Constrained RESTful Environments WG (core)

Chairs:

Jaime Jiménez <jaime.jimenez@ericsson.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 8179 and its updates**

üBlue sheets
üScribe(s)

Note Well

Any submission to the IETF intended by the Contributor for publication as all or part of an IETF Internet-Draft or RFC and any statement made within the context of an IETF activity is considered an "IETF Contribution". Such statements include oral statements in IETF sessions, as well as written and electronic communications made at any time or place, which are addressed to:

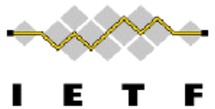
- The IETF plenary session
- The IESG, or any member thereof on behalf of the IESG
- Any IETF mailing list, including the IETF list itself, any working group or design team list, or any other list functioning under IETF auspices
- Any IETF working group or portion thereof
- Any Birds of a Feather (BOF) session
- The IAB or any member thereof on behalf of the IAB
- The RFC Editor or the Internet-Drafts function

All IETF Contributions are subject to the rules of RFC 5378 and RFC 8179.

Statements made outside of an IETF session, mailing list or other function, that are clearly not intended to be input to an IETF activity, group or function, are not IETF Contributions in the context of this notice. Please consult RFC 5378 and RFC 8179 for details.

A participant in any IETF activity is deemed to accept all IETF rules of process, as documented in Best Current Practices RFCs and IESG Statements.

A participant in any IETF activity acknowledges that written, audio and video records of meetings may be made and may be available to the public.



http://www.ietf.org/about/note-well.html

Agenda Bashing

All times are in time-warped WET (UTC+00:00)

Monday (120 min)

- 13:30–13:40 Intro, Agenda, Status
- 13:40–13:50 Post-WGLC: Links-JSON (chairs)
- 13:50–14:20 Post-WGLC: OSCORE (GS)
- 14:20–14:45 Post-WGLC: SenML (AK)
- 14:45–15:15 Up for WGLC soon: RD/DNS-SD (CA)
- 15:15–15:30 Up for WGLC soon: COMI (AP)

All times are in time-warped WET (UTC+00:00)

Tuesday (150 min)

- 09:30–09:35 Intro, Agenda
- 09:35–10:00 Post-WGLC: CoCoA (CG)
- 10:00–10:15 Getting ready: ERT (CA)
- 10:15–10:25 Getting ready: OSCORE-Group (MT)
- 10:25–10:40 New response codes (AK)
- 10:40–10:55 Pending for EST (PV)
- 10:55–11:05 Pubsub (MK)
- 11:05–11:15 Dynlink/Interfaces (BS)
- 11:15–11:25 Negotiation, AT (BS)
- 11:25–11:35 dev URN (JA)
- 11:35–12:00 Flextime: OPC/UA (CP), Time scale (LT), …

Draft-ietf-coap-tcp-tls
→ RFC 8323

Published 2018-02-15

Supporting: RFC 8307 (2018-01-03)

Advertisements

- T2TRG Coexistence (see draft-feeney-t2trg-inter-network-01): Mon 17:30..18:00 Waterloo
- 6TiSCH stateless-proxy option (in draft-ietf-6tisch-minimal-security-05): Wed 13:30..15:00 Viscount
- DNSSD: Thu 09:30..12:00 Buckingham

RIOT Summit

September 13 - 14, 2018

Meet in Amsterdam!

We already support the Summit!

RIPE NCC
RIPE NETWORK COORDINATION CENTRE

wolfSSL

Get involved too!

Call for Sponsors & Contributions

https://summit.riot-os.org/

All times are in time-warped WET (UTC+00:00)

Monday (120 min)

- 13:30–13:40 Intro, Agenda, Status
- 13:40–13:50 Post-WGLC: Links-JSON (chairs)
- 13:50–14:20 Post-WGLC: OSCORE (GS)
- 14:20–14:45 Post-WGLC: SenML (AK)
- 14:45–15:15 Up for WGLC soon: RD/DNS-SD (CA)
- 15:15–15:30 Up for WGLC soon: COMI (AP)

draft-ietf-core-links-json: Status

- Started Feb 2012 as a JSON version of 6690-to-be
  - Avoid the need for another parser
- Added CBOR variants mid-2015
- Focus: roundtrippable with RFC 6690
  - Inherit limitations of RFC 6690 (e.g., percent-encoding)
- Submitted to IESG on 2017-04-02
  - Lots of feedback
  - Related concepts in OCF spec
- Proposed Re-focus:
  - Still cover all of RFC 6690
  - Don’t inherit the limitations

Web Linking: RFC 5988 vs. RFC 8288

- RFC 6690 was based on RFC 5988
- Has since been updated to RFC 8288
  - More conscious use of ABNF
  - Clearer approach to Unicode and language tags
  - Clarifies role of serialization (of which RFC 6690 is one)
- RFC 6690 *not* updated to RFC 8288

- Links-json should use RFC 8288 as a base

Language tags

- RFC 5988 (and this 8288) defines “starred” attributes
  - Encoding Unicode content, language tag
- RFC 6690 supports “title*”, but doesn’t do much with that
- JSON/CBOR should not be concerned with weird encoding issues
- Language tags are useful for human readable values
- So: do support them, but get rid of the “*” hack:

  {“href”: “…”, “rel”: “…”,
  “title”: {“de_AT”: “Übergrößenträger”}}

Is this the right way forward?

- Rebase on RFC 8288
  - Clean up “title*” etc.
- Explain how RFC 6690 documents become Links-json documents
- Otherwise, keep Links-json generally applicable and free of RFC 6690 idiosyncrasies

- Do not change the mandate that “/.well-known/core” is RFC 6690 link-format (!?)

All times are in time-warped WET (UTC+00:00)

Monday (120 min)

- 13:30–13:40 Intro, Agenda, Status
- 13:40–13:50 Post-WGLC: Links-JSON (chairs)
- 13:50–14:20 Post-WGLC: OSCORE (GS)
- 14:20–14:45 Post-WGLC: SenML (AK)
- 14:45–15:15 Up for WGLC soon: RD/DNS-SD (CA)
- 15:15–15:30 Up for WGLC soon: COMI (AP)

OSCORE

draft-ietf-core-object-security-11

Göran Selander, Ericsson
John Mattsson, Ericsson
Francesca Palombini, Ericsson
Ludwig Seitz, RISE SICS

IETF 101, CoRE WG, London, Mar 19, 2018

Status (v-11)

- Several implementations
  - Java (Californium): https://bitbucket.org/lseitz/oscoap_californium
  - C (Contiki, Erbium): https://github.com/Gunzter/contiki-oscoap
  - Python (aiocoap): https://github.com/chrysn/aiocoap
  - C# (CoAP-CSharp): https://github.com/Com-AugustCellars/CoAP-CSharp
  - Python (CoAP for openwsn): https://github.com/openwsn-berkeley/coap
  - C (openwsn-fw): https://github.com/openwsn-berkeley/openwsn-fw
  - Java (Californium, v-03) https://github.com/lukadschaak/oscore

- Several interops done
  - Spec and reports: https://github.com/EricssonResearch/OSCOAP

Status (v-11)

- IETF Last Call ended: IESG evaluation
- Some post-Last-Call reviews
- Up-to-date handling of review comments on the wiki: https://github.com/core-wg/oscoap/wiki
- All but a few specific review comments addressed.

Review Comments

- “The document needs a security analysis section”
- "implications of modifications of unprotected fields"
- Proposal: Add an appendix describing the security properties of the protocol:
  - Assumptions on intermediaries
  - Protected header fields, security guarantees
  - Unprotected fields, consequences

19

Review Comments

- "Nonce construction: Why is Sender ID included in the nonce?"
- Answer: Designed for supporting notifications and interchange of client and server roles
- Proposal: Prove (key, nonce) uniqueness in the new appendix

20

Review Comments

- “But this design actively works against any involvement of intermediaries.”

- Answer: The design supports intermediaries e.g. performing forwarding and translation
- In the general case, proxies can read but not modify without being detected.

- Proposal: Clarify this in the new appendix.

Review Comments

- “neglecting to address important and difficult parts of the problem like key exchange”

- Answer: Key establishment is addressed.
  - The ACE/OAuth 2.0 framework may be used.
  - Some IoT deployments require PSK.

- Key exchange for OSCORE is discussed in ACE since IETF#95.

Review Comments: HTTP 1(2)

- “This protocol abuses HTTP by tunneling over it”
- Answer: Yes. This was requested.

- "Missing [A]BNF"
- Answer: Agreed, included

- "Does the COAP-HTTP gateway understand the significance of the new header field and insert the media type when translating? "
- Answer: Yes

Review Comments: HTTP 2(2)

- "A new media type is defined, but I don't see any mention of a codepoint for use with COAP"
- Proposal: Not needed for this draft, but will include that for other potential use

- "What if the request is redirected by a server that doesn't understand OSCORE?"
- Question for WG: shall we support HTTP redirects?

24

- Question for WG: Rename HTTP header field:
- 'Object-Security' → 'CoAP-Object-Security'

Reviews Comments: Summary Proposal

- Clarifications of the points brought up
- Editorials
- New appendix:
  - D. Overview of Security Properties
    - D.1. Supporting Proxy Operations
    - D.2. Protected Message Fields
    - D.3. Uniqueness of (key, nonce)
    - D.4. Unprotected Message Fields

- Details on the CoRE WG Github Commits

All times are in time-warped WET (UTC+00:00)

Monday (120 min)

- 13:30–13:40 Intro, Agenda, Status
- 13:40–13:50 Post-WGLC: Links-JSON (chairs)
- 13:50–14:20 Post-WGLC: OSCORE (GS)
- 14:20–14:45 Post-WGLC: SenML (AK)
- 14:45–15:15 Up for WGLC soon: RD/DNS-SD (CA)
- 15:15–15:30 Up for WGLC soon: COMI (AP)

Media Types for Sensor Measurement Lists (SenML)

IETF 101, London

draft-ietf-core-senml-13

Ari Keränen

Status

- Done!
  - IETF LC ongoing
  - IESG Telechat April 19th
- Since -12: "+exi" -> "-exi" & editorial fixes
- Still: could add expert guidance clarification for new values: must have "Value" in the long name

Early assignments

- Suggested CoAP Content-Format IDs
  - XML IDs in 2-byte range

Media type	ID
application/senml+json	110
application/sensml+json	111
application/senml+cbor	112
application/sensml+cbor	113
application/senml-exi	114
application/sensml-exi	115
application/senml+xml	310
application/sensml+xml	311

Early assignments

- How about SenML Fields?

Media types for FETCH & PATCH with SenML

IETF 101, London

draft-keranen-senml-fetch-00

Ari Keränen & Mojan Mohajer

SenML IPSO SO example

```
[ {"bn":"2001:db8::2/3306/0/",
   "n":"5850", "vb":true},
  {"n":"5851", "v":42},
  {"n":"5852", "v":1200},
  {"n":"5750", "vs":"Ceiling light"} ]
```

SenML IPSO SO example

```
[ {"bn":"2001:db8::2/3306/0/",
   "n":"5850", "vb":true},
  {"n":"5851", "v":42},
  {"n":"5852", "v":1200},
  {"n":"5750", "vs":"Ceiling light"} ]
```

- Want to retrieve/change only 5850 **and** 5851
- And want to avoid exchanging full representations or doing multiple requests

CoAP FETCH / PATCH (RFC 8132)

- CoAP methods, FETCH, PATCH, and iPATCH, which are used to access and update parts of a resource
- Needs payload format; dependent on the resource representation format

SenML FETCH format

- Modeled after SenML JSON format: simple parsing on constrained things with SenML support
- Just indicate names, and potentially times, of the SenML records to fetch

```
[ {"bn":"2001:db8::2/3306/0/", "n":"5850"},
  {"n":"5851"} ]
```

SenML PATCH format

- Same as FETCH format, but with the value(s) to set
  - Essentially a subset of the JSON Merge Patch format

```
[ {"bn":"2001:db8::2/3306/0/", "n":"5850", "vb":false},
  {"n":"5851", "v":10} ]
```

Wild cards

- Optimization for selecting many SenML Records with one FETCH/PATCH Record
- Useful with large amounts of SenML Records (e.g., many IPSO objects on a device)
  - "Get all temperature sensor values"
  - "Dim all lights to 10%"

Proposed format

- New SenML Field "ff" ("fetch filter")
  - Used instead of the name field and concatenated to base name like the name field
  - Contains wild card characters "*"
  - Matched to SenML Record Names
- Wild card matches all characters until next "/" or ":"

[{"bn":"2001:db8::2/", "ff":"3306/0/58*"}]

(This matches all records in the example except "3306/0/5750")

(Wild Card) Considerations

- Need something **simple** now: constrained devices
  - Wild card **seemed** most suitable
- Using new Field(s) enables easy extensibility
  - Alternative: re-purpose "n" and "bn" fields
- Should wild card support be MUST?
  - How to indicate "not supporting wild cards"? Now suggesting "4.00 Bad Request" but doesn't seem right
- Regular expressions? New field probably

- PATCH operation codes needed (append, delete, …)?
- Can just re-use SenML content format IDs?
- Interest in CoRE WG to work on this?

All times are in time-warped WET (UTC+00:00)

Monday (120 min)

- 13:30–13:40 Intro, Agenda, Status
- 13:40–13:50 Post-WGLC: Links-JSON (chairs)
- 13:50–14:20 Post-WGLC: OSCORE (GS)
- 14:20–14:45 Post-WGLC: SenML (AK)
- 14:45–15:15 Up for WGLC soon: RD/DNS-SD (CA)
- 15:15–15:30 Up for WGLC soon: COMI (AP)

Resource Directory

draft-ietf-core-resource-directory

draft-ietf-core-rd-dns-sd

draft-amsuess-rd-replication

Zach Shelby, Michael Koster, Carsten Bormann,
Peter van der Stok, *Christian Amsüss*
Kerry Lynn

2018-03-19

Status

pretty much ready

Issue tracker / pull requests

107 down, ~~1 to go~~ 2 to go

plug test upcoming

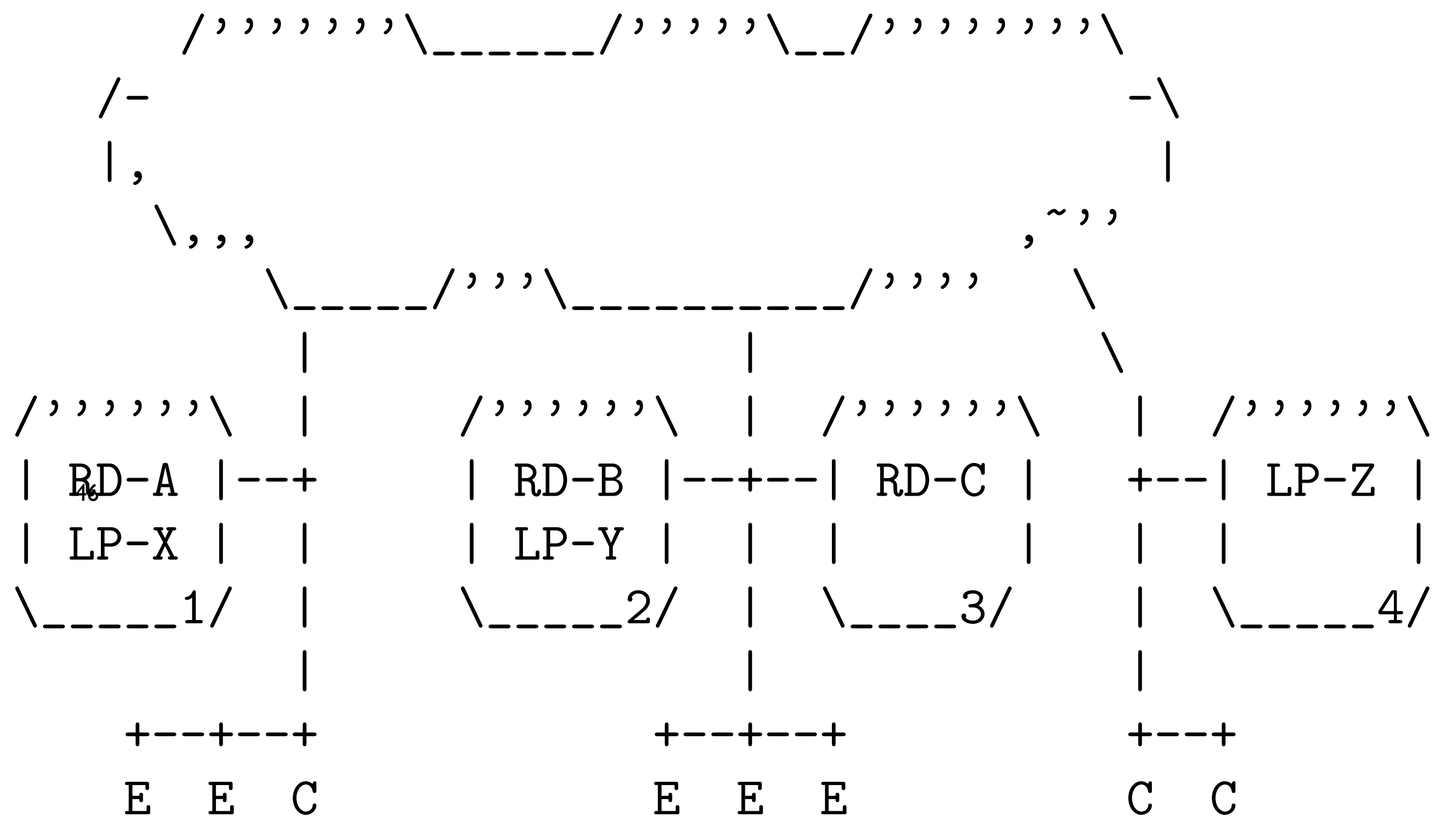
contact me: c@amsuess.com

Changes since -12

- Cleanup and clarification
  - Clarified observation behavior
  - Refer to t2trg-rel-impl for server metadata / versioning
  - Reduced the significance of domains (removed from figure 2)
- Added "all resource directory" nodes MC address
- Resolve RFC6690-vs-8288 resolution ambiguities
  - Require registered links not to be relative when using anchor
  - Return absolute URIs in resource lookup
- Work with replication without really changing the RD[45]
  - Multiple RDs can be found, and can have absolute addresses
  - Endpoints from other RDs can be members of a group

rd-replication

- Different registration addresses
- Different lookup addresses
- Eventually consistent results



-01: updated with introduction

hooks into RD extension points

Next steps for `resource-directory`

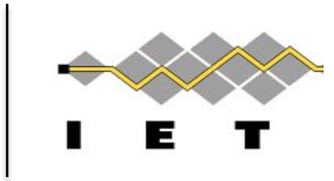
reviews

plug test

All times are in time-warped WET (UTC+00:00)

Monday (120 min)

- 13:30–13:40 Intro, Agenda, Status
- 13:40–13:50 Post-WGLC: Links-JSON (chairs)
- 13:50–14:20 Post-WGLC: OSCORE (GS)
- 14:20–14:45 Post-WGLC: SenML (AK)
- 14:45–15:15 Up for WGLC soon: RD/DNS-SD (CA)
- 15:15–15:30 Up for WGLC soon: COMI (AP)



CoMI – update

draft-ietf-core-comi-01

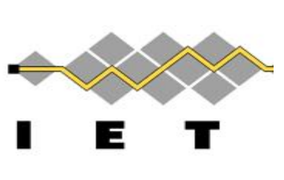
Andy Bierman

Michel Veillette

Peter van der Stok

Alexander Pelov <a@ackl.io>

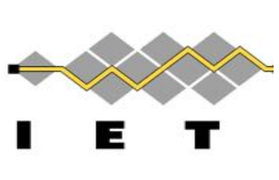
Draft status



Actions from last time:
- Official Hackathon @ I
- Improve interop (simp
- SID registry

Draft	Version	Status	
ietf-core-yang-cbor	6	Stable since IETF 97	**Ready for WGLC**
ietf-core-sid	3	Stable since IETF 98	Need to add YANG Template WGLC afterwards (April)
ietf-core-comi	2	Stable since IETF 99	Minor editions/check − need to check YANG Template, YANG attach, NMDA YANG Push is OK
veillette-core-yang-library	2	Stable since IETF 98	CoMI model introspection In scope for Core? Normative reference in CoMI

Implementations
CoMI with YANG-CBOR



Existing implementations

- GoLang: server + client
- C: server + client
- 2 more partial proprietary implementations

Virtual interop @ Hackathon IETF100

- FETCH with ietf-system

Hackathon 101 – Semantic interoperability

- YANG -> Thing Description (W3C)
- CoMI bindings to TD
  - GET is a MUST

CoMI test on F-Interop

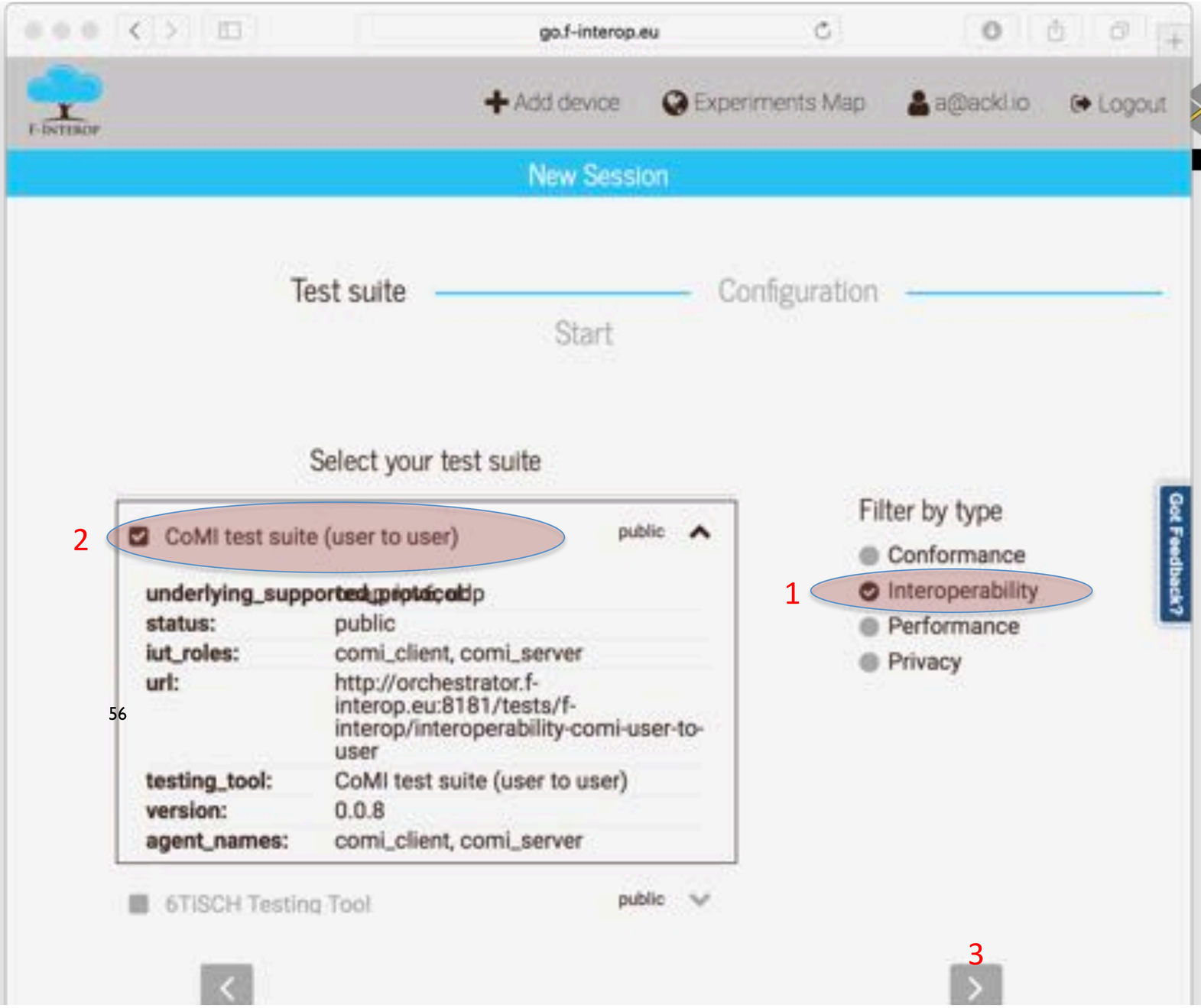
F-interop

- Environment for executing an online and remote interoperability test session (VPN-lil setup)
- Cordinate the interop test
- Sniff the traffic (generate PCAP files records)
- Dissect the messages (include Wireshark-like view)
- Analyze the exchanged traffic (automatically issue PASS/FAIL/INCONCLUSIVE verdicts

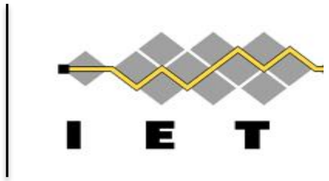
F-interop reference implementation of CoMI published

- CoMI Server
- CoMI Client
- GET, FETCH, PUT, IPATCH and DELETE

go.f-interop.eu
F-INTEROP
Login
Email address
Password
Log In
Forgot password?
Don't have an account yet? Sign Up
Got Feedback?


go.f-interop.eu
F-INTEROP
Add device
Experiments Map
a@ackl.io
Logout
New Session
Test suite
Configuration
Start
Select your test suite
2
CoMI test suite (user to user)
public
underlying_supported_protocol: p
status: public
iut_roles: comi_client, comi_server
url: http://orchestrator.f-interop.eu:8181/tests/f-interop/interoperability-comi-user-to-user
testing_tool: CoMI test suite (user to user)
version: 0.0.8
agent_names: comi_client, comi_server
6TiSCH Testing Tool
public
Filter by type
1
Conformance
Interoperability
Performance
Privacy
Got Feedback?
3

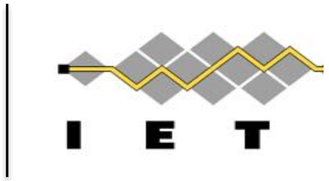
Test file



interop@2017-12-12.yang

```
le comi-interop {


ntainer interface {
  leaf ip-address {
    type string;
  }

  leaf name {
    type string;
  }

  leaf throughput {
    type int64;
  }
```

Test file



-interop@2017-12-12.yang

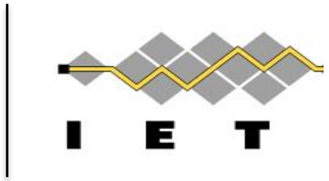
```
le comi-interop {


ntainer interface {
  leaf ip-address {
    type string;
  }

  leaf name {
    type string;
  }

  leaf throughput {
    type int64;
  }
```

comi-interop@2017-12-12.sid

```
{
  "namespace": "data",
  "identifier": "/comi-interop:interface",
  "sid": 70001
},
{
  "namespace": "data",
  "identifier": "/comi-interop:interface/ip-address",
  "sid": 70002
},
{
  "namespace": "data",
  "identifier": "/comi-interop:interface/name",
  "sid": 70003
},
{
  "namespace": "data",
  "identifier": "/comi-interop:interface/throughput",
  "sid": 70004
}
```

EXPERIMENTAL RANGE (see draft-ietf-core-sid)

Test file



-interop@2017-12-12.yang

```
le comi-interop {


ntainer interface {
  leaf ip-address {
    type string;
  }

  leaf name {
    type string;
  }

  leaf throughput {
    type int64;
  }
```

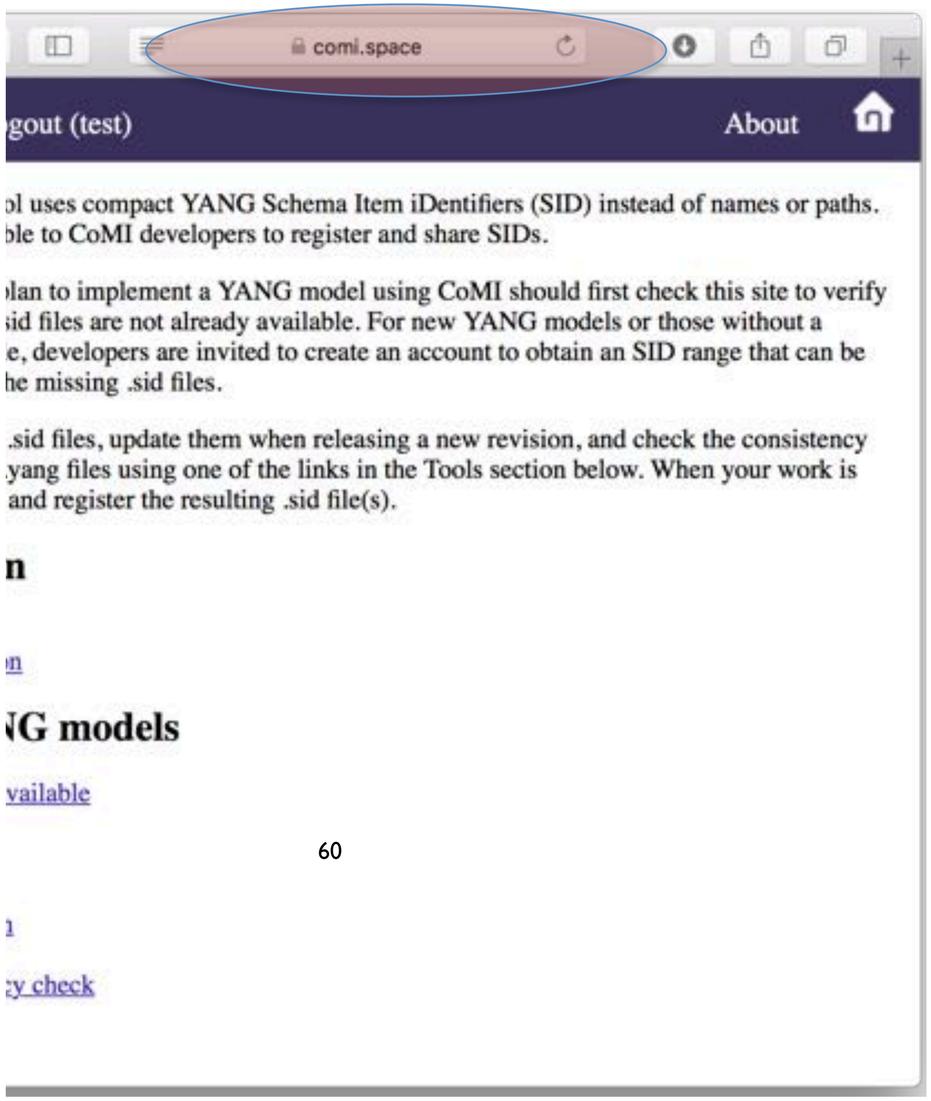
CoAP GET /c/RFy

comi-interop@2017-12-12.sid

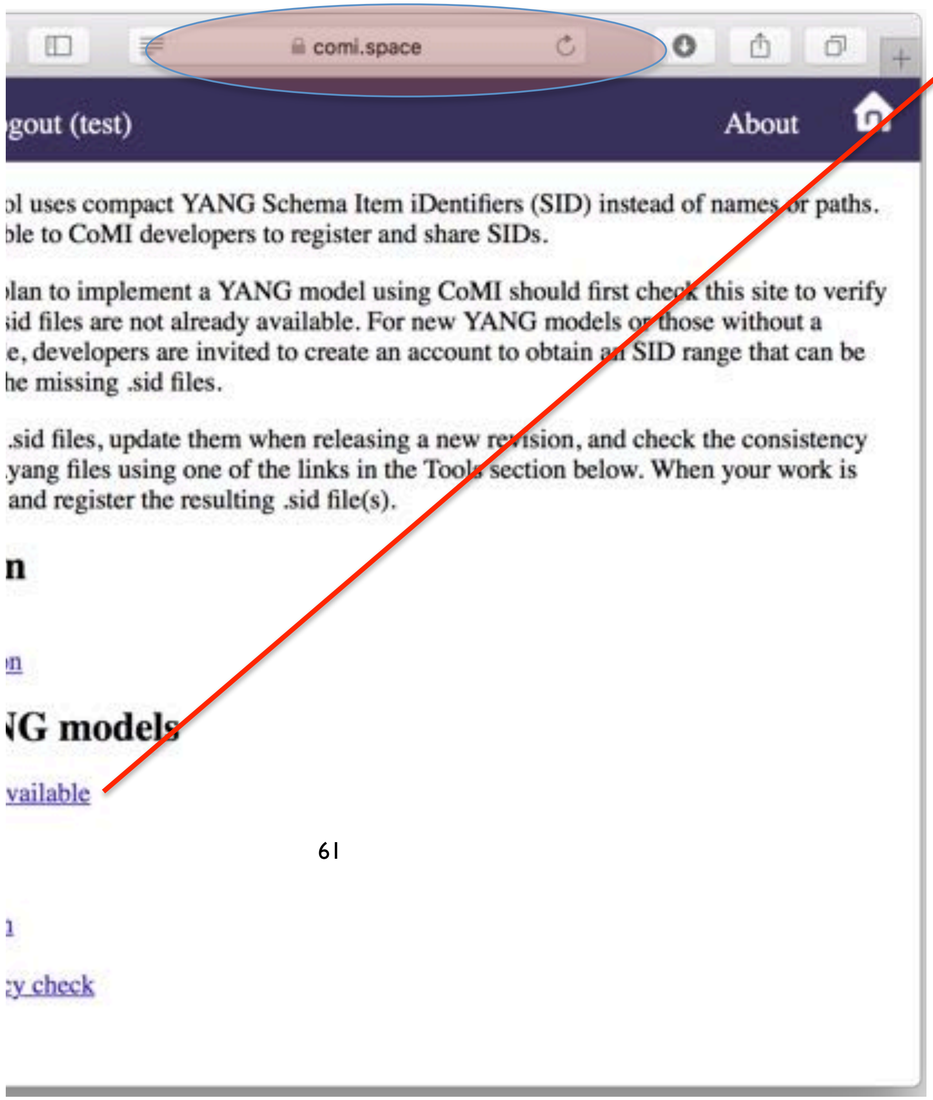
```
{
  "namespace": "data",
  "identifier": "/comi-interop:interface",
  "sid": 70001
},
{
  "namespace": "data",
  "identifier": "/comi-interop:interface/ip-address",
  "sid": 70002
},
{
  "namespace": "data",
  "identifier": "/comi-interop:interface/name",
  "sid": 70003
},
{
  "namespace": "data",
  "identifier": "/comi-interop:interface/throughput",
  "sid": 70004
}
```

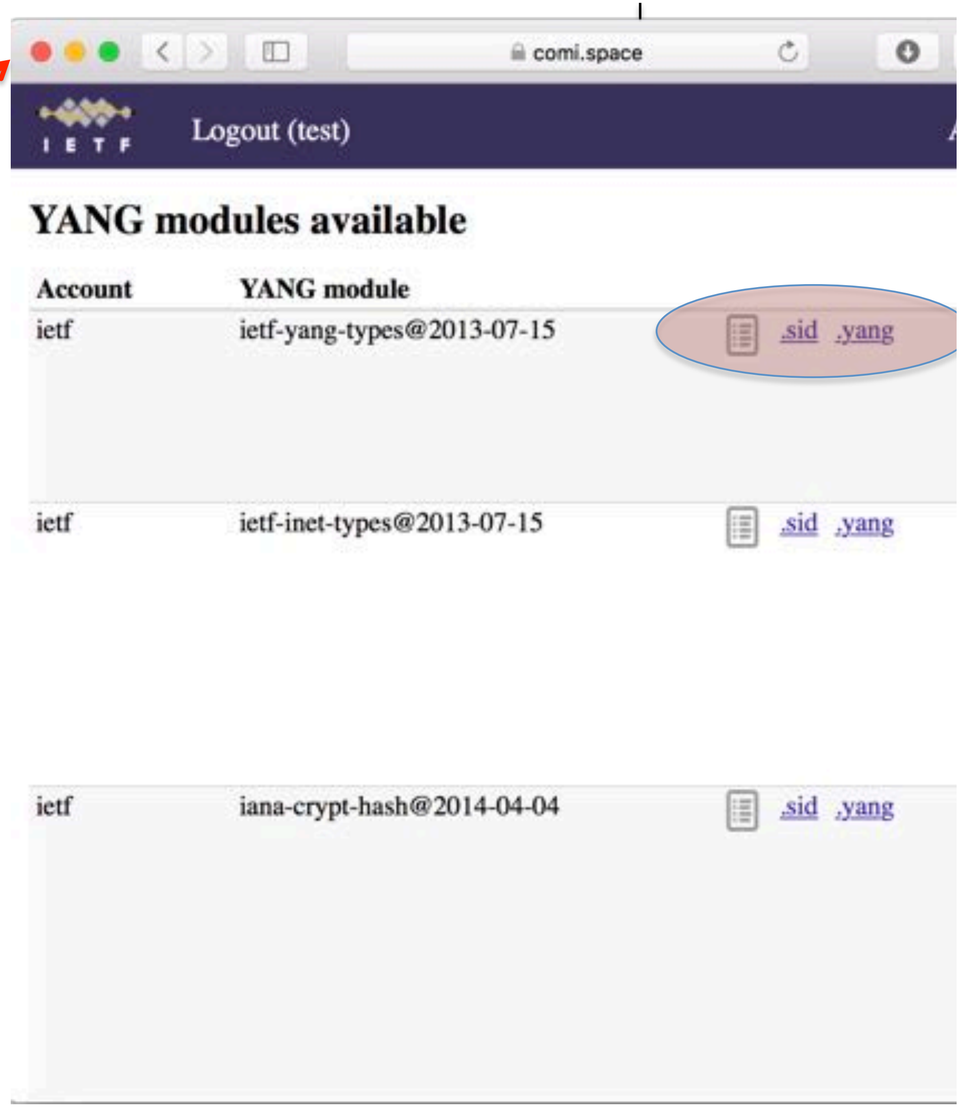
EXPERIMENTAL RANGE (see draft-ietf-core-sid)

SID registry

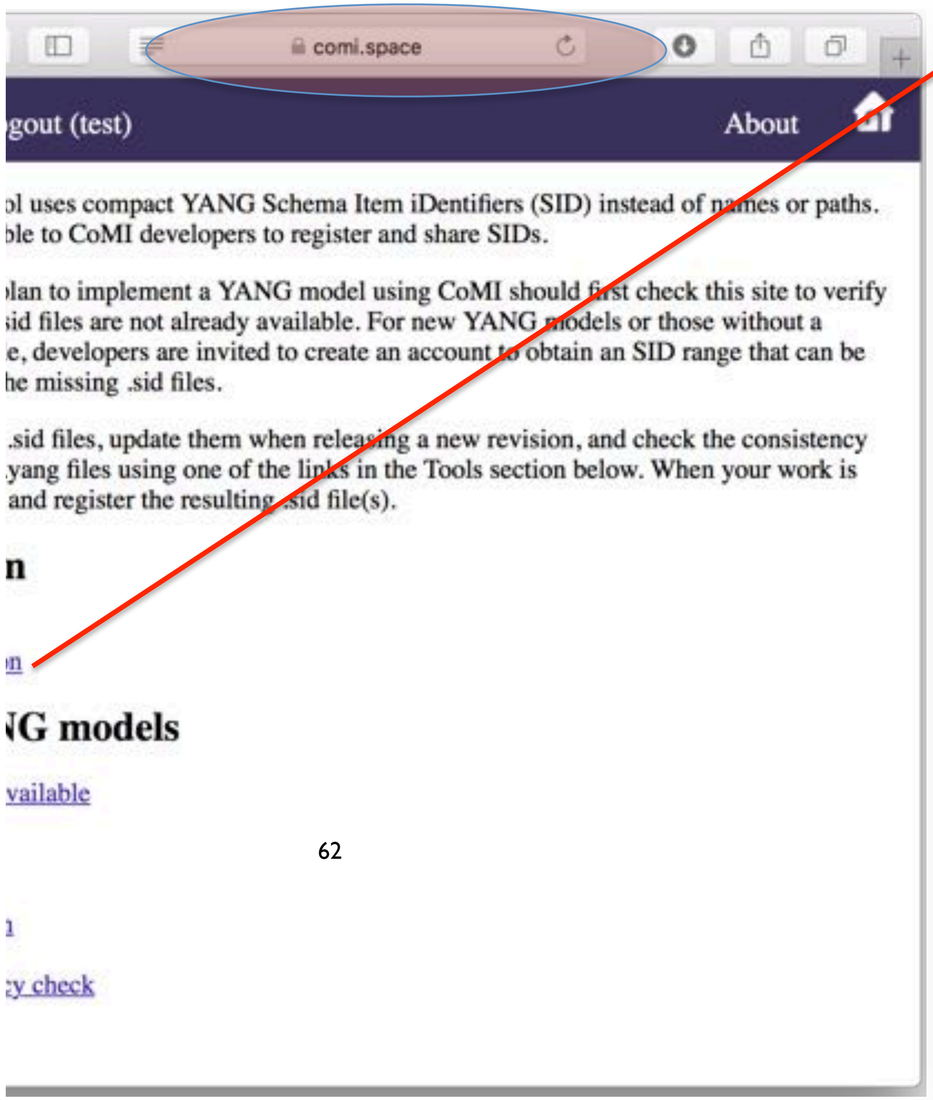


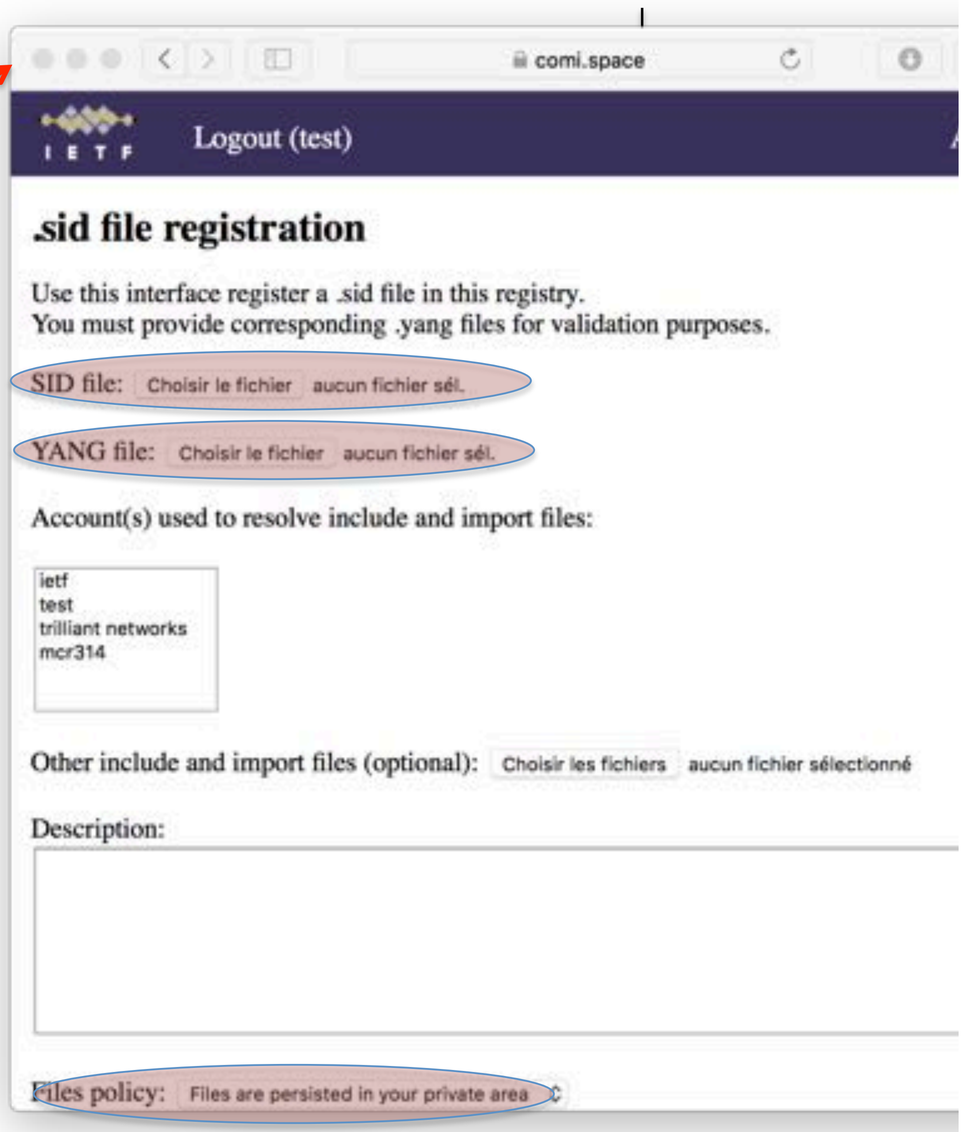
SID registry



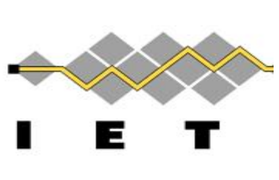


SID registry





Next steps



etf-core-yang-cbor

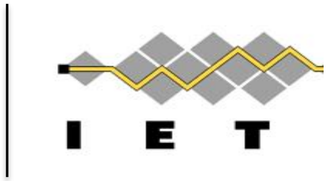
- **Start WGLC?**

etf-core-sid and ietf-core-comi

- **Shepherd, reviewers?**
- TODO Check all OK for YANG Template, YANG attach, NMDA
- WGLC in April

\doption of veillette-core-yang-library as WG item?

n the mean time − do the Interop



Thanks!

All times are in time-warped WET (UTC+00:00)

Tuesday (150 min) ➔ Monday

- 09:30–09:35 Intro, Agenda
- 09:35–10:00 Post-WGLC: CoCoA (CG)
- 10:00–10:15 Getting ready: ERT (CA)
- 10:15–10:25 Getting ready: OSCORE-Group (MT)
- 10:25–10:40 New response codes (AK)
- 10:40–10:55 Pending for EST (PV)
- 10:55–11:05 Pubsub (MK)
- 11:05–11:15 Dynlink/Interfaces (BS)
- 11:15–11:25 Negotiation, AT (BS)
- 11:25–11:35 dev URN (JA)
- 11:35–12:00 Flextime: OPC/UA (CP), Time scale (LT), …

Secure group communication for CoAP

draft-ietf-core-oscore-groupcomm-01

Marco Tiloca, RISE SICS
Göran Selander, Ericsson
Francesca Palombini, Ericsson
Jiye Park, Universität Duisburg-Essen

IETF 101, CoRE WG, London, March 20th, 2018

Updates from -00 (1/2)

› Major updates and restructuring to address reviews
  – Thanks to Esko Dijk and Peter van der Stok

› Section 1.1 – Terminology
  – Added definition of group as “security group”
  – Not to be confused with “network group” or “application group”

› Section 2 – Security Context
  – Clarified establishment/derivation of contexts
  – Added table for additional elements wrt OSCORE

Updates from -00 (2/2)

- Section 3 – COSE Object
  - Examples or request and response (before and after compression)
  - CounterSignature0 is used rather than CounterSignature
  - ‘external_aad’ includes also the signature algorithm
  - ‘external_aad’ does not include the Group Identifier (Gid) any more

- Section 6 – NEW
  - List of responsibilities of the Group Manager

- Appendices
  - Appendix A: assumptions and security objectives (former section)
  - Appendix B: additional details on considered use cases
  - Appendix C: added actual example of Gid format (prefix + epoch)
  - Appendix D: join description aligned with *draft-palombini-ace-key-groupcomm*

Points for discussion (1/2)

› Independence of Security Group from IP addresses
  – Requests may be multicast or unicast (e.g. selective retransmissions)
  – Current context retrieval based on Gid and multicast IP address
  – Change to use only the Gid as kid context for context retrieval ?

› Fixed part of the Gid
  – Currently random and large enough to avoid global collisions
  – Change to neglect randomness and large size ?
  – Tie-breaker can be trying the keying material from multiple contexts

Points for discussion (2/2)

- Current terminology explicitly points at multicast
  - Replace “Multicaster” with “Sender” ?
  - Replace “(Pure) Listener” with “(Pure) Recipient”?
  - This would simplify request/assignment of roles upon joining

- Current description of the join process
  - Appendix D.1: exchanged information
  - Appendix D.2: provisioning/retrieval of public keys
  - Appendix D.3: pointer to the ACE-based approach
  - What should be kept in this document?
  - Should we keep a general description in case ACE is not used?

Implementation

- OSRAM Innovation
  - Developed in C
  - MediaTek LinkIt Smart 7688
  - Aligned with individual submission at IETF99
- Proof-of-concept for Contiki OS
  - Wismote (MSP430; TI CC2520)
  - SmartRF (MSP430; TI CC2538)
  - Aligned with individual submission at IETF99
  - https://github.com/tdrlab/mcast[7]
- Next steps
  - Move forward to interoperability tests
  - Is it feasible already at IETF102?

Related activity

› *draft-tiloca-ace-oscoap-joining*
  – Referred by Appendix D.3

› Join an OSCORE group using the ACE framework
  – Joining node → Client
  – Group Manager → Resource Server
  – Message formats aligned with *draft-palombini-ace-key-groupcomm*

72

› Leverage protocol-specific profiles of ACE
  – CoAP-DTLS profile *draft-ietf-ace-dtls-authorize*
  – OSCORE profile *draft-ietf-ace-oscore-profile*

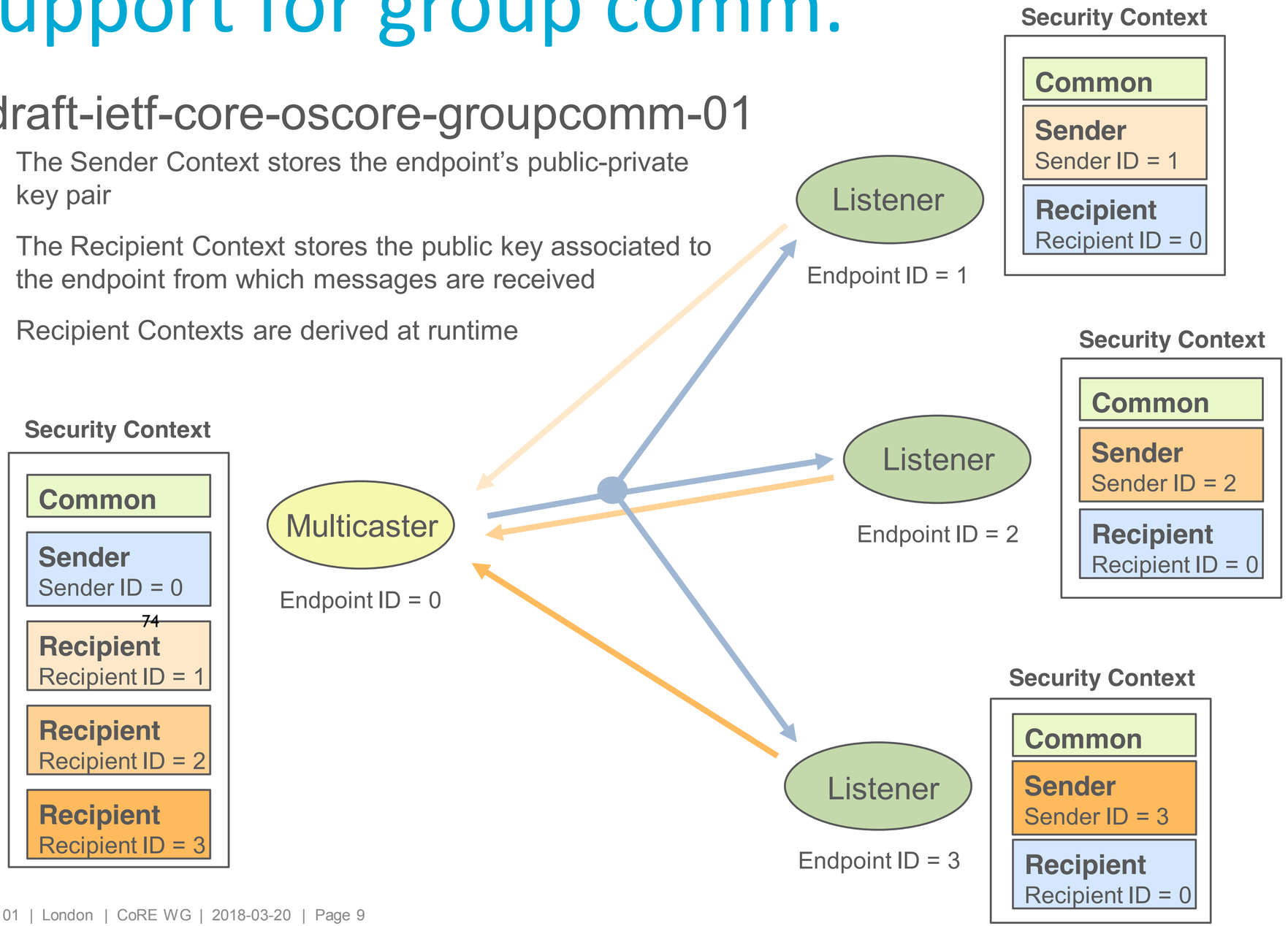
Thank you!

Comments/questions?

https://github.com/core-wg/oscore-groupcomm

Support for group comm.

- draft-ietf-core-oscore-groupcomm-01
  - The Sender Context stores the endpoint's public-private key pair
  - The Recipient Context stores the public key associated to the endpoint from which messages are received
  - Recipient Contexts are derived at runtime



All times are in time-warped WET (UTC+00:00)

Tuesday (150 min) ➔ Monday

- 09:30–09:35 Intro, Agenda
- 09:35–10:00 Post-WGLC: CoCoA (CG)
- 10:00–10:15 Getting ready: ERT (CA)
- 10:15–10:25 Getting ready: OSCORE-Group (MT)
- 10:25–10:40 New response codes (AK)
- 10:40–10:55 Pending for EST (PV)
- 10:55–11:05 Pubsub (MK)
- 11:05–11:15 Dynlink/Interfaces (BS)
- 11:15–11:25 Negotiation, AT (BS)
- 11:25–11:35 dev URN (JA)
- 11:35–12:00 Flextime: OPC/UA (CP), Time scale (LT), …

Too Many Requests Response Code for CoAP

IETF 101, London

draft-keranen-core-too-many-reqs-00

Ari Keränen

Background

- CoAP client can cause overload in server with too frequent requests
- How can server tell client to back off
- HTTP error code 429 “Too many requests”
- Proposal: register 4.29 for CoAP
  - With MaxAge to indicate when it’s OK to request again
- Originally part of CoAP Pub/sub Broker draft; also OCF interest

What requests are OK?

- Current text: Client “SHOULD NOT send the same request to the server before the time indicated in the Max-Age option has passed”
- Other requests? Should server be able to give guidance what else is (not) OK during this time?
  - Example: GET instead of PUBLISH
- Sounds like a generic problem worth a generic solution; probably out of scope for this draft

Next steps

- Bundle with other non-controversial Response Codes?
- WG item?

- **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 8179 and its updates**

üBlue sheets
üScribe(s)

Note Well

Any submission to the IETF intended by the Contributor for publication as all or part of an IETF Internet-Draft or RFC and any statement made within the context of an IETF activity is considered an "IETF Contribution". Such statements include oral statements in IETF sessions, as well as written and electronic communications made at any time or place, which are addressed to:

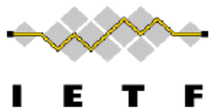
- The IETF plenary session
- The IESG, or any member thereof on behalf of the IESG
- Any IETF mailing list, including the IETF list itself, any working group or design team list, or any other list functioning under IETF auspices
- Any IETF working group or portion thereof
- Any Birds of a Feather (BOF) session
- The IAB or any member thereof on behalf of the IAB
- The RFC Editor or the Internet-Drafts function

All IETF Contributions are subject to the rules of RFC 5378 and RFC 8179.

Statements made outside of an IETF session, mailing list or other function, that are clearly not intended to be input to an IETF activity, group or function, are not IETF Contributions in the context of this notice. Please consult RFC 5378 and RFC 8179 for details.

A participant in any IETF activity is deemed to accept all IETF rules of process, as documented in Best Current Practices RFCs and IESG Statements.

A participant in any IETF activity acknowledges that written, audio and video records of meetings may be made and may be available to the public.



http://www.ietf.org/about/note-well.html

All times are in time-warped WET (UTC+00:00)

Tuesday (150 min)

- 09:30–09:35 Intro, Agenda
- 09:35–10:00 Post-WGLC: CoCoA (CG)
- 10:00–10:15 Getting ready: ERT (CA)
- 10:15–10:25 Getting ready: OSCORE-Group (MT)
- 10:25–10:40 New response codes (AK)
- 10:40–10:55 Pending for EST (PV)
- 10:55–11:05 Pubsub (MK)
- 11:05–11:15 Dynlink/Interfaces (BS)
- 11:15–11:25 Negotiation, AT (BS)
- 11:25–11:35 dev URN (JA)
- 11:35–12:00 Flextime: OPC/UA (CP), Time scale (LT), …

All times are in time-warped WET (UTC+00:00)

Tuesday (150 min)

- 09:30–09:35 Intro, Agenda
- 09:35–10:00 Post-WGLC: CoCoA (CG)
- 10:00–10:10 dev URN (JA)
- 10:10–10:25 Getting ready: ERT (CA)
- ~~10:15–10:25 Getting ready: OSCORE-Group (MT)~~
- ~~10:25–10:40 New response codes (AK)~~
- 10:25–10:40 Pending for EST (PV)
- 10:40–10:50 Pubsub (MK)
- 10:50–11:00 Dynlink/Interfaces (BS)
- 11:00–11:10 Negotiation, AT (BS)
- 11:10–12:00 Flextime: OPC/UA (CP), Time scale (LT), …

All times are in time-warped WET (UTC+00:00)

Tuesday (150 min)

- 09:30–09:35 Intro, Agenda
- 09:35–10:00 Post-WGLC: CoCoA (CG)
- 10:00–10:10 dev URN (JA)
- 10:10–10:25 Getting ready: ERT (CA)
- ~~10:15–10:25 Getting ready: OSCORE-Group (MT)~~
- ~~10:25–10:40 New response codes (AK)~~
- 10:25–10:40 Pending for EST (PV)
- 10:40–10:50 Pubsub (MK)
- 10:50–11:00 Dynlink/Interfaces (BS)
- 11:00–11:10 Negotiation, AT (BS)
- 11:10–12:00 Flextime: OPC/UA (CP), Time scale (LT), …

All times are in time-warped WET (UTC+00:00)

Tuesday (150 min)

- 09:30–09:35 Intro, Agenda
- 09:35–10:00 Post-WGLC: CoCoA (CG)
- 10:00–10:10 dev URN (JA)
- 10:10–10:25 Getting ready: ERT (CA)
- ~~10:15–10:25 Getting ready: OSCORE-Group (MT)~~
- ~~10:25–10:40 New response codes (AK)~~
- 10:25–10:40 Pending for EST (PV)
- 10:40–10:50 Pubsub (MK)
- 10:50–11:00 Dynlink/Interfaces (BS)
- 11:00–11:10 Negotiation, AT (BS)
- 11:10–12:00 Flextime: OPC/UA (CP), Time scale (LT), …

CoAP Simple Congestion Control/Advanced (CoCoA)

draft-ietf-core-cocoa-03

Carsten Bormann – Universität Bremen TZI
August Betzler – Fundació i2Cat
Carles Gomez, Ilker Demirkol – Univ. Politècnica de Catalunya

Status

- WG state: “Submitted to IESG for publication”
- Last revision is -03
  - Mostly editorial updates
  - Addresses comments by:
    - Wesley Eddy (TSVART Early Review)
    - Mirja Kühlewind (Responsible AD)
- Next revision
  - Needs to address comments by:
    - Scott Bradner (OPSDIR Telechat Review)
    - Vincent Roca (SECDIR Review)
    - Christer Holmberg (Gen-ART Telechat Review)

Updates in -03 (I)

- Section 1
  - Paragraph previously in Section 5, now more general: overview on CoCoA
    - RTO based on (weak or strong) RTTs
    - Weak RTTs: reaction to congestion with a lower sending rate
    - For NONs, sending rate limited to 1/RTO
      - More conservative than RFC 7641 (Observe): 1/RTT

Updates in -03 (II)

- Section 3
  - Added details on scenarios where CoCoA has been found to perform well
    - Latencies: milliseconds to peaks of dozens of seconds
      - Comment from Jaime: which reference contributes to what within this range
    - Single-hop and multihop network topologies
    - Link technologies: IEEE 802.15.4, GPRS, UMTS, Wi-Fi
  - Added that CoCoA is also expected to work suitably across the general Internet

Updates in -03 (III)

- Section 4.2
  - Added that default weight values for strong and weak RTO estimators have been found to work well in evaluations (Appendix A)

- Section 4.2.1
  - Added an explicit note on VBF replacing RFC 6298 simple exponential backoff

Updates in -03 (IV)

- Section 4.3
  - State of RTO estimators for an endpoint
    - Should be kept long enough to avoid frequent returns to inappropriate initial values
    - For default parameters in CoAP, it is RECOMMENDED to keep it for at least 255 s
      - Was a “MUST” in -02
- Minor editorial updates throughout the document

Next revision (I)

- Scott Bradner's comment
  - The draft makes no reference to RFC 5033…
    - "Specifying New Congestion Control Algorithms"
  - … But we have taken RFC 5033 into account in the design of CoCoA

Next revision (II)

- RFC 5033 guidelines
  - 0. Differences with congestion control principles (RFC 2914)
    - CoCoA design considers such principles (preventing congestion collapse, fairness, optimizing performance)
  - 1. Impact on standard TCP, SCTP, DCCP
    - No negative impact
  - 2. Difficult environments
    - CoCoA has been designed for “difficult environments”
  - 3. Investigating a range of environments
    - Done (see slide 4)

Next revision (III)

- RFC 5033 guidelines
  - 4. Protection against congestion collapse
    - VBF of 1.5, 2 or 3 (always greater than 1)
  - 5. Fairness within the alternate cong. control mech.
    - High fairness measured (thanks to the VBF)
  - 6. Performance with misbehaving nodes
    - Considered. Weak estimator role
  - 7. Responses to sudden or transient events
    - CoCoA restores “normal” network state quickly
  - 8. Incremental deployment
    - CoCoA runs correctly in current CNNs and in CNN-cloud

Thanks!

Carsten Bormann – Universität Bremen TZI
cabo@tzi.org
August Betzler, Carles Gomez, Ilker Demirkol
Universitat Politècnica de Catalunya
carlesgo@entel.upc.edu

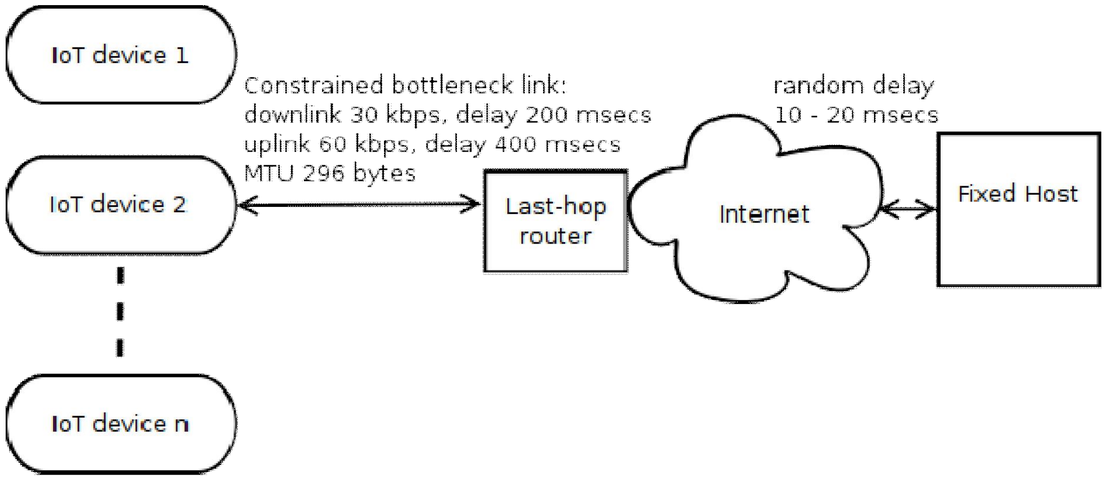
Experimental Results with Default CoAP, CoCoA and CoAP over TCP RTO Management & Congestion Control

Ilpo järvinen, Iivo Raitahila, Laura Pesola, Zhen Cao[§],
Markku Kojo

[96] Department of Computer Science
University of Helsinki
[§] Huawei

System under Study

- 1 to 400 IoT devices communicate with a fixed host over a constrained link
- Emulated wireless NB-IoT like network
- Varying router buffer sizes
  - 2500 B (recommended ~ BDP of the link)
  - 14100 B [97]
  - 28200 B
  - 1410000 B (“infinite”, extreme buffer bloat)



Transport & Congestion Control

- Client and server implemented using libcoap
- Default CoAP as implemented in libcoap (+some bugfixes)
- CoCoA implemented as per draft-ietf-core-cocoa-01 and draft-ietf-core-cocoa-03
- For Default CoAP and CoCoA
  - MAX RETRANSMIT = 20 (EXCHANGE_LIFETIME and MAX_TRANSMIT_WAIT adjusted accordingly)
  - Max RTO: 60 secs, 32 secs for CoCoA as specified
- Implemented CoAP over TCP as per draft-ietf-core-coap-tcp-tls-09
  - Only necessary features implemented
  - Run on top of Linux TCP
- Linux TCP (modified)
  - Use NewReno, Disabled: SACK, Cubic, Timestamps, F-RTO, CBI
  - Experimental features disabled: TCP RACK, TFO
  - Initial RTO: 2 secs
  - Delayed Ack timer: constant 200 msec
  - SYN and SYN/ACK retries: 40 and 41
  - Max RTO: 120 secs (Linux default)

Workloads

- Small request-response exchanges
- Request and response both fit in one CoAP message
- 1, 10, 50, 100, 200, 400 simultaneous client-server pairs (flows)
- Two types of clients:
  - **Continuous**: 50 successful request-reply exchanges
    - TCP connection for each client is pre-established and the three-way handshake is not included in the measurements
  - **Random**: emulates short-lived clients (Random clients)
    - 99% A short-lived random client sends 1-10 requests followed by another random client until 50 request-response pairs successfully exchanged
      - Retransmission timer reinitialized for each new random client
      - A new TCP connection opened for a new random client

Flow Completion Time (FCT) - 1 and 10 Clients

Table 1: Flow completion time (FCT) of **1 Continuous** Client (secs)

Buffer Size	CC algorithm	Min	10	25	Median	75	90	max
2500B	Default CoAP	33.003	33.003	33.003	33.003	33.003	33.003	33.003
2500B	CoCoA	33.002	33.003	33.003	33.003	33.003	33.003	33.003
2500B	CoAP over TCP	33.208	33.208	33.208	33.208	33.208	33.208	33.208

Table 2: Flow completion time (FCT) of **1 Random** Client (secs)

Buffer Size	CC algorithm	Min	10	25	Median	75	90	max
2500B	Default CoAP	33.003	33.003	33.003	33.003	33.003	33.003	33.004
2500B	CoCoA	33.003	33.003	33.003	33.003	33.003	33.003	33.004
2500B	CoAP over TCP	41.900	41.924	43.590	45.584	46.239	46.417	46.580

Table 3: Flow completion time (FCT) of **10 Continuous** Clients (secs)

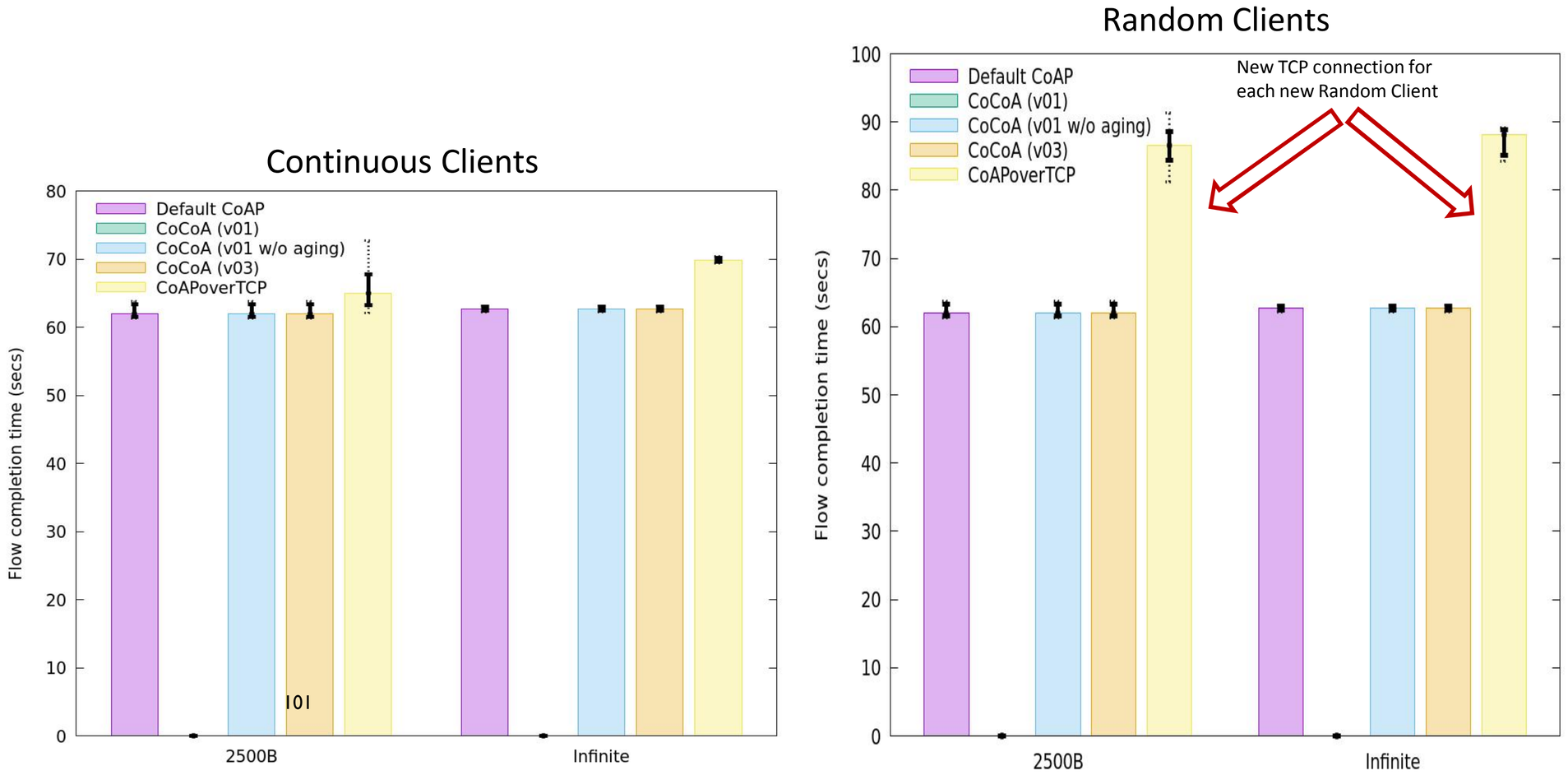
Buffer Size	CC algorithm	Min	10	25	Median	75	90	max
2500B	Default CoAP	33.043	33.044	33.126	33.220	33.300	33.335	33.387
2500B	CoCoA	33.043	33.044	33.126	33.220	33.299	33.334	33.387
2500B	CoAP over TCP	33.236	33.243	33.343	33.440	33.521	33.571	33.616

Table 4: Flow completion time (FCT) of **10 Random** Clients (secs)

Buffer Size	CC algorithm	Min	10	25	Median	75	90	max
2500B	Default CoAP	33.044	33.044	33.126	33.220	33.300	33.335	33.387
2500B	CoCoA	33.043	33.044	33.126	33.220	33.300	33.335	33.387
2500B	CoAP over TCP	40.494	43.245	43.821	46.166	46.663	46.872	47.202

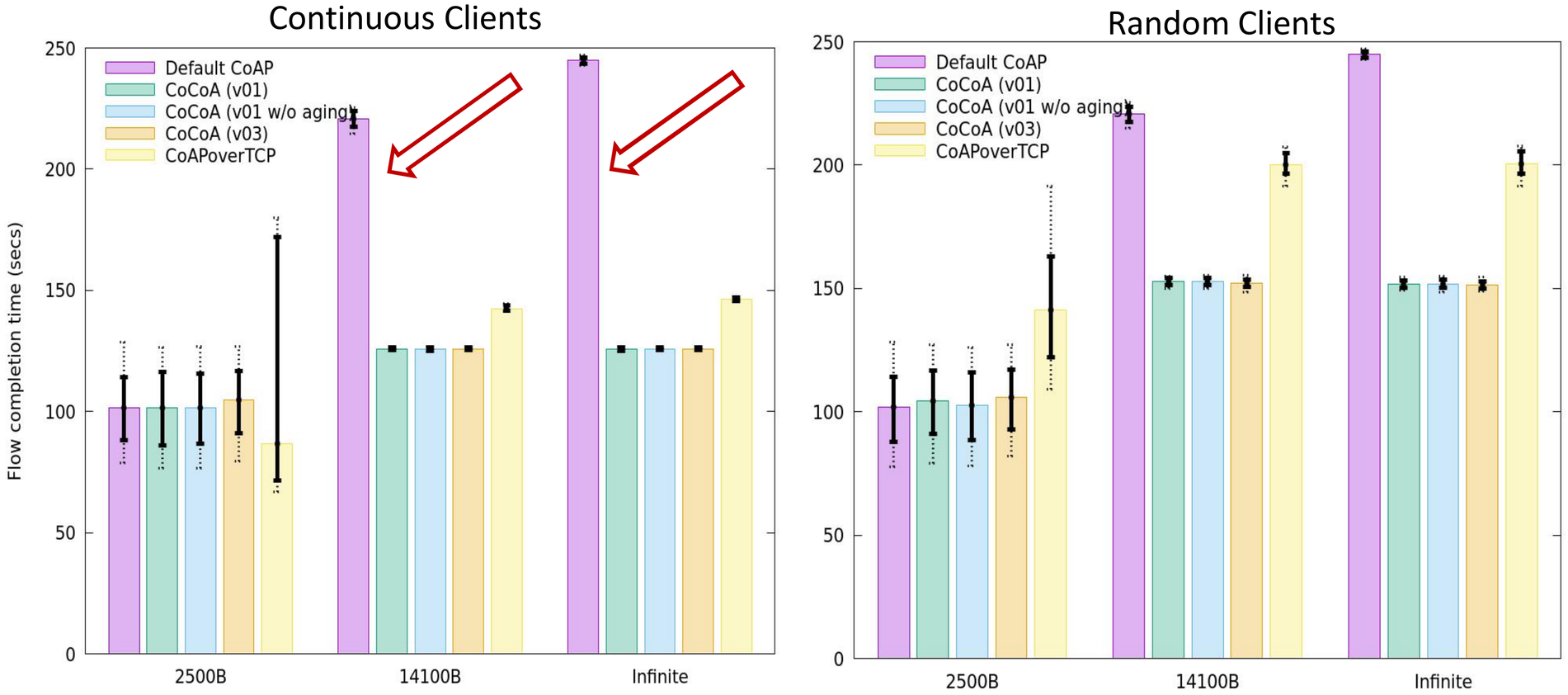
- With CoAP over TCP Random clients are clearly slower to complete compared to Continuous clients due to an additional RTT for TCP 3WHS when a new random client starts
- Larger TCP header causes some additional overhead for CoAP over TCP
- Queuing delay increases the flow completion times of 10 clients by up to a few hundreds milliseconds

FCT - 50 Clients



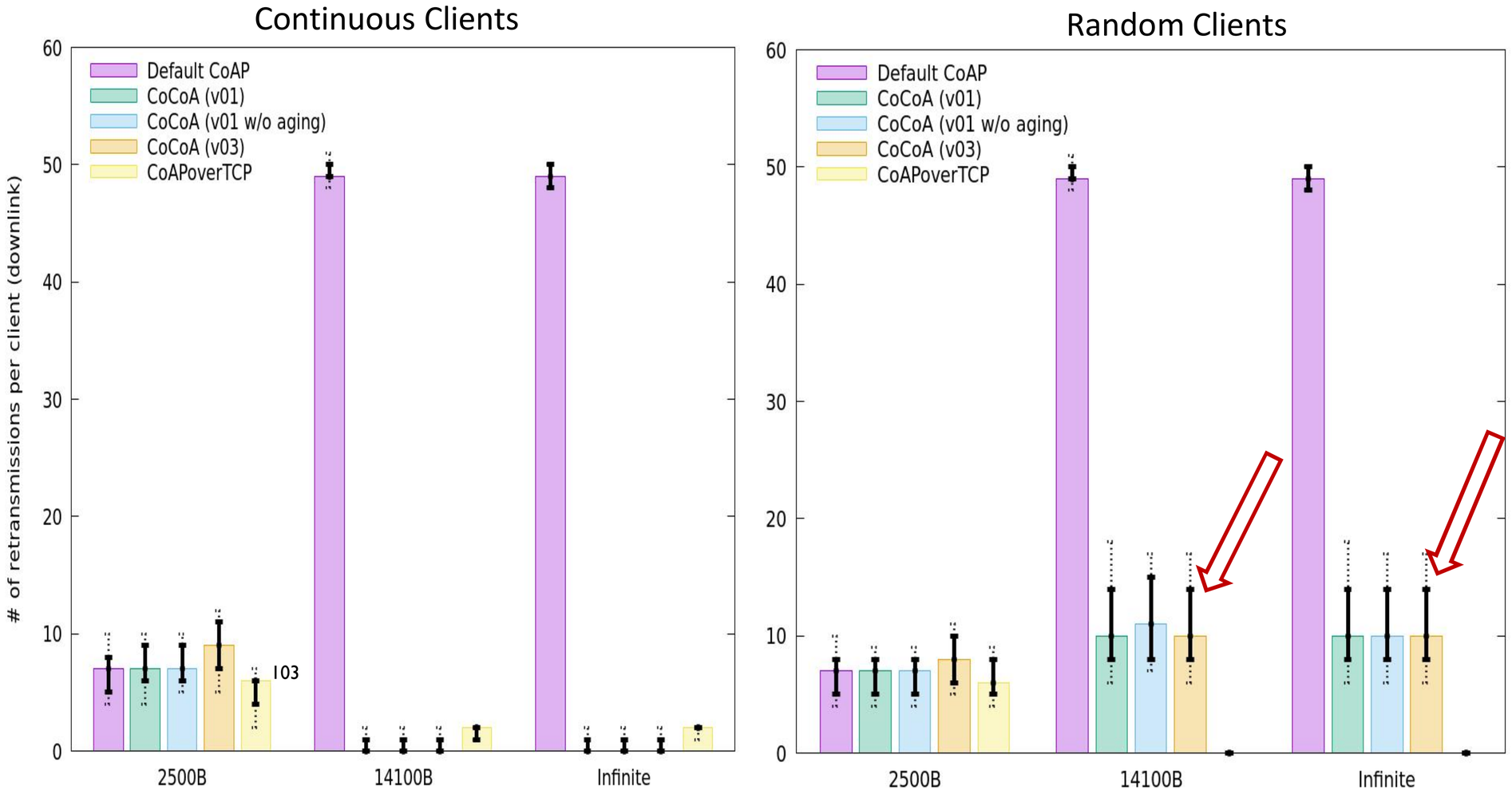
- Queuing delay increases and some congestion occurs resulting in a few packet losses
  - The major reason for the increased FCT is still in increased queuing delay
  - CoAP over TCP flows that encounter drops retransmit & back off -> FCT increases at higher percentiles
- “Infinite” queue: can absorb more packets eliminating packet losses with TCP
  - TCP more stanble, but slightly increased queuing delay because packets dropped with small buffer now fit into the buffer

FCT - 100 Clients



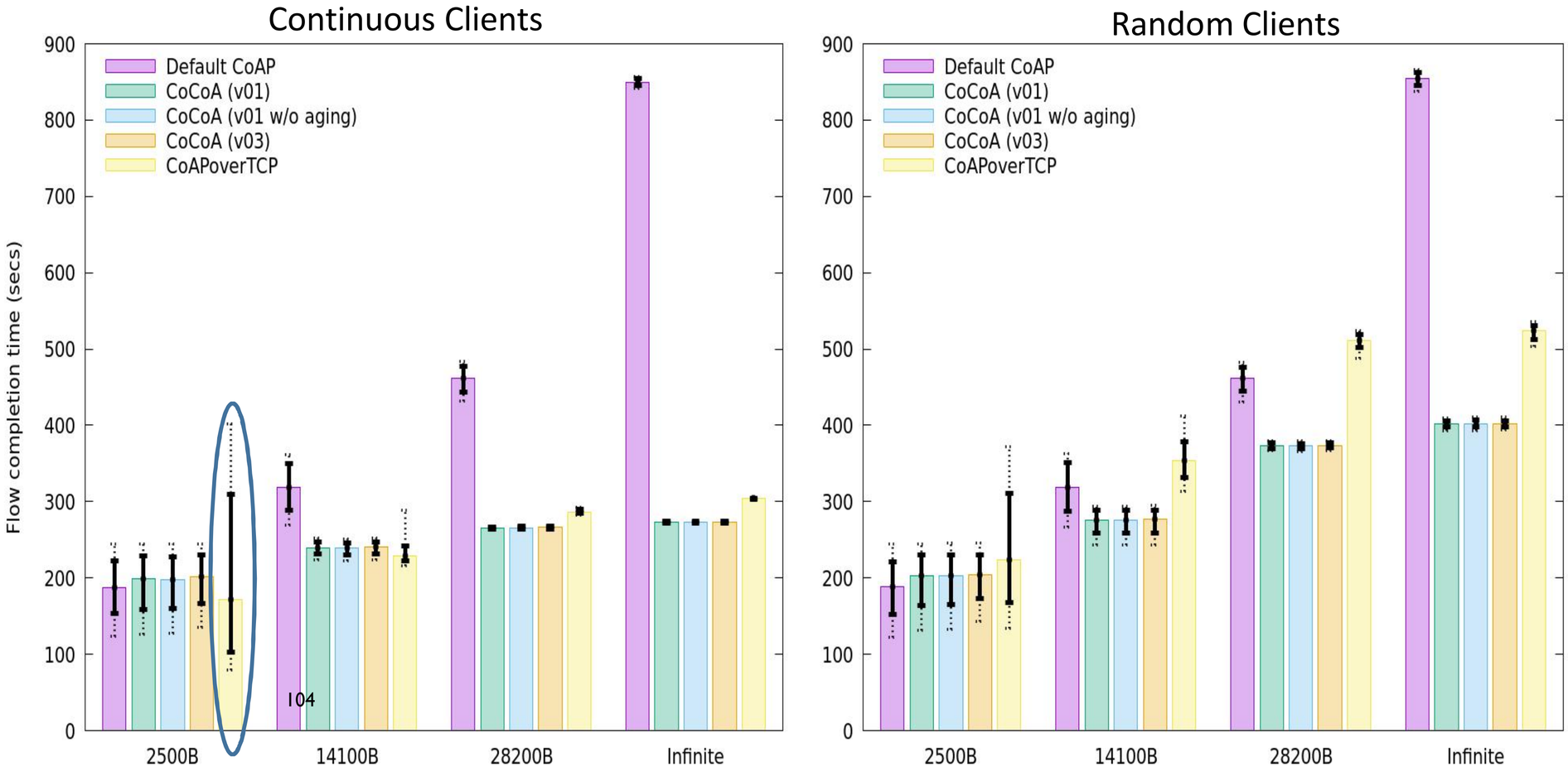
- Larger router buffers: longer queuing delay takes RTT over 2,5 secs
  - Unnecessary retransmissions with initial RTO
  - Default CoAP FCT increases significantly
    - Default CoAP unable to adjust its RTO unlike CoCoA and CoAP over TCP
    - Cannot avoid unnecessary retransmissions

Number of retransmissions - 100 Clients



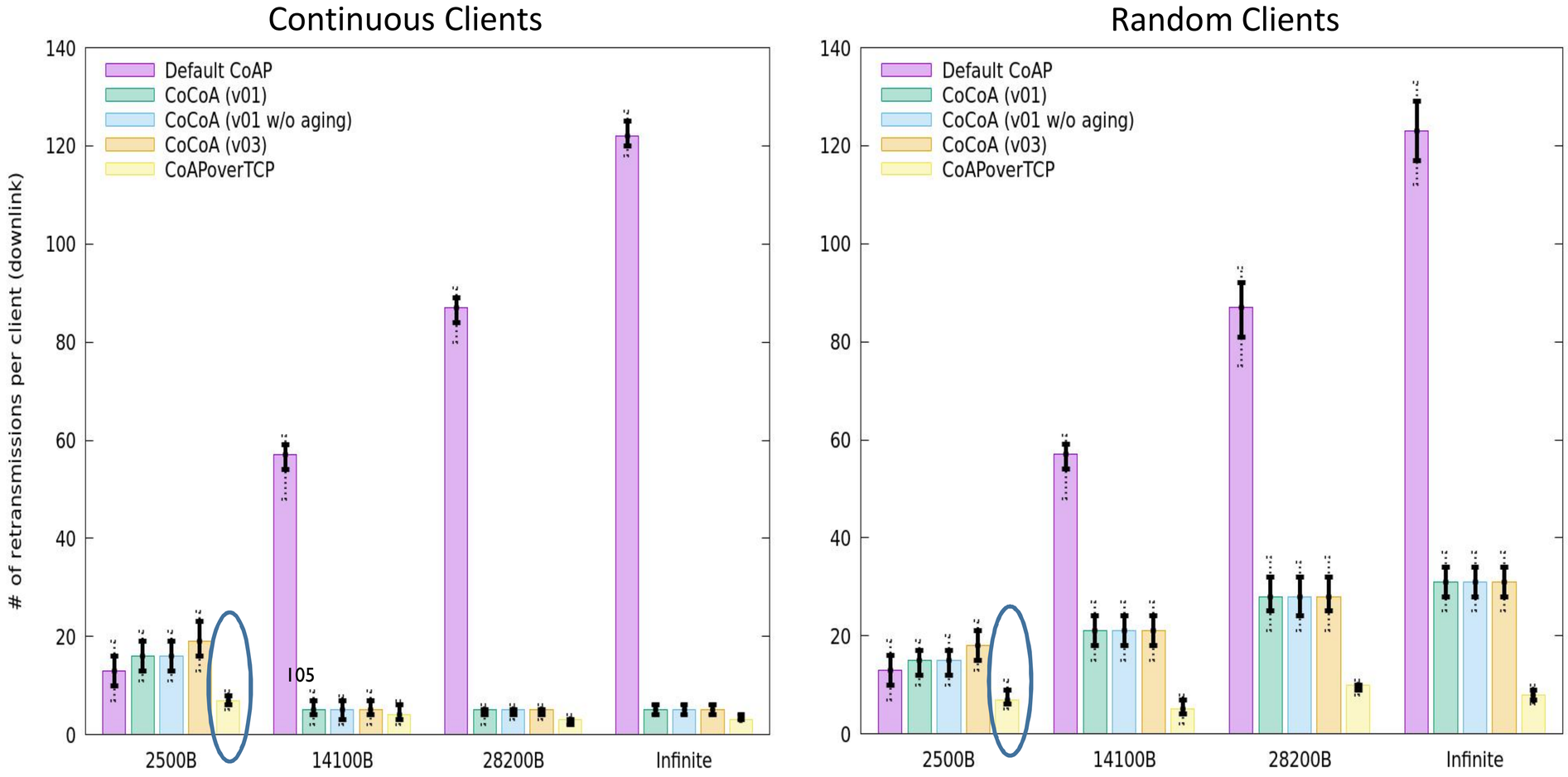
- With larger buffers Default CoAP unnecessarily retransmits nearly every requests once
- CoCoA and CoAP over TCP able to adjust RTO
  - CoCoA has more difficulties in adjusting RTO with Random Clients

FCT - 200 Clients



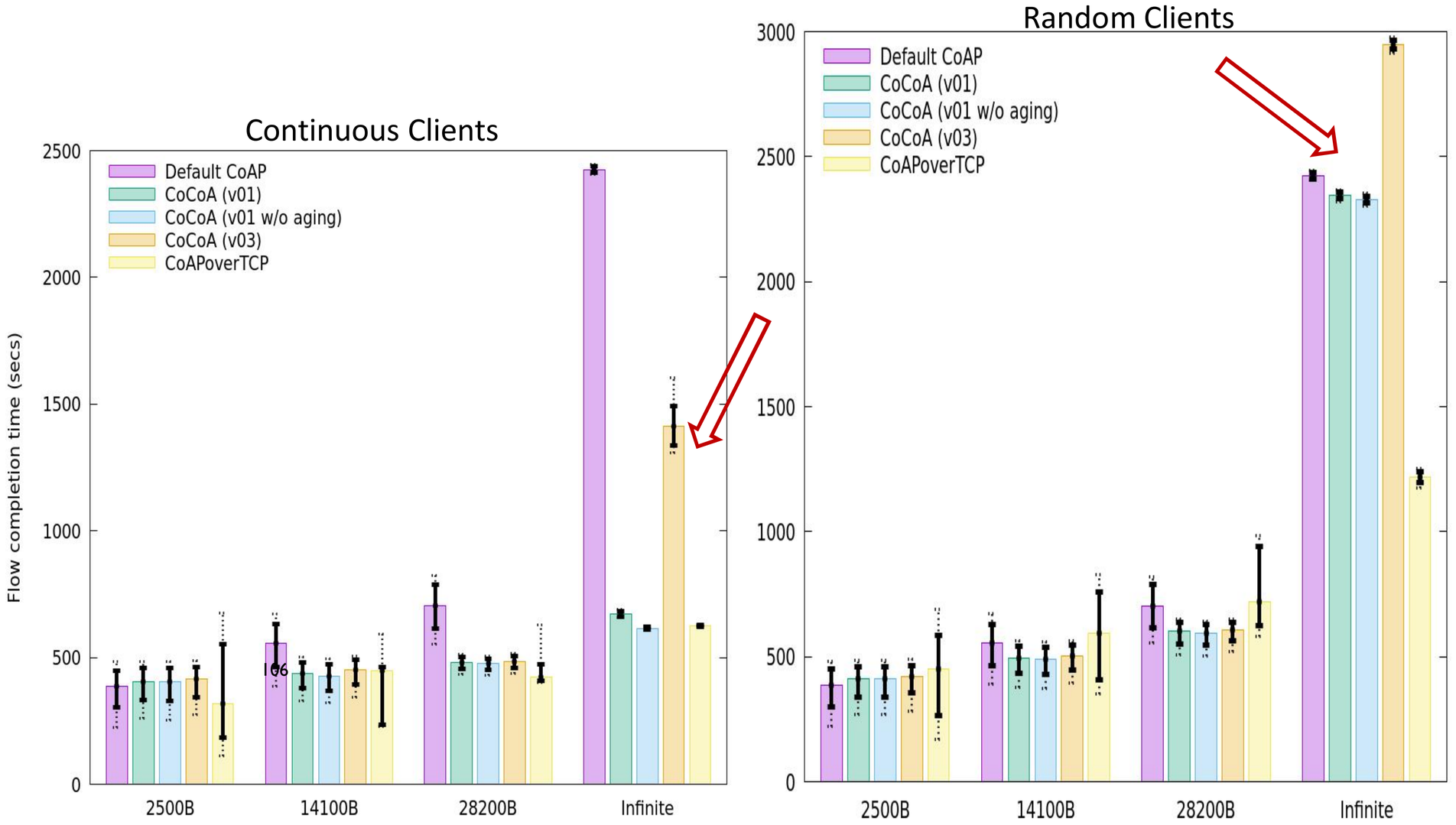
- Default CoAP Congestion Collapsive behavior with infinite buffer
  - A vast number of unnecessary retransmissions add to the queuing delay
- With smaller 2500 B buffer TCP responds congestion more effectively

Number of Retransmission – 200 Clients



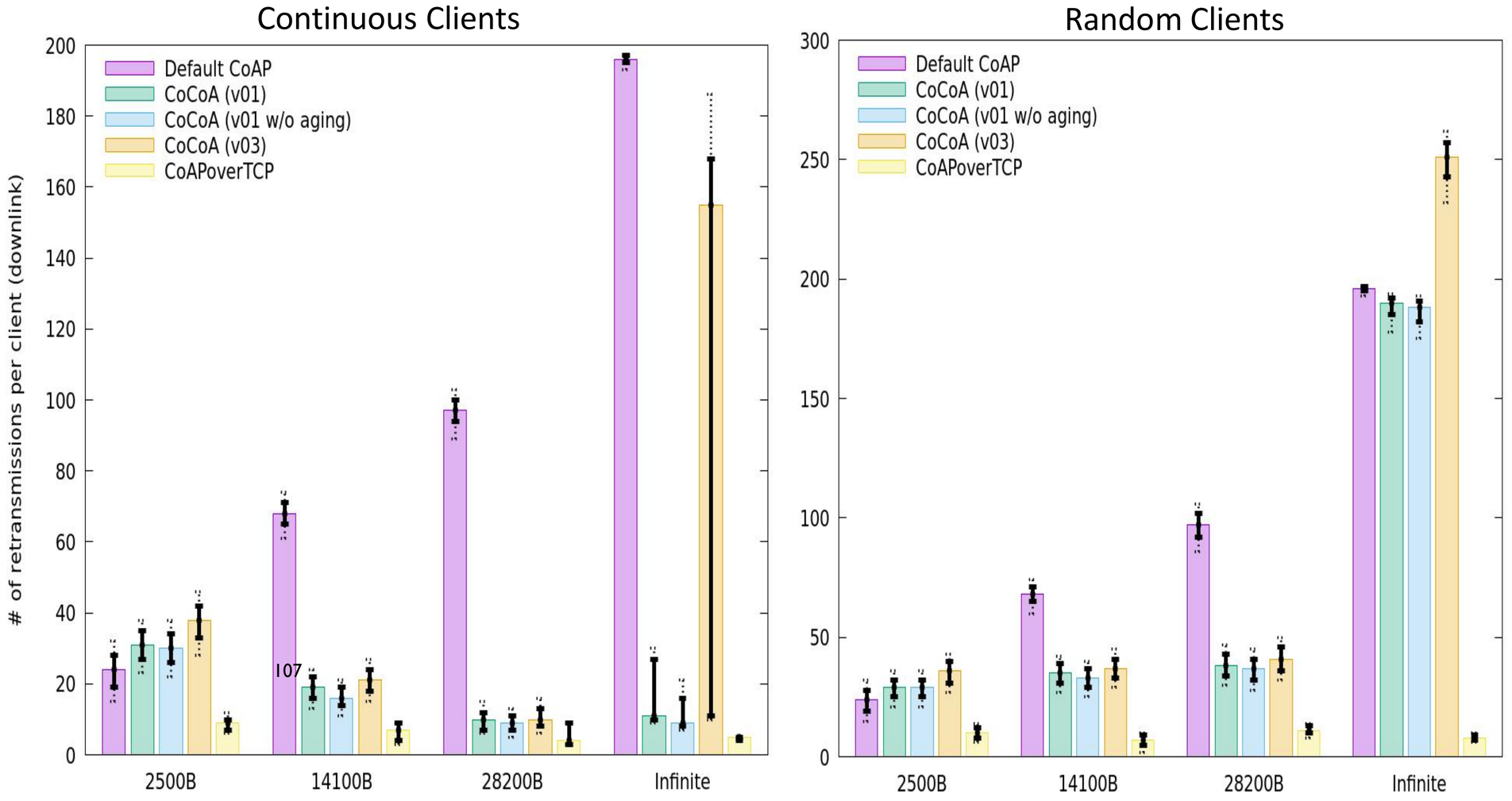
- Default CoAP: degree of Congestion Collapse increases with infinite buffer
  - Less forward progress made as most requests are unnecessarily retransmitted at least twice
- With smaller 2500 B buffer CoAP over TCP has clearly less lost packets & retransmissions
  - This decreases the congestion level and allows TCP to complete with a lower number of retransmissions than Default CoAP and CoCoA that have more undelivered retransmissions

FCT - 400 Clients



- Default CoAP Congestion Collapse degree even more higher with “infinite” buffer
  - Little forward progress made as almost all requests are unnecessarily retransmitted several times
- CoCoA -03 starts to collapse with Continuous Clients & “infinite” buffer
- Both CoCoA -01 and CoCoA -03 collapse with Random Clients & “infinite” buffer

Number of Retransmissions - 400 Clients



- CoCoA –v03 with Continuous Clients & “infinite” buffer:
  - RTT increases well above 10 secs -> Bit more than half of the clients are not able to adjust RTO
  - VBF of 1.5 does not allow CoAP exchanges with initial RTO (2-3 secs) to complete with 2 rexmits
  - Many clients that manage to get weak sample and RTO > 3 secs suffer from aging
  - CoCoA RTO has upper bound of 32 secs

Protocol Actions Needed

- Problems with Default CoAP (RFC 7252)
  - Does not employ full back off that is TCP-compatible
    - After retransmitting and backing off, restores 2 secs initial RTO for the next exchange
- Problems with CoCoA
  - Does not employ full back off that is TCP-compatible
    - After retransmitting and backing off, starts the next exchange with current RTO estimate
  - With RTO estimate > 3, applies aging that blindly decreases RTO estimate
    - Even if increase would be appropriate
  - Applies upper bound of 32 secs for RTO (in conflict with draft-ietf-tcpm-rto-consider)
- Action points (edit draft-ietf-core-cocoa; write a short I-D that updates RFC 7252):
  - Default CoAP and CoCoA: With confirmable message exchanges, add full congestion control back off (TCP-compatible):
    - After retransmitting and backing off RTO, retain the backed off RTO as initial RTO for the next new CoAP message exchange (CON-ACK)
    - Back off RTO further, if retransmissions needed
    - Restore RTO only after no retransmissions are needed to complete CoAP message exchange
  - CoCoA: Reconsider the use of aging with RTO > 3
  - CoCoA: Reconsider the use of 32 secs upper bound for RTO

Thank You!

Q & A

109

Backup Slides

110

Frequency of transmissions with 400 Continuous Clients

of (re)transmisssions (0= no retransmissions needed)

Buffer	CC algorithm	0.	1.	2.	3.	4.	5.	6.	7.	8.	9.	10.	11.	12.	13.	14.	15.	16.	17.	18.	19.	20.
2500B	Default CoAP	322141	29596	18627	11152	7285	4397	2797	1765	1046	614	313	224	43	0	0	0	0	0	0	0	0
2500B	CoCoA v01 no aging	305515	35350	22141	14023	8877	5552	3566	2029	1257	785	406	253	129	62	29	16	5	4	1	0	0
2500B	CoCoA v01	304088	35997	22306	14326	9050	5561	3563	2185	1269	731	404	255	135	75	26	13	10	5	1	0	0
2500B	CoCoA v03	288602	40215	25730	16393	10521	6845	4373	2805	1755	1143	673	400	244	150	77	34	28	8	1	2	1
2500B	CoAP over TCP	379379	14813	3145	1796	651	155	55	6	0	0	0	0	0	0	0	0	0	0	0	0	0
14100B	Default CoAP	13300	313354	37533	15324	9110	5041	2844	1614	918	478	261	123	61	26	11	2	0	0	0	0	0
14100B	CoCoA v01 no aging	336302	32459	14243	7702	4280	2420	1323	715	328	133	58	22	8	3	2	2	0	0	0	0	0
14100B	CoCoA v01	320863	45840	15275	8340	4464	2574	1318	729	336	175	43	18	14	3	5	2	1	0	0	0	0
14100B	CoCoA v03	313245	49480	16845	9167	5141	2883	1606	802	413	206	98	61	29	12	4	2	3	3	0	0	0
14100B	CoAP over TCP	381981	12839	4445	725	10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
28200B	Default CoAP	8940	90542	264319	17141	8971	4730	2562	1305	730	397	186	89	56	20	7	4	1	0	0	0	0
28200B	CoCoA v01 no aging	352079	31748	11140	3069	1265	484	150	50	8	2	2	3	0	0	0	0	0	0	0	0	0
28200B	CoCoA v01	344901	38161	11680	3192	1374	482	151	40	12	3	2	2	0	0	0	0	0	0	0	0	0
28200B	CoCoA v03	342004	39217	12601	3842	1441	576	200	81	25	6	4	3	0	0	0	0	0	0	0	0	0
28200B	CoAP over TCP	382098	10657	6516	722	7	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
infinite	Default CoAP	1474	3984	4793	7795	381954	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
infinite	CoCoA v01 no aging	347094	22138	19501	10422	845	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
infinite	CoCoA v01	330793	21864	24080	22051	1212	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
infinite	CoCoA v03	163015	15715	17113	106964	97031	162	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
infinite	CoAP over TCP	376388	10786	12646	180	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

If MAX_RETRANSMIT=4, requests starting from 5th retransmissions would have never completed

Frequency of transmissions with 200 Continuous Clients

of (re)transmisssions (0= no retransmissions needed)

Buffer	CC algorithm	0.	1.	2.	3.	4.	5.	6.	7.	8.	9.	10.	11.	12.	13.	14.	15.	16.	17.	18.	19.	20.
2500B	Default CoAP	174944	10911	6194	3440	2017	1266	769	398	61	0	0	0	0	0	0	0	0	0	0	0	0
2500B	CoCoA v01 no aging	171172	12415	7158	4062	2370	1324	821	394	189	70	25	0	0	0	0	0	0	0	0	0	0
2500B	CoCoA v01	170783	12688	7249	4194	2261	1338	767	440	186	70	24	0	0	0	0	0	0	0	0	0	0
2500B	CoCoA v03	166863	13643	8025	4781	2903	1622	987	570	344	178	62	22	0	0	0	0	0	0	0	0	0
2500B	CoAP over TCP	191300	7338	1055	233	63	6	5	0	0	0	0	0	0	0	0	0	0	0	0	0	0
14100B	Default CoAP	13781	165203	13139	4291	2057	914	392	149	59	15	0	0	0	0	0	0	0	0	0	0	0
14100B	CoCoA v01 no aging	184597	11985	2369	730	212	81	21	5	0	0	0	0	0	0	0	0	0	0	0	0	0
14100B	CoCoA v01	183300	13197	2366	812	248	55	21	1	0	0	0	0	0	0	0	0	0	0	0	0	0
14100B	CoCoA v03	183285	13149	2455	790	238	61	17	4	0	0	1	0	0	0	0	0	0	0	0	0	0
14100B	CoAP over TCP	189559	9369	982	88	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
28200B	Default CoAP	6483	52253	136114	3372	1195	403	135	31	8	6	0	0	0	0	0	0	0	0	0	0	0
28200B	CoCoA v01 no aging	183147	16100	729	24	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
28200B	CoCoA v01	183277	15982	716	25	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
28200B	CoCoA v03	183151	15863	930	50	6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
28200B	CoAP over TCP	189870	9378	746	6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
infinite	Default CoAP	1489	4281	98156	96074	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
infinite	CoCoA v01 no aging	181374	17878	748	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
infinite	CoCoA v01	181283	18027	690	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
infinite	CoCoA v03	181517	17525	958	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
infinite	CoAP over TCP	188465	9251	2284	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

112

Frequency of transmissions with 100 Continuous Clients

of (re)transmisssions (0= no retransmissions needed)

Buffer	CC algorithm	0.	1.	2.	3.	4.	5.	6.	7.	8.	9.	10.	11.	12.	13.	14.	15.	16.	17.	18.	19.	20.
2500B	Default CoAP	92211	4045	1925	945	540	275	59	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2500B	CoCoA v01 no aging	92270	4016	1834	969	518	277	113	3	0	0	0	0	0	0	0	0	0	0	0	0	0
2500B	CoCoA v01	92372	3867	1883	999	486	291	101	1	0	0	0	0	0	0	0	0	0	0	0	0	0
2500B	CoCoA v03	91073	4384	2251	1127	603	302	184	76	0	0	0	0	0	0	0	0	0	0	0	0	0
2500B	CoAP over TCP	96128	3358	427	72	15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
14100B	Default CoAP	3069	95295	1529	88	18	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
14100B	CoCoA v01 no aging	99133	867	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
14100B	CoCoA v01	99103	897	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
14100B	CoCoA v03	99072	928	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
14100B	CoAP over TCP	96968	3031	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
28200B	Default CoAP	2100	97900	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
28200B	CoCoA v01 no aging	99058	942	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
28200B	CoCoA v01	99148	852	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
28200B	CoCoA v03	99091	909	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
28200B	CoAP over TCP	96269	3731	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
infinite	Default CoAP	2086	97914	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
infinite	CoCoA v01 no aging	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
infinite	CoCoA v01	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
infinite	CoCoA v03	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
infinite	CoAP over TCP	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

113

Frequency of transmissions with 400 Random Clients

of (re)transmisssions (0= no retransmissions needed)

Buffer	CC algorithm	0.	1.	2.	3.	4.	5.	6.	7.	8.	9.	10.	11.	12.	13.	14.	15.	16.	17.	18.	19.	20.
2500B	Default CoAP	322890	29597	18359	11029	6863	4351	2818	1838	1093	589	324	183	66	0	0	0	0	0	0	0	0
2500B	CoCoA v01 no aging	309389	34263	21393	13419	8506	5109	3205	2009	1190	699	391	220	104	52	31	12	4	4	0	0	0
2500B	CoCoA v01	308777	34686	21459	13420	8521	5182	3341	1914	1198	653	387	237	114	58	22	18	8	3	2	0	0
2500B	CoCoA v03	293158	38706	24678	15960	10131	6544	4238	2626	1617	1013	551	334	191	101	77	35	26	4	5	5	0
2500B	CoAP over TCP	371064	27009	1613	260	37	13	4	0	0	0	0	0	0	0	0	0	0	0	0	0	0
14100B	Default CoAP	13522	313438	37171	15449	8998	5075	2858	1638	901	459	278	115	60	24	14	0	0	0	0	0	0
14100B	CoCoA v01 no aging	225204	132527	20737	10185	5487	2919	1486	757	396	162	79	36	16	7	1	1	0	0	0	0	0
14100B	CoCoA v01	214906	141060	21524	10745	5636	3000	1551	835	380	212	91	31	17	6	5	1	0	0	0	0	0
14100B	CoCoA v03	207247	143612	23599	11700	6448	3461	1844	996	512	299	138	73	45	11	7	4	2	0	2	0	0
14100B	CoAP over TCP	376011	22802	1060	104	19	1	2	0	1	0	0	0	0	0	0	0	0	0	0	0	0
28200B	Default CoAP	9166	90205	264458	17290	8773	4736	2508	1366	768	350	206	91	51	23	7	2	0	0	0	0	0
28200B	CoCoA v01 no aging	198652	127711	63157	6249	2519	1087	381	161	53	22	6	2	0	0	0	0	0	0	0	0	0
28200B	CoCoA v01	188636	135550	64720	6560	2757	1115	428	149	62	13	8	1	0	0	1	0	0	0	0	0	0
28200B	CoCoA v03	182168	135792	66850	9689	3195	1395	552	221	78	35	17	6	1	1	0	0	0	0	0	0	0
28200B	CoAP over TCP	335028	64066	816	70	18	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
infinite	Default CoAP	1482	3995	4797	7965	381761	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
infinite	CoCoA v01 no aging	3264	15066	14599	25528	341539	4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
infinite	CoCoA v01	2415	11288	15167	25785	345344	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
infinite	CoCoA v03	1620	9531	14208	5175	11801	256522	101143	0	0	0	0	0	0	0	0	0	0	0	0	0	0
infinite	CoAP over TCP	336174	62862	964	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

114

If MAX_RETRANSMIT=4, requests starting from 5th retransmissions would have never completed

Frequency of transmissions with 200 Random Clients

of (re)transmisssions (0= no retransmissions needed)

Buffer	CC algorithm	0.	1.	2.	3.	4.	5.	6.	7.	8.	9.	10.	11.	12.	13.	14.	15.	16.	17.	18.	19.	20.
2500B	Default CoAP	175150	10862	6036	3410	2055	1237	790	418	42	0	0	0	0	0	0	0	0	0	0	0	0
2500B	CoCoA v01 no aging	172660	11920	6825	3891	2228	1244	667	336	159	57	13	0	0	0	0	0	0	0	0	0	0
2500B	CoCoA v01	172693	12081	6811	3752	2170	1191	711	354	149	70	18	0	0	0	0	0	0	0	0	0	0
2500B	CoCoA v03	168360	13375	7870	4481	2560	1491	893	482	256	136	71	25	0	0	0	0	0	0	0	0	0
2500B	CoAP over TCP	187094	12209	620	64	11	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
14100B	Default CoAP	13802	165213	13214	4200	1984	969	392	149	61	16	0	0	0	0	0	0	0	0	0	0	0
14100B	CoCoA v01 no aging	126184	66825	4839	1439	501	154	41	17	0	0	0	0	0	0	0	0	0	0	0	0	0
14100B	CoCoA v01	125849	67186	4818	1399	523	158	50	12	5	0	0	0	0	0	0	0	0	0	0	0	0
14100B	CoCoA v03	125696	67161	4873	1554	487	151	53	20	4	1	0	0	0	0	0	0	0	0	0	0	0
14100B	CoAP over TCP	187009	12546	425	19	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
28200B	Default CoAP	6815	51773	136145	3518	1169	400	123	37	10	7	3	0	0	0	0	0	0	0	0	0	0
28200B	CoCoA v01 no aging	105131	77645	17175	47	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
28200B	CoCoA v01	104805	77896	17232	60	7	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
28200B	CoCoA v03	104491	77773	17618	114	2	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
28200B	CoAP over TCP	162192	37783	25	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
infinite	Default CoAP	1493	4257	95393	98857	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
infinite	CoCoA v01 no aging	96754	82542	20704	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
infinite	CoCoA v01	96775	82855	20370	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
infinite	CoCoA v03	97178	81939	20880	3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
infinite	CoAP over TCP	168244	31756	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

115

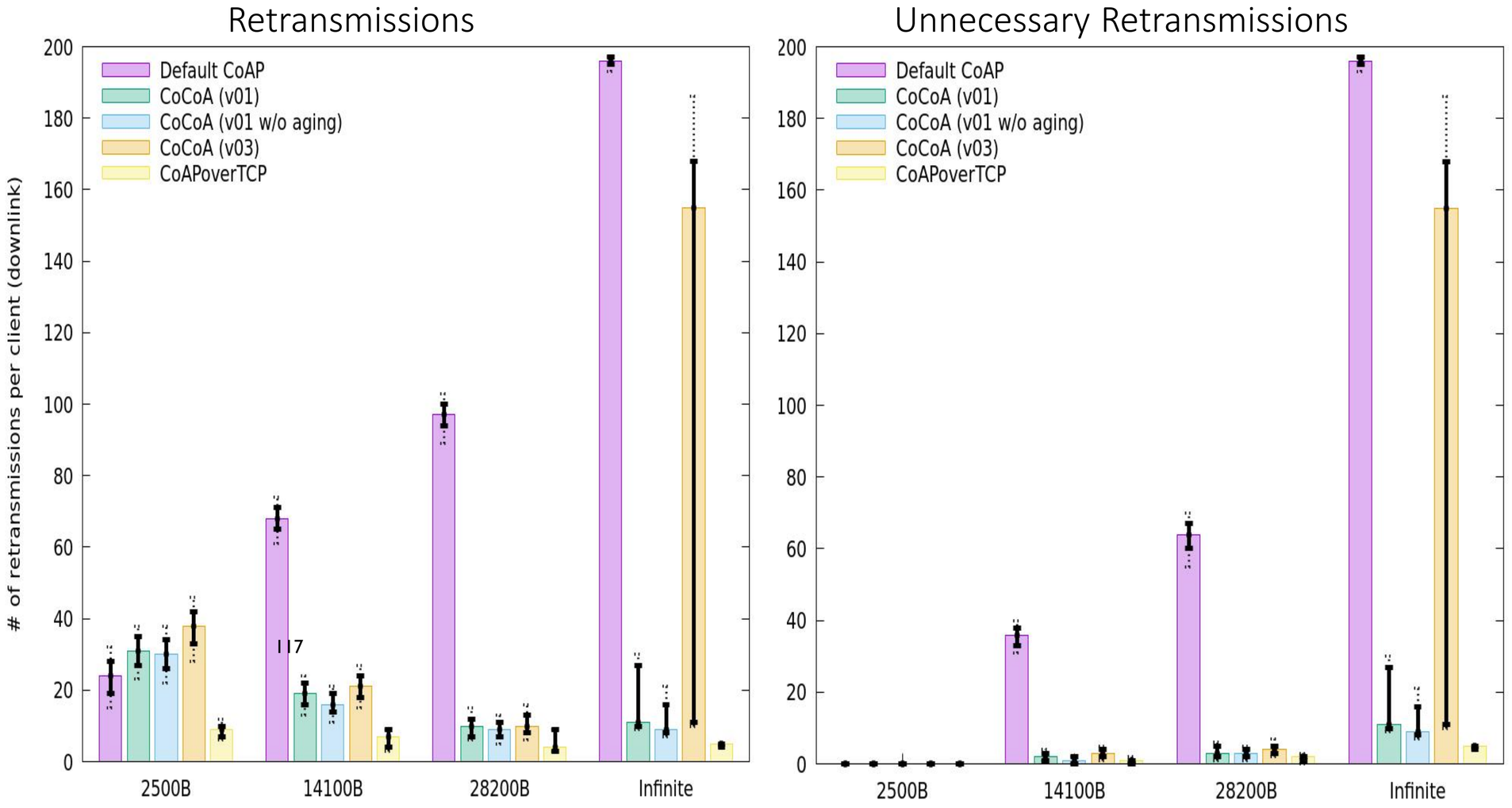
Frequency of transmissions with 100 Random Clients

of (re)transmisssions (0= no retransmissions needed)

Buffer	CC algorithm	0.	1.	2.	3.	4.	5.	6.	7.	8.	9.	10.	11.	12.	13.	14.	15.	16.	17.	18.	19.	20.
2500B	Default CoAP	92328	3968	1895	949	515	282	63	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2500B	CoCoA v01 no aging	92955	3694	1734	882	459	197	76	3	0	0	0	0	0	0	0	0	0	0	0	0	0
2500B	CoCoA v01	92665	3843	1823	926	480	206	56	1	0	0	0	0	0	0	0	0	0	0	0	0	0
2500B	CoCoA v03	91655	4292	2069	1059	518	241	131	33	2	0	0	0	0	0	0	0	0	0	0	0	0
2500B	CoAP over TCP	92917	6832	233	18	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
14100B	Default CoAP	3105	95327	1450	100	16	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
14100B	CoCoA v01 no aging	77404	22596	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
14100B	CoCoA v01	77358	22641	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
14100B	CoCoA v03	77969	22031	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
14100B	CoAP over TCP	100000	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
28200B	Default CoAP	2114	97886	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
28200B	CoCoA v01 no aging	78134	21866	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
28200B	CoCoA v01	77983	22017	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
28200B	CoCoA v03	78343	21657	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
28200B	CoAP over TCP	100000	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
infinite	Default CoAP	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
infinite	CoCoA v01 no aging	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
infinite	CoCoA v01	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
infinite	CoCoA v03	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
infinite	CoAP over TCP	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

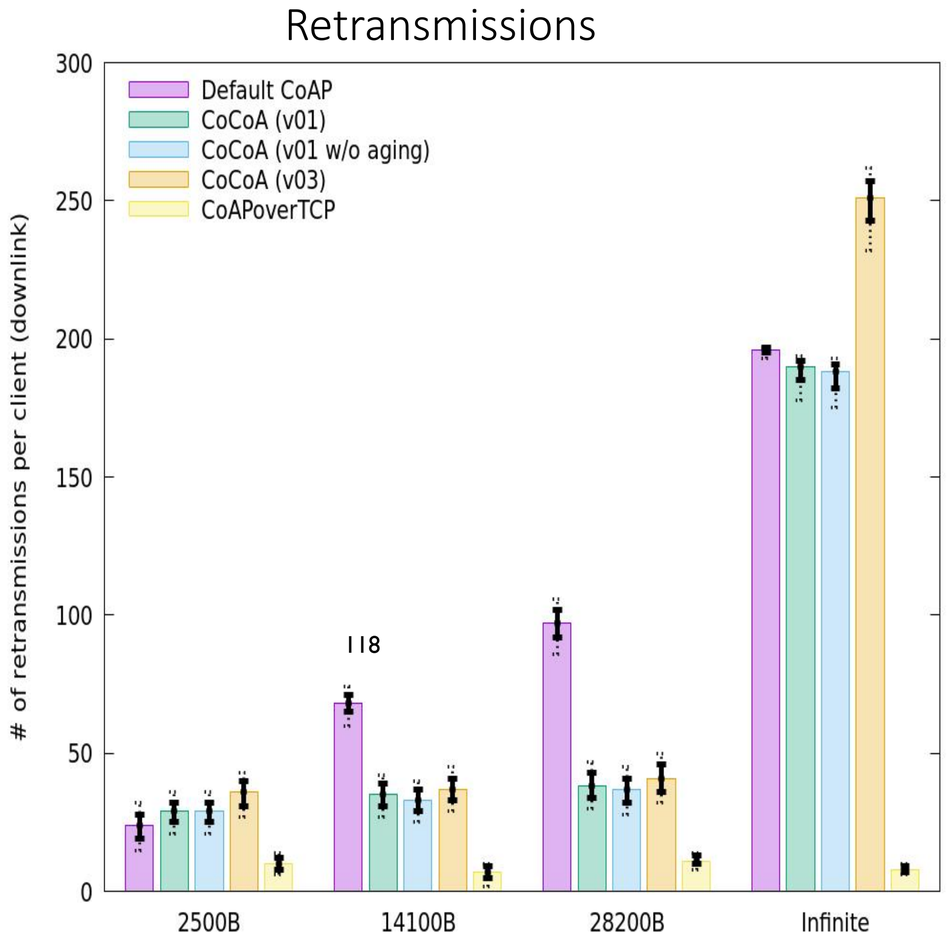
116

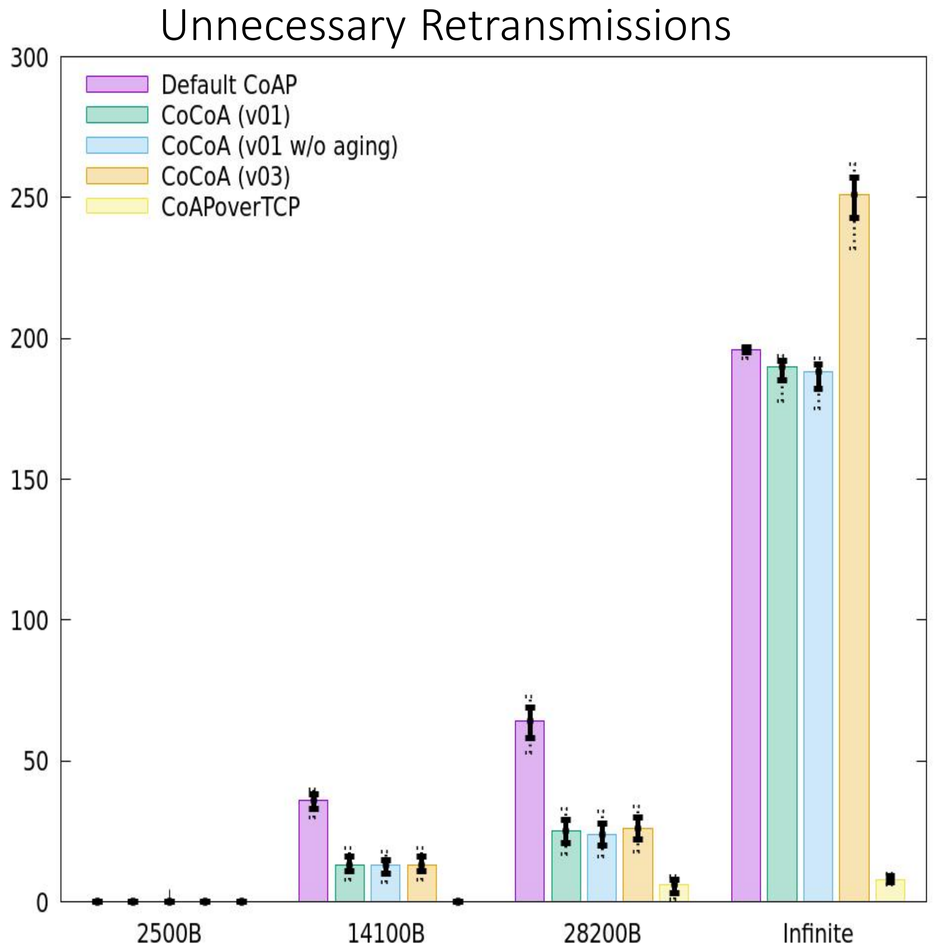
Number of Retransmissions - 400 Continuous Clients



- Default CoAP
- With small 2500 B buffer TCP responds congestion more effectively

Number of retransmissions - 400 Random Clients





All times are in time-warped WET (UTC+00:00)

Tuesday (150 min)

- 09:30–09:35 Intro, Agenda
- 09:35–10:00 Post-WGLC: CoCoA (CG)
- 10:00–10:10 dev URN (JA)
- 10:10–10:25 Getting ready: ERT (CA)
- ~~10:15–10:25 Getting ready: OSCORE-Group (MT)~~
- ~~10:25–10:40 New response codes (AK)~~
- 10:25–10:40 Pending for EST (PV)
- 10:40–10:50 Pubsub (MK)
- 10:50–11:00 Dynlink/Interfaces (BS)
- 11:00–11:10 Negotiation, AT (BS)
- 11:10–12:00 Flextime: OPC/UA (CP), Time scale (LT), …

Draft-ietf-core-dev-urn-01

Arkko, Jennings & Shelby

A Uniform Resource Name (URN) namespace for hardware device identifiers.

Potentially useful in applications such as in sensor data streams and storage, or equipment inventories.

Complements other similar identifiers NIs (RFC 6920), UUIDs (RFC 4122), IMEIs (RFC 7254) etc. Supports, e.g., MAC and EUI-64, identifiers.

urn:dev:mac:0024befffe804ff1

Versions -00 and -01

- -01 was published this week
- Fixed a typo in the ABNF (“dn:” => “org:”)
- Conformance to the URN registration template

Next Steps

- Can people read the new template (Section 3)?
- What should the draft say about q-, r-, and f-components?
- Needs text and decision: adding device IDs specified in OneM2M and LWM2M (urn:dev:os and urn:dev:ops)?
  - And would BBF USP protocol identifiers be useful to add as well?
- Adding other, new device identification schemes related to Web of Things work (e.g., urn:dev:wot:something:mysensor1)
  - Note: the DEV URN scheme allows extension to new types, do not have to define everything now
  - But getting the initial set of the relevant ones would be very useful

All times are in time-warped WET (UTC+00:00)

Tuesday (150 min)

- 09:30–09:35 Intro, Agenda
- 09:35–10:00 Post-WGLC: CoCoA (CG)
- 10:00–10:10 dev URN (JA)
- 10:10–10:25 Getting ready: ERT (CA)
- ~~10:15–10:25 Getting ready: OSCORE-Group (MT)~~
- ~~10:25–10:40 New response codes (AK)~~
- 10:25–10:40 Pending for EST (PV)
- 10:40–10:50 Pubsub (MK)
- 10:50–11:00 Dynlink/Interfaces (BS)
- 11:00–11:10 Negotiation, AT (BS)
- 11:10–12:00 Flextime: OPC/UA (CP), Time scale (LT), …

Echo and Request Tag

draft-ietf-core-echo-request-tag

Christian Amsüss, John Mattson, Göran Selander

2018-03-20

Update 7252 Token processing

mitigates attacks described in `coap-actuators`

Echo updated for readability

Request-Tag can be simpler

as we understand block-wise

see “Strictness of RFC7959” thread

We want YOU for. . .

reviews

All times are in time-warped WET (UTC+00:00)

Tuesday (150 min)

- 09:30–09:35 Intro, Agenda
- 09:35–10:00 Post-WGLC: CoCoA (CG)
- 10:00–10:10 dev URN (JA)
- 10:10–10:25 Getting ready: ERT (CA)
- ~~10:15–10:25 Getting ready: OSCORE-Group (MT)~~
- ~~10:25–10:40 New response codes (AK)~~
- 10:25–10:40 Pending for EST (PV)
- 10:40–10:50 Pubsub (MK)
- 10:50–11:00 Dynlink/Interfaces (BS)
- 11:00–11:10 Negotiation, AT (BS)
- 11:10–12:00 Flextime: OPC/UA (CP), Time scale (LT), …

‘Pending’ response code

Peter van der Stok, Klaus Hartke

IETF 101 - CoRE Working Group

Motivation

RFC 7030:
Enrollment over Secure Transport (EST)
uses http response 202 when result is not immediately available (say: 3 hours) in response to GET or POST.

131

No such response code exists for coap.
This functionality is needed for EST over coap.

HTTP 202

The request has been accepted for processing, but the processing has not been completed. The request might or might not eventually be acted upon, as it might be disallowed when processing actually takes place.
The representation sent with this response ought to describe the request's current status and point to (or embed) a status monitor that can provide the user with an estimate of when the request will be fulfilled.

132

Use cases

draft-ietf-ace-coap-est specifies requests to servers to verify a node's identity; this may need manual intervention and takes a minimum response time

draft-ietf-core-coap-pubsub specifies a server to send a response to the client to indicate a valid request but may contain an empty payload.

133

draft-keranen-core-too-many-reqs specifies that response is available after minimum response time

History

A new response code (e.g. 2.06) was deemed harmful for proxies. (They will return 5.01 (Not Implemented))

An extension to response code 5.03 “Service Unavailable” does not cover the case because service is available[134]

This draft specifies a content format “60001” extension to existing response codes

Details

- Pending response indicates that target resource exists, but no representation is available yet.
- Location may be specified where result will become available.
- Allows multiple clients to have multiple concurrent requests open at the server.
- Client has to retry with GET request after Max-Age.
- Can be used in conjunction with “observe”

Pushing application-specific state machines into CoAP?

- How should application-specific state machines be added to CoAP applications?
- REST approach: transfer **representations**
- Need to define **media types** for those application states
- Related trial balloon: draft-bormann-core-maybe-00

All times are in time-warped WET (UTC+00:00)

Tuesday (150 min)

- 09:30–09:35 Intro, Agenda
- 09:35–10:00 Post-WGLC: CoCoA (CG)
- 10:00–10:10 dev URN (JA)
- 10:10–10:25 Getting ready: ERT (CA)
- ~~10:15–10:25 Getting ready: OSCORE-Group (MT)~~
- ~~10:25–10:40 New response codes (AK)~~
- 10:25–10:40 Pending for EST (PV)
- 10:40–10:50 Pubsub (MK)
- 10:50–11:00 Dynlink/Interfaces (BS)
- 11:00–11:10 Negotiation, AT (BS)
- 11:10–12:00 Flextime: OPC/UA (CP), Time scale (LT), …

All times are in time-warped WET (UTC+00:00)

Tuesday (150 min)

- 09:30–09:35 Intro, Agenda
- 09:35–10:00 Post-WGLC: CoCoA (CG)
- 10:00–10:10 dev URN (JA)
- 10:10–10:25 Getting ready: ERT (CA)
- ~~10:15–10:25 Getting ready: OSCORE-Group (MT)~~
- ~~10:25–10:40 New response codes (AK)~~
- 10:25–10:40 Pending for EST (PV)
- 10:40–10:50 Pubsub (MK)
- 10:50–11:00 Dynlink/Interfaces (BS)
- 11:00–11:10 Negotiation, AT (BS)
- 11:10–12:00 Flextime: OPC/UA (CP), Time scale (LT), …

All times are in time-warped WET (UTC+00:00)

Tuesday (150 min)

- 09:30–09:35 Intro, Agenda
- 09:35–10:00 Post-WGLC: CoCoA (CG)
- 10:00–10:10 dev URN (JA)
- 10:10–10:25 Getting ready: ERT (CA)
- ~~10:15–10:25 Getting ready: OSCORE-Group (MT)~~
- ~~10:25–10:40 New response codes (AK)~~
- 10:25–10:40 Pending for EST (PV)
- 10:40–10:50 Pubsub (MK)
- 10:50–11:00 Dynlink/Interfaces (BS)
- 11:00–11:10 Negotiation, AT (BS)
- 11:10–12:00 Flextime: OPC/UA (CP), Time scale (LT), …

CoAP Protocol Negotiation

draft-silverajan-core-coap-protocol-negotiation-08

Bill Silverajan TUT
Mert Ocak Ericsson

Context

- Aimed at CoAP nodes that have multiple transports, and wish to allow CoAP requests and responses over some or all these transports
- Both per-server and per-resource models supported
- Allows clients to directly query origin servers for available transports and communicate using an alternative transport (using a CoAP Option, or a link attribute)
- When a CoRE Resource Directory is present, origin servers can also register transport availability to RD for clients to query (using new parameter types)

Current Status

- Updates from -07 to -08
  - ‘ol’ is now a repeatable attribute allowing multiple base URIs, to align it with OCF ‘ep’
  - ‘at’ is now a repeatable parameter for registering alternate transports at the RD
  - Better examples provided
  - Updated example usage with RD, based on suggestions found in draft-ietf-core-resource-directory-13

Next Steps

- Evaluate other means to obtain transport endpoints from the origin server in place of Alternative-Transports Option
  - Using FETCH
  - Using an entry in .well-known/ for site-wide metadata (either core or something else)
  - Using a resource such as ”/pn/” with resource type ”core.pn” and content type application/link-format

CoAP Communication with Alternative Transports

draft-silverajan-core-coap-alternative-transports-11

Bill Silverajan TUT
Teemu Savolainen Nokia

Context

- Draft's focus is on the URI design work for CoAP over alternative transports
  - If you need to embed the transport information in a CoAP URI, which URI component should be used?

scheme://host:port/path/to/resource?query

- The URI query, path and authority components were all disqualified based on identified requirements
- Technical requirements leave only the URI scheme as the best place to embed transport identification

Current Status

- Draft -11 is a small delta to -10
- The work has been completed
  - Listed as an informative reference to RFC 8323
- Next step is for WG adoption

All times are in time-warped WET (UTC+00:00)

Tuesday (150 min)

- 09:30–09:35 Intro, Agenda
- 09:35–10:00 Post-WGLC: CoCoA (CG)
- 10:00–10:10 dev URN (JA)
- 10:10–10:25 Getting ready: ERT (CA)
- ~~10:15–10:25 Getting ready: OSCORE Group (MT)~~
- ~~10:25–10:40 New response codes (AK)~~
- 10:25–10:40 Pending for EST (PV)
- 10:40–10:50 Pubsub (MK)
- 10:50–11:00 Dynlink/Interfaces (BS)
- 11:00–11:10 Negotiation, AT (BS)
- 11:10–12:00 Flextime: OPC/UA (CP), Time scale (LT), …

OPC UA Message Transmission Method over CoAP

draft-wang-core-opcua-transmission-03

Ping Wang, Chenggen Pu,
Heng Wang, Junrui Wu, Yi Yang,
Lun Shao, Jianqiang Hou

London, March 20, 2018

Status

- Last version is 02.
- Made some meaningful changes according to the last meeting comments.
- Keep the draft updated.

What We Have Updated

Three use cases:

Offline/Online diagnostic system for resource-constrained factories,
Factory data monitoring based on web pages,
Factory data analysis based on cloud.

Consolidate two transmission schemes into one:

Consolidate the proxy for OPC UA-CoAP and the direct transmission into one to realize better transmission performance.

Next Steps

Contact with OPC Foundation to get feedback.

Implement the transmission schemes mentioned above over a reasonable architecture.

Comments or Questions?
Thank you!