update interface
- **Added group support**

Group Management

- RD now includes group management features
- New Group Function Set
  - Register a group
    - Group name, domain, multicast address, endpoint members
  - Remove a group
- Group lookup using the Lookup Function Set
  - List registered groups
  - Find the members of a group
  - Find groups with certain endpoints or resources

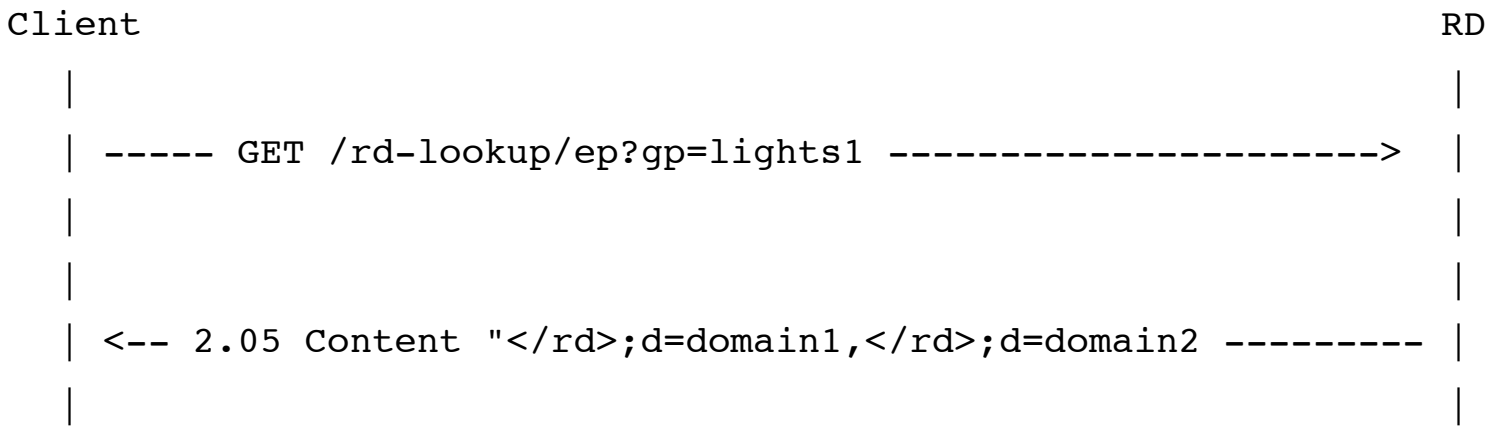
Registering a Group




```
Req: POST coap://rd.example.com/rd-group?gp=lights
Payload:
<>;ep="node1",
<>;ep="node2"

Res: 2.01 Created
Location: /rd-group/12
```

Looking up Group Members



Req: GET /rd-lookup/ep?gp=lights1

Res: 2.05 Content
<coap://host:port>;ep="node1",
<coap://host:port>;ep="node2”

Things for the future

- Support for JSON representation of Web Links
  - http://tools.ietf.org/id/draft-bormann-core-links-json-02.txt
- Registry for RD registration query parameters
  - Already OMA Lightweight has defined some new ones
- More examples

Group 3:“new work” (continued)

draft-greevenbosch-core-profile-description-01

Bert Greevenbosch
Jeroen Hoebeke
Isam Ishaq
Floris Van den Abeele

Description

- JSON format for signalling the server's capabilities.
- Signalling of supported options, media types and block size.
  - Other items can be added.
- Filtering on specific fields through URI-Query.
- Link: http://datatracker.ietf.org/doc/draft-greevenbosch-core-profile-description/

.well-known/profile

- To acquire profile data of resources on a particular service, the .well-known/profile URI-path is introduced.
- For example, to get all information from sensors served by www.example.org, we can do a GET to coap://www.example.org/.well-known/profile.
- Filtering on particular resources is done through URI-Queries.

Format

- Currently the following fields are defined:
  - “path”: contains the URI-path associated with a resource;
  - “op”: a numerical list of supported option numbers;
  - “cf”: a numerical list of supported content-format numbers;
  - “b1s”, “b2s”: supported block sizes for Block1 and Block2, respectively.

Example

- On the right is an example of a camera sensor at "coap://www.example.org/cam", that supports the "Uri-Host" (3), "ETag" (4), "Uri-Port" (7), "Uri-Path" (11), "Content-Format" (12), "Token" (19), "Block2" (23) and "Proxy-Uri" (35) options.
- The supported content formats are "text/plain" (0), "application/ link-format" (40) and "application/ json" (50).
- The supported Block2 can use 256 or 512 byte blocks.

```
Req:
GET coap://www.example.org/.well-known/profile
Res:
2.05 Content (application/json)
{
  "profile":
  {
    "path":"cam",
    "op":[3,4,7,11,12,19,23,35],
    "cf":[0,40,50],
    "b2s":[4,5]
  }
}
```

Filtering through use of URI-Query

- Filtering can be done through the URI-Query.
- Query format: *N*=*V*
  - *N* is a profile field;
  - *V* is the desired value.
- Examples:
  - To find resources that support content format "application/json":

    GET www.example.org/.well-known/profile?cf=50
  - To get information about the camera:

    GET www.example.org/.well-known/profile?path=cam

Open issues

- Which other profile data needs signalling?
  - Content-type for different methods?
- Fix the order in which the profile fields must appear?
  - Non-fixed order easier extensible?
- Inheritance of a profile description?
- Extend usage to signal the client profile?
- Integration with link format?

Thank you!

Questions?

draft-greevenbosch-core-minimum-request-interval-00

Bert Greevenbosch

Description

- The “MinimumRequestInterval" option can be used to indicate the minimum time between two requests in a transaction.
- Originally intended for Block, but also usable for other transactions.
  - Example: browsing a server.
- The goal is to reduce the server load and network traffic.
- Link: https://datatracker.ietf.org/doc/draft-greevenbosch-core-minimum-request-interval/

Usage

- The client keeps the server informed about its request speed through inclusion of the “MinimumRequestInterval” option.
- In responses, the server fixes the minimum request interval through the same option.
- The client obeys the speed indicated by the server.
- The client can send slower, but not faster.

Advantages

- The server has means to limit the amount of incoming traffic.
- The server can prevent to become overloaded with too many tasks.
- The server does not need artificially slow down the client by sending late ACKs.
  - No need to keep track of delayed ACKs.
  - The server can perform other tasks instead.
- Reduced network traffic.

Thank you!

Questions?

draft-bormann-core-links-json-02.txt

- RFC 6690 (link-format) documents are somewhat foreign to many web app developers
  - would prefer to have them in JSON format
- There is no standard way to represent link-format documents in applications
  - but everyone knows how to handle JSON

➔ Define a standard JSON translation for link-format

```
</sensors>;ct=40;title="Sensor Index",
</sensors/temp>;rt="temperature-c";if="sensor",
</sensors/light>;rt="light-lux";if="sensor",
<http://www.example.com/sensors/t123>
  ;anchor="/sensors/temp";rel="describedby",
</t>;anchor="/sensors/temp";rel="alternate"
```

➔

```
[{"href":"/sensors","ct":"40","title":"Sensor Index"},
  {"href":"/sensors/temp","rt":"temperature-c","if":"sensor"},
  {"href":"/sensors/light","rt":"light-lux","if":"sensor"},
  {"href":"http://www.example.com/sensors/t123",
   "anchor":"/sensors/temp","rel":"describedby"},
  {"href":"/t","anchor":"/sensors/temp","rel":"alternate"}]
```

draft-bormann-core-ipsec-for-coap-00.txt

- IPsec definitions for CoAP
- Mostly placeholder at this time
- Who has implementation experience with this?

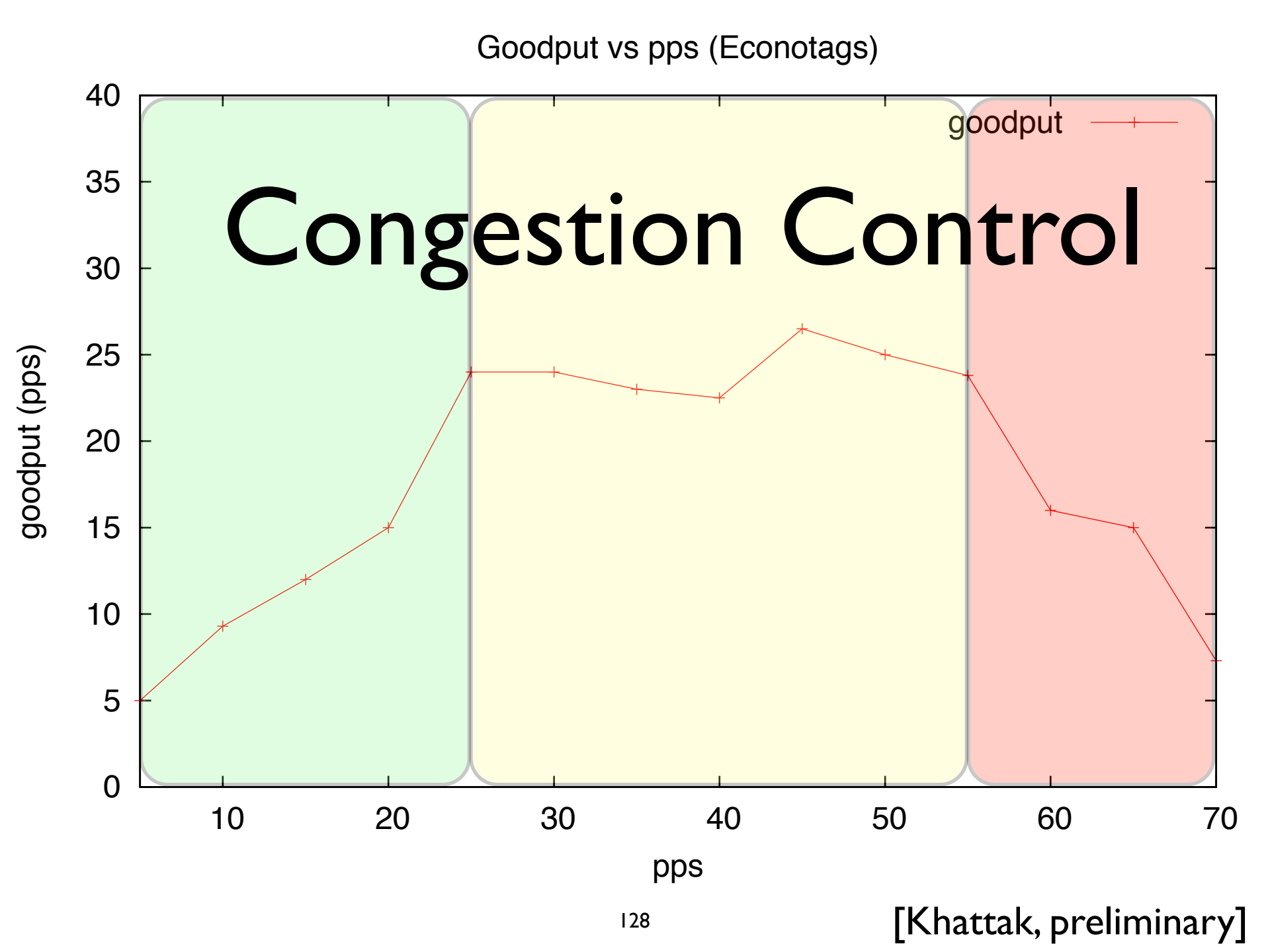
draft-bormann-core-cross-reverse-convention-00.txt

- This draft has two lines of payload (based on a WGLC comment by Cullen Jennings):

- coap://1.2.3.4:4567/foo/bar?a=3 →
- http://www.proxy.com/.wellknown/core-translate/1.2.3.4_4567/foo/bar?a=3

- Is that a useful convention to have?

Insert CI slides here



[Khattak, preliminary]

draft-bormann-cocoa-00.txt

- “advanced” cc
- (The draft is just a first shot)
- Others may have better ideas

Insert post-WGLC discussion slides here

Stop, Continue, Start

An Agile Retrospective

Stop

- **Endless new CoAP options**
  - **instead, use RESTful as much as possible**
- **Move general security lifecycle out**
- **Looking at theoretical problems**

Continue

- Interops
- Running code
- Look at real-world examples and deployments
- Solve their actual problems
  - Unilateral enhancements, e.g. higher performance congestion control
  - Transports (SMS, …)
  - Make Web linking and Directories more and more useful
  - RESTful Design Paradigms for real-world problems

Start

- Further push into the (browser) web world (HTML5, browser security)
- Specific supporting work
  - SenML?
  - URI work (dev URN, …)
  - Security Processes (next slide)
- Spin out?
  - (We don't really get to decide where this is done.)

Security

- Integrate Application, Network, Group Keying
  - Tunnel-free setup
- Don’t redo what has been done already
  - ZigBee IP is fine
  - But what about other scenarios?
- Simple REST interfaces for working with keying materials and authorization
  - Start with defining objectives
- Object Security for Constrained Nodes?

Re-chartering CoRE

What do we want to do now that we're grown up?

Zach Shelby

CoRE WG @ IETF-86 Orlando

What is the scope?

- So we have a protocol and Web Linking, what next?
- Keep with what we know best…
  1. Enhancements around CoAP
  2. New transports for CoAP
  3. New representations for Web Links
  4. General application layer solutions

Enhancements around CoAP

- Advanced congestion control

New Transports for CoAP

- Infrastructure for CoAP over foo
  - http://tools.ietf.org/id/draft-silverajan-core-coap-alternative-transports-01.txt
- CoAP over SMS
  - http://tools.ietf.org/id/draft-becker-core-coap-sms-gprs-03.txt
  - OMA Lightweight
- CoAP over TCP?
  - http://tools.ietf.org/id/draft-li-core-coap-payload-length-option-01.txt

New support for Web Links

- JSON Web Links
  - http://tools.ietf.org/id/draft-bormann-core-links-json-02.txt

General application layer solutions

- Resource Directory
  - http://tools.ietf.org/id/draft-shelby-core-resource-directory-05.txt
  - OMA Lightweight