A Concise Binary Object Representation (CBOR) of DNS Messages

draft-lenders-dns-cbor
(https://datatracker.ietf.org/doc/draft-lenders-dns-cbor/)

Martine S. Lenders (martine-lenders@tu-dresden.de),
Carsten Bormann, Thomas C. Schmidt, Matthias Wählisch

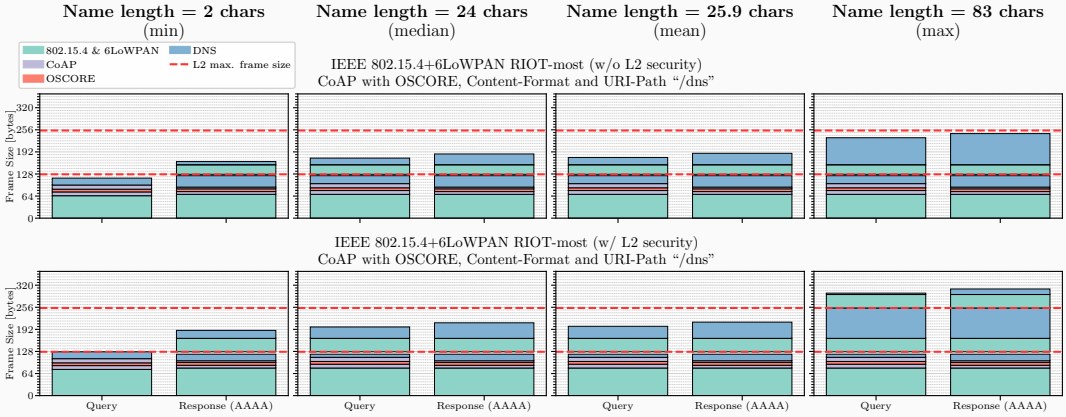
IETF 118, CBOR WG Session, 2023-11-07

Outline

Motivation: DNS in Constrained Networks

Packet size in DoC exceeds 802.15.4 PDU depending on queried name length

⇒ Fragmentation

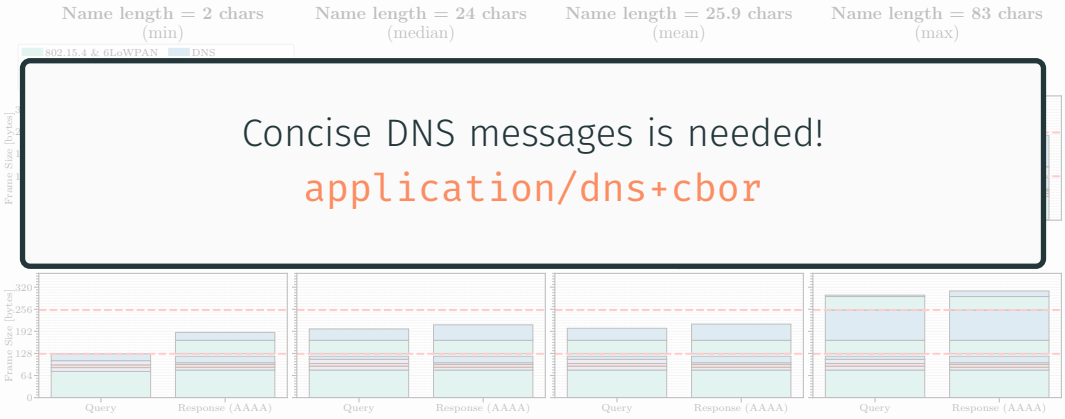


DNS over CoAP (`draft-ietf-core-dns-over-coap`) messages for different name lengths

Motivation: DNS in Constrained Networks

Packet size in DoC exceeds 802.15.4 PDU depending on queried name length

⇒ Fragmentation



Concise DNS messages is needed!

`application/dns+cbor`

DNS over CoAP (`draft-ietf-core-dns-over-coap`) messages for different name lengths

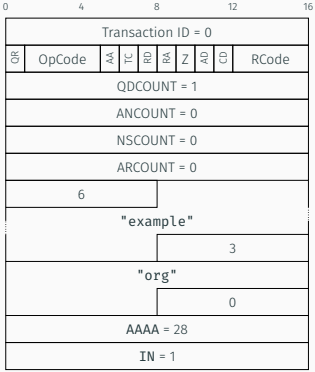
Objectives of draft-lenders-dns-cbor (`application/dns+cbor`)

Provide concise packet format and compressed names and addresses in DNS queries and replies:

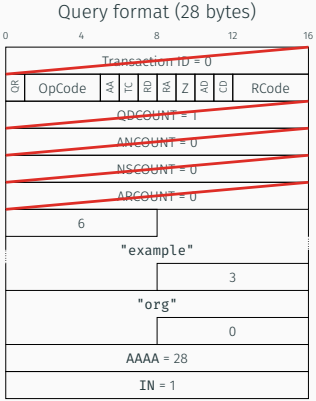
1. Using existing implementation: CBOR
2. Encoding of DNS messages in CBOR (conciseness)
3. Omit (redundant) DNS fields in DNS queries and responses (conciseness)
4. Address and name compression using CBOR-packed (compression, optional)

Omitting DNS fields? Some Examples.

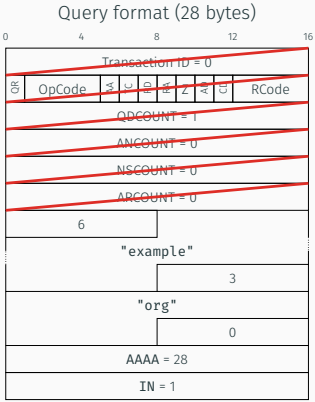
Query format (28 bytes)



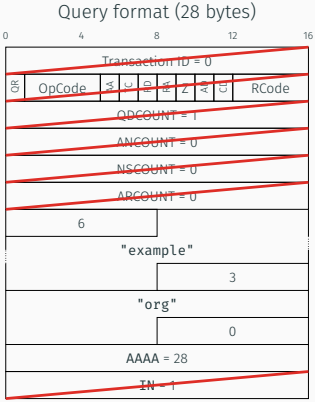
Omitting DNS fields? Some Examples.



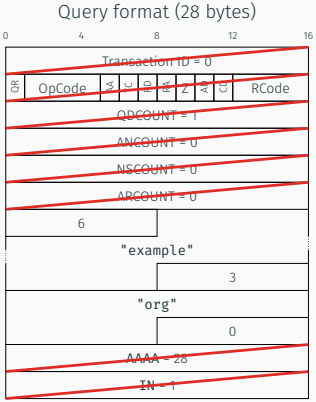
Omitting DNS fields? Some Examples.



Omitting DNS fields? Some Examples.

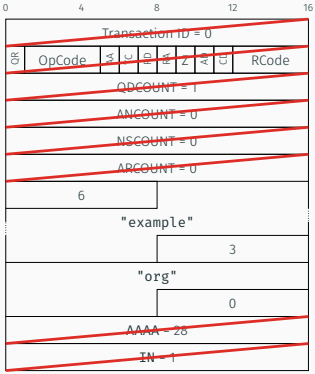


Omitting DNS fields? Some Examples.



Omitting DNS fields? Some Examples.

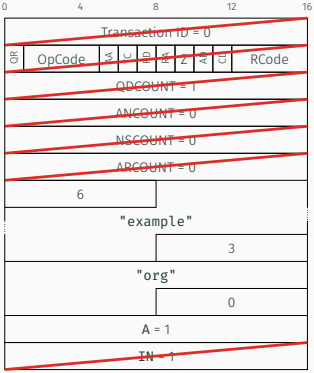
Query format (28 bytes)




```
81 81 6b 6578616d706c652e6f7267 (14 bytes)
[  [  "example.org"]]
```

Omitting DNS fields? Some Examples.

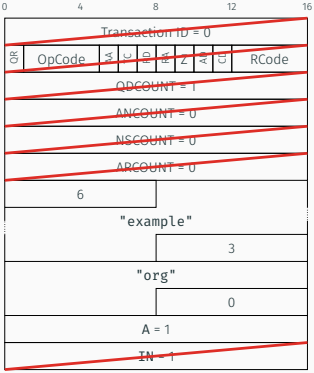
Query format (28 bytes)




```
81 81 6b 6578616d706c652e6f7267 (14 bytes)
[  [  "example.org"]]
```

Omitting DNS fields? Some Examples.

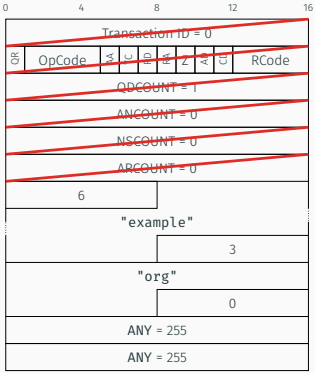
Query format (28 bytes)




```
81 82 6b 6578616d706c652e6f7267 01 (15 bytes)
[  [  "example.org",            1]]
```

Omitting DNS fields? Some Examples.

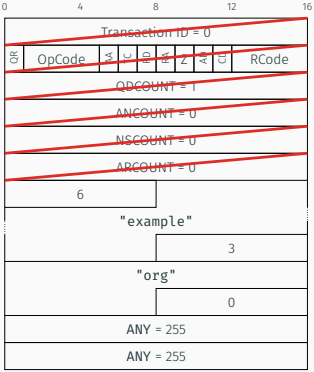
Query format (28 bytes)




```
81 82 6b 6578616d706c652e6f7267 01 (15 bytes)
[  [  "example.org",            1]]
```

Omitting DNS fields? Some Examples.

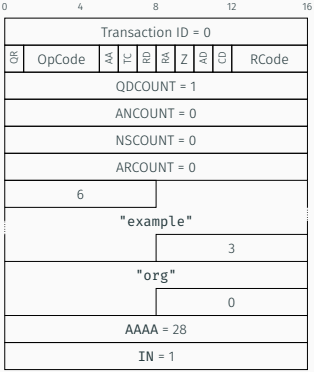
Query format (28 bytes)



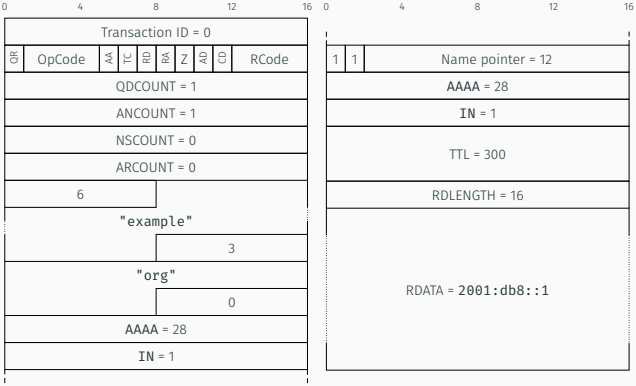

```
81 83 6b 6578616d706c652e6f7267 18 ff 18 ff (18 bytes)
[  [  "example.org",            255,  255]]
```

Omitting DNS fields? Some Examples.

Query format (28 bytes)

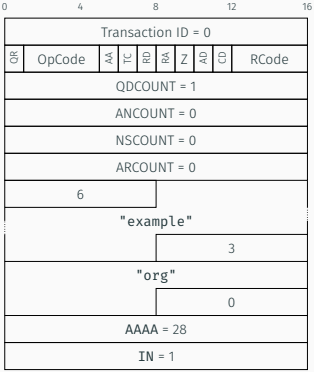


Response format (56 bytes)

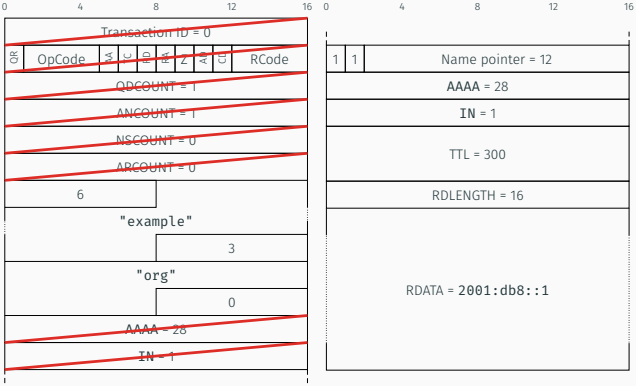


Omitting DNS fields? Some Examples.

Query format (28 bytes)

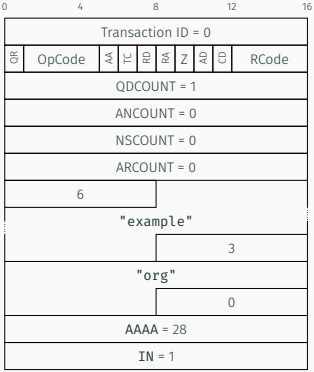


Response format (56 bytes)

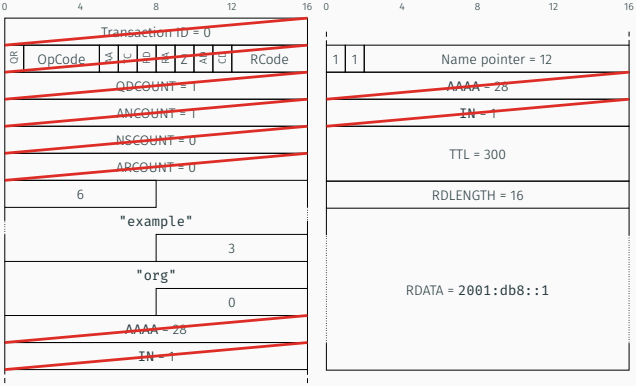


Omitting DNS fields? Some Examples.

Query format (28 bytes)

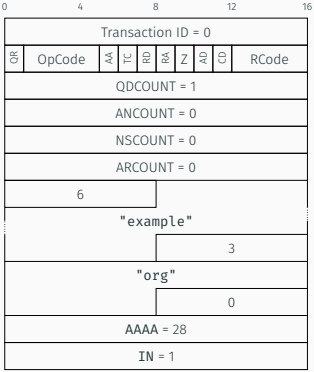


Response format (56 bytes)

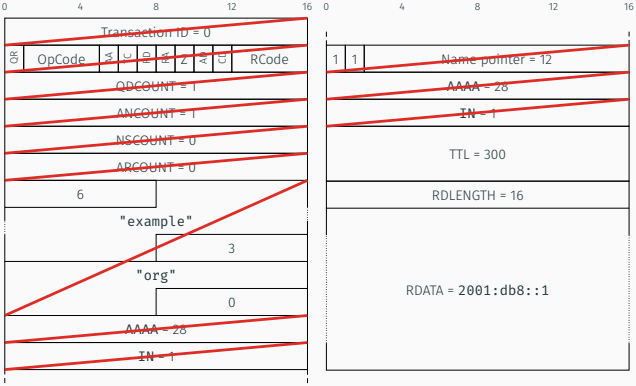


Omitting DNS fields? Some Examples.

Query format (28 bytes)

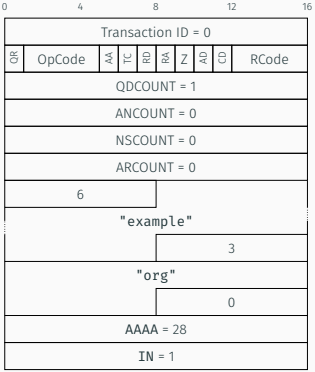


Response format (56 bytes)

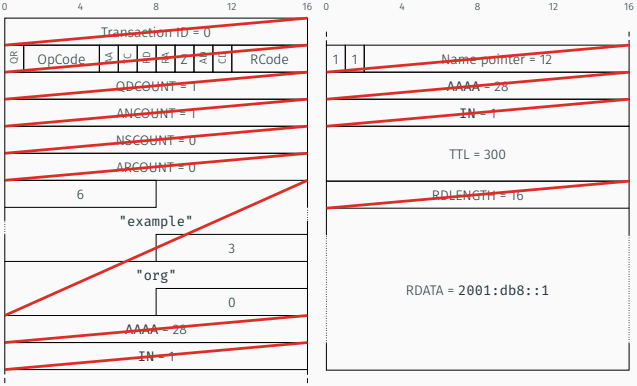


Omitting DNS fields? Some Examples.

Query format (28 bytes)

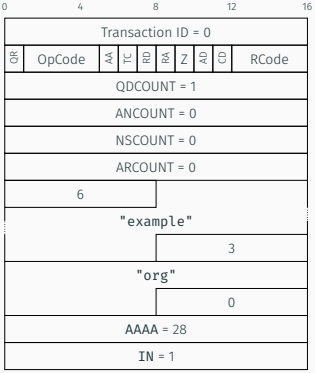


Response format (56 bytes)

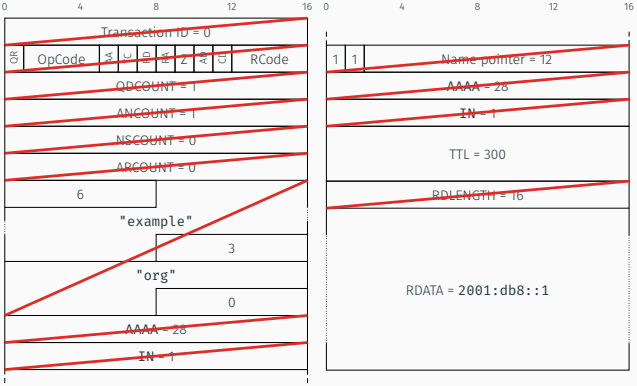


Omitting DNS fields? Some Examples.

Query format (28 bytes)



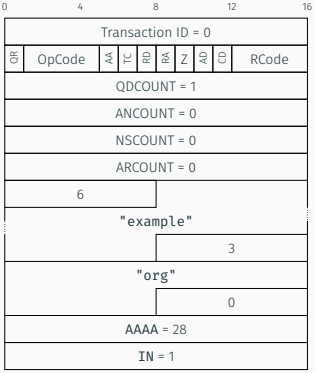
Response format (56 bytes)



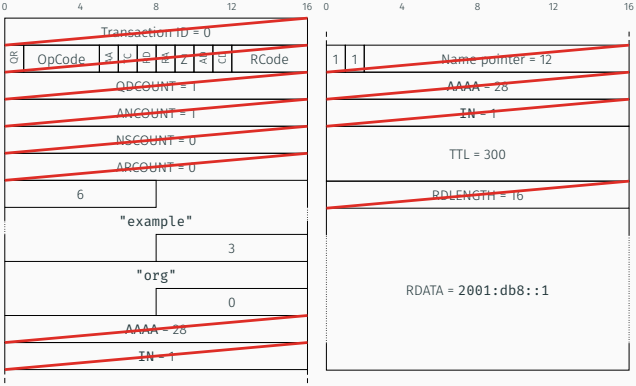

```
81
   81 82 19 012c 50 20010db8000000000000000000000001 (23 bytes)
[
   [  [  300,    h'20010db8000000000000000000000001']]
```

Omitting DNS fields? Some Examples.

Query format (28 bytes)



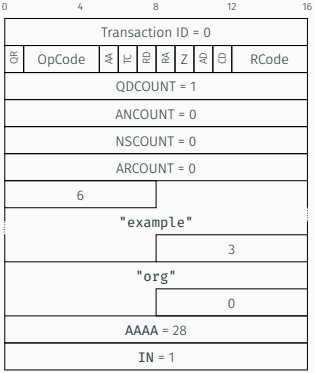
Response format (56 bytes)




```
81
   81 82 19 012c 50 20010db8000000000000000000000001 (23 bytes)
[
   [  [  300,    h'20010db8000000000000000000000001']]
```

Omitting DNS fields? Some Examples.

Query format (28 bytes)



Response format (56 bytes)


```
82
   81 6b 6578616d706c652e6f7267
   81 82 19 012c 50 20010db8000000000000000000000001 (36 bytes)
[
   [  "example.org"],
   [  [  300,    h'20010db8000000000000000000000001']]
```

Changes to DNS+CBOR Draft Since IETF 117 and `-03`

Now at -05

+ Add note on representation of more structured RDATA
+ Provide format description for EDNS OPT Pseudo-RRs
+ Add “Implementation Status” section
- Simplify CDDL to more idiomatic style
- Other clean-up and housekeeping
– Remove DNS transaction IDs
– Remove `int` as representation for `rdata`

EDNS OPT Pseudo-RRs

```
opt-rr = [
  ? udp-payload-size: uint .default 512,
  options: [* opt],
  ? opt-rcode-v-flags,
]
opt = (
  ocode: uint,
  odata: bstr,
)
opt-rcode-v-flags = (
  flags: uint .default 0,
  ? opt-rcode-v,
)
opt-rcode-v = (
  rcode: uint .default 0,
  ? version: uint .default 0,
)
```

Classic DNS format (23 bytes)

```
00 00 29 02 00 00 00 00
00 00 0c 00 0a 00 08 96
0c db 3b 79 af 95 26
```

CBOR (14 bytes)

```
141(                    |d8 8d
 [                      | 81
  [                     |  82
   10,                  |   0a
                        |   48
   h'960cdb3b79af9526'  |    96 0c db 3b
  ]]                    |    79 af 95 26
)                       |
```

Evaluation: Data Corpus

YourThings[1] IoTFinder[2] MonIoTr[3]

- Collected throughout 2019
- 90 consumer IoT devices from 50 vendors
  + Phones, Tablets, PCs, ... (in IoTFinder & YourThings)
- 1.20 million queries
- 2.74 million responses
  - 2.07 million w/o mapped query (IoTFinder contains only responses)
  - 0.66 million w/ mapped query

---

[1] O. Alrawi, C. Lever, M. Antonakakis, and F. Monrose. 2019. **SoK: Security Evaluation of Home-Based IoT Deployments**. In *IEEE S&P 2019*. 1362–1380.

[2] R. Perdisci, T. Papastergiou, O. Alrawi, and M. Antonakakis. 2020. **IoTFinder: Efficient Large-Scale Identification of IoT Devices via Passive DNS Traffic Analysis**. In *IEEE EuroS&P 2020*. 474–489.

[3] J. Ren, D.J. Dubois, D. Choffnes, A.M. Mandalari, R. Kolcun, and H. Haddadi. 2019. **Information Exposure for Consumer IoT Devices: A Multidimensional, Network-Informed Measurement Approach**. In *Proc. of the Internet Measurement Conference (IMC)*. ACM.

Evaluation: Method

- Implementation: `https://github.com/netd-tud/cbor4dns`
- Applied CBOR+DNS to data corpus
  - Elide question section in response if query is present in data corpus
  - For all: unpacked and packed

    unpacked "classic" CBOR+DNS (`application/cbor+dns`)

    packed CBOR-packed+DNS (`application/cbor+dns;packed=1`)

Compression Ratios & Byte Savings

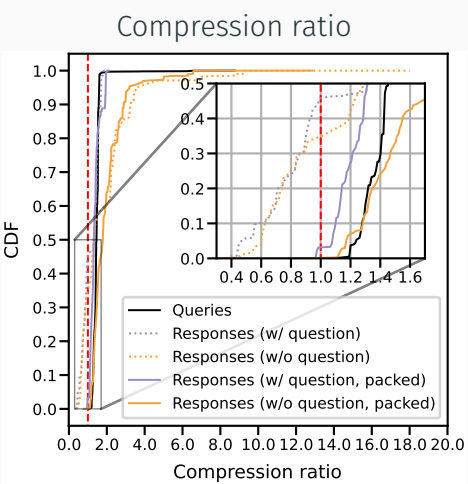
Compression ratio

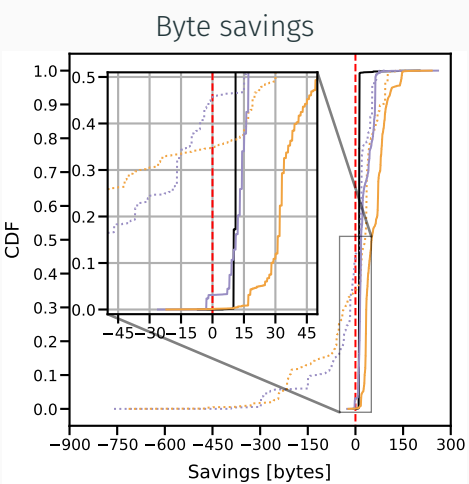
$$\frac{\text{len(classic format)}}{\text{len(CBOR format)}}$$

Byte savings

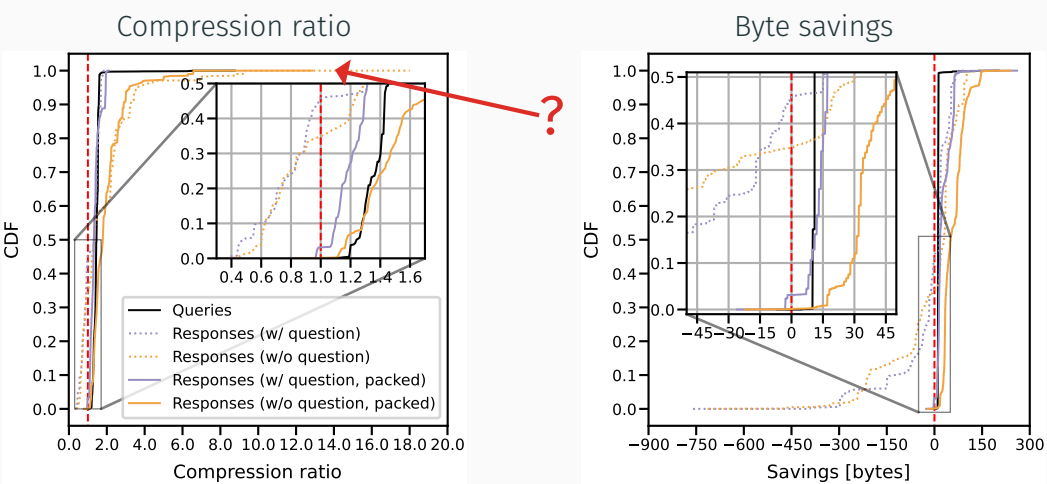
$$\text{len(classic format)} - \text{len(CBOR format)}$$

Compression Ratios & Byte Savings

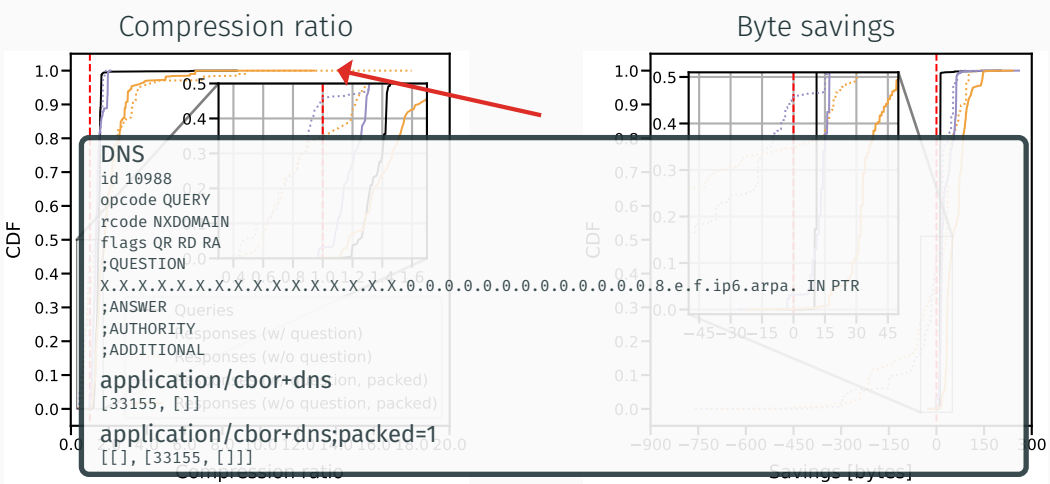




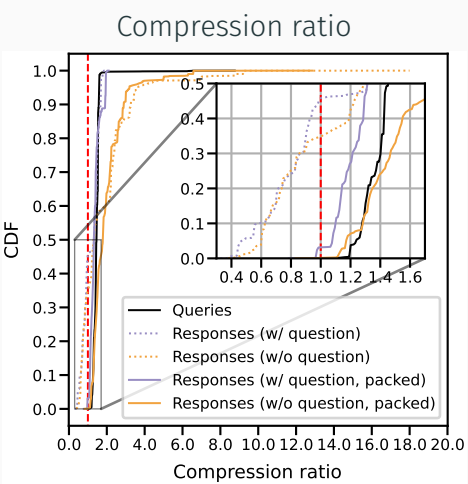
Compression Ratios & Byte Savings

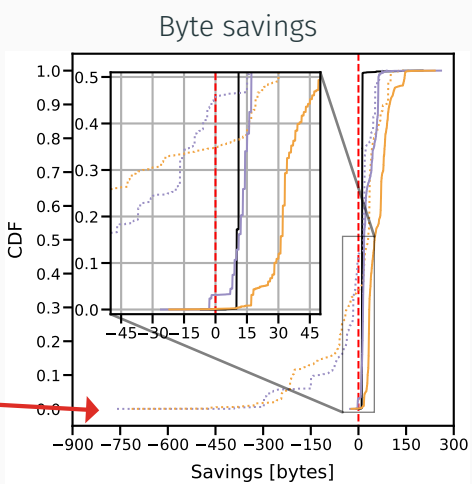


Compression Ratios & Byte Savings



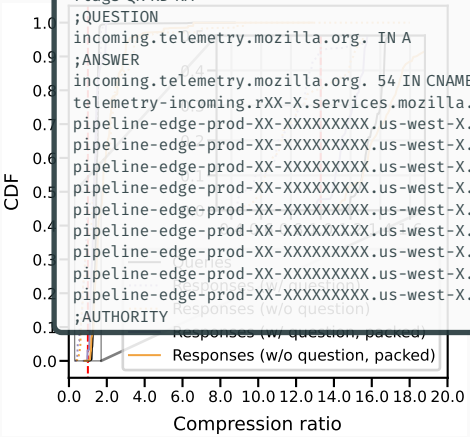
Compression Ratios & Byte Savings

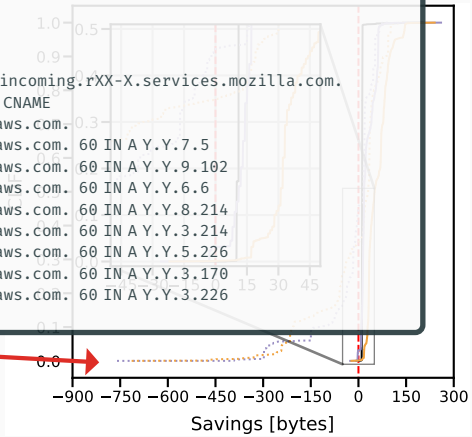




Compression Ratios & Byte Savings

```
DNS
id 63962
opcode QUERY
rcode NOERROR
flags QR RD RA
;QUESTION
incoming.telemetry.mozilla.org. IN A
;ANSWER
incoming.telemetry.mozilla.org. 54 IN CNAME telemetry-incoming.rXX-X.services.mozilla.com.
telemetry-incoming.rXX-X.services.mozilla.com. 299 IN CNAME
pipeline-edge-prod-XX-XXXXXXXXX.us-west-X.elb.amazonaws.com.
pipeline-edge-prod-XX-XXXXXXXXX.us-west-X.elb.amazonaws.com. 60 IN A Y.Y.7.5
pipeline-edge-prod-XX-XXXXXXXXX.us-west-X.elb.amazonaws.com. 60 IN A Y.Y.9.102
pipeline-edge-prod-XX-XXXXXXXXX.us-west-X.elb.amazonaws.com. 60 IN A Y.Y.6.6
pipeline-edge-prod-XX-XXXXXXXXX.us-west-X.elb.amazonaws.com. 60 IN A Y.Y.8.214
pipeline-edge-prod-XX-XXXXXXXXX.us-west-X.elb.amazonaws.com. 60 IN A Y.Y.3.214
pipeline-edge-prod-XX-XXXXXXXXX.us-west-X.elb.amazonaws.com. 60 IN A Y.Y.5.226
pipeline-edge-prod-XX-XXXXXXXXX.us-west-X.elb.amazonaws.com. 60 IN A Y.Y.3.170
pipeline-edge-prod-XX-XXXXXXXXX.us-west-X.elb.amazonaws.com. 60 IN A Y.Y.3.226
;AUTHORITY
```

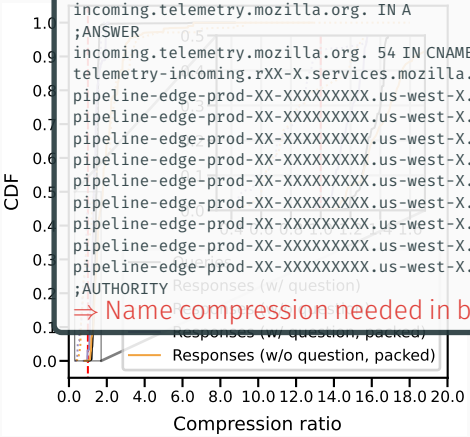


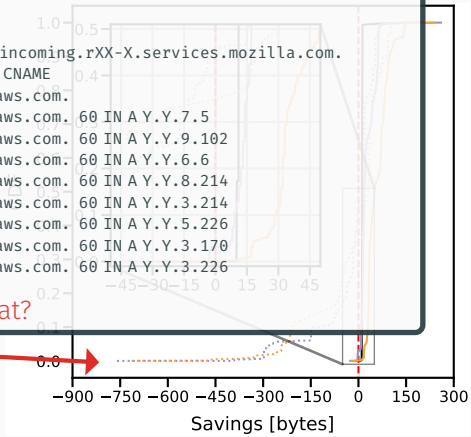


Compression Ratios & Byte Savings

```
DNS
id 63962
opcode QUERY
rcode NOERROR
flags QR RD RA
;QUESTION
incoming.telemetry.mozilla.org. IN A
;ANSWER
incoming.telemetry.mozilla.org. 54 IN CNAME telemetry-incoming.rXX-X.services.mozilla.com.
telemetry-incoming.rXX-X.services.mozilla.com. 299 IN CNAME
pipeline-edge-prod-XX-XXXXXXXXX.us-west-X.elb.amazonaws.com.
pipeline-edge-prod-XX-XXXXXXXXX.us-west-X.elb.amazonaws.com. 60 IN A Y.Y.7.5
pipeline-edge-prod-XX-XXXXXXXXX.us-west-X.elb.amazonaws.com. 60 IN A Y.Y.9.102
pipeline-edge-prod-XX-XXXXXXXXX.us-west-X.elb.amazonaws.com. 60 IN A Y.Y.6.6
pipeline-edge-prod-XX-XXXXXXXXX.us-west-X.elb.amazonaws.com. 60 IN A Y.Y.8.214
pipeline-edge-prod-XX-XXXXXXXXX.us-west-X.elb.amazonaws.com. 60 IN A Y.Y.3.214
pipeline-edge-prod-XX-XXXXXXXXX.us-west-X.elb.amazonaws.com. 60 IN A Y.Y.5.226
pipeline-edge-prod-XX-XXXXXXXXX.us-west-X.elb.amazonaws.com. 60 IN A Y.Y.3.170
pipeline-edge-prod-XX-XXXXXXXXX.us-west-X.elb.amazonaws.com. 60 IN A Y.Y.3.226
;AUTHORITY
```

⇒ Name compression needed in base format?





Possible Choices for Names in Constrained Networks

Keep Uncompressed

Possible Choices for Names in Constrained Networks

Keep Uncompressed

Bespoke DNS solution

Possible Choices for Names in Constrained Networks

Keep Uncompressed

Bespoke DNS solution

Use existing solution (CBOR-packed)

Ideas for Name Compression

Discussed @ 2023-10-04 CBOR Interim:

1. Classic DNS format style: Reference name components within CBOR object
2. CBOR-packed “lite”: Reference name suffixes in pre-defined table

Example Response for Idea Explanations

A PTR (record type 12) request for “example.org”:

```
[ ["example.org", 12, 1],
  [[3600, "_coap._udp.local"]],
  [[3600, 2, "ns1.example.org"], [3600, 2, "ns2.example.org"]],
  [ [ "_coap._udp.local", 3600, h'20010db8000000000000000000000001'],
    [ "ns1.example.org", 3600, h'20010db8000000000000000000000002'],
    [ "ns2.example.org", 3600, h'20010db8000000000000000000003535'] ] ]
```

Idea 1: Reference Name Components

- Loosely based on Christian's idea[1]:
  - Current draft: `domain-name = tstr .regexp "([^.]+[.])*[^.]+"`
  - Proposal: `domain-name = (*comp)`
    (*cf.* `core-href`)
- Use componified name to allow for `tag(uint)` in name, *i.e.*, reference:

  `comp = tstr / #6.`*t*`(uint)`

  To be decided: 1+0 bytes tag (≈2 bytes for reference like DNS) or larger tag

[1] `https://mailarchive.ietf.org/arch/msg/cbor/JOHCCBOzC46PrSq-61MMev--mpU/`

Idea 1: Reference Name Components

- Loosely based on Christian's idea[1]:
  - Current draft: `domain-name = tstr .regexp "([^.]+[.])*[^.]+"`
  - Proposal: `domain-name = (*comp)`
    (*cf.* `core-href`)
- Use componified name to allow for `tag(uint)` in name, *i.e.*, reference:

  `comp = tstr / #6.`*t*`(uint)`

  To be decided: 1+0 bytes tag (≈2 bytes for reference like DNS) or larger tag

Semantics of *t*(*i*) for name construction:

- Start at name components from *i*-th `tstr` in CBOR object
- Stop if object is neither `tstr` nor tag *t* (i.e. not `comp`)
- Follow next *t* until there is nothing to consume

[1] `https://mailarchive.ietf.org/arch/msg/cbor/JOHCCBOzC46PrSq-61MMev--mpU/`

Idea 1: Reference Name Components

- Loosely based on Christian's idea[1]:
  - Current draft: `domain-name = tstr .regexp "([^.]+[.])*[^.]+"`
  - Proposal: `domain-name = (*comp)`
    (*cf.* `core-href`)
- Use componified name to allow for `tag(uint)` in name, *i.e.*, reference:

  `comp = tstr / #6.`*t*`(uint)`

  To be decided: 1+0 bytes tag (≈2 bytes for reference like DNS) or larger tag

Semantics of *t*(*i*) for name construction:

- Start at name components from *i*-th `tstr` in CBOR object
- Stop if object is neither `tstr` nor tag *t* (i.e. not `comp`)
- Follow next *t* until there is nothing to consume

```
[ ["example", "org", 12, 1],
  [[3600, "_coap", "_udp", "local"]],
  [[3600, 2, "ns1", t(0)], [3600, 2, "ns2", t(0)]],
  [ [ t(2), 3600,
      h'20010db8000000000000000000000001'],
    [ t(5), 3600,
      h'20010db8000000000000000000000002'],
    [ t(6), 3600,
      h'20010db8000000000000000000003535']
    ] ]
```

[1] https://mailarchive.ietf.org/arch/msg/cbor/JOHCCBOzC46PrSq-61MMev--mpU/

Idea 1: Reference Name Components

- Loosely based on Ch
  - Current
  - Prop
    (*cf.*
- Use com

To be decided:

"([^.]+[.])*[^.]+"

.e., reference:

uint)

e like DNS) or larger tag

Implicit table

```
/0/ [["example", "org"],
/1/  ["org"],
/2/  ["_coap", "_udp", "local"],
/3/  ["_udp", "local"],
/4/  ["local"],
/5/  ["ns1", t(0)],
/6/  ["ns2", t(0)]]
```

Semantics of $t(i)$ for name construction:

- Start at name components from *i*-th `tstr` in CBOR object
- Stop if object is neither `tstr` nor tag *t* (i.e. not `comp`)
- Follow next *t* until there is nothing to consume

```
[ ["example", "org", 12, 1],
  [[3600, "_coap", "_udp", "local"]],
  [[3600, 2, "ns1", t(0)], [3600, 2, "ns2", t(0)]],
  [ [ t(2), 3600,
      h'20010db8000000000000000000000001'],
    [ t(5), 3600,
      h'20010db8000000000000000000000002'],
    [ t(6), 3600,
      h'20010db8000000000000000000003535']
    ] ]
```

[1] https://mailarchive.ietf.org/arch/msg/cbor/JOHCCBOzC46PrSq-61MMev--mpU/

Idea 1: Reference Name Components

- Loosely based on Christian's idea[1]:
  - Current draft: `domain-name = tstr .regexp "([^.]+[.])*[^.]+"`
  - Proposal: `domain-name = (*comp)`
    (cf. `core-href`)
- Use componified name to allow for `tag(uint)` in name, *i.e.*, reference:
  To be decided: 1+0 bytes tag (≈2 bytes for reference like DNS) or larger tag

Could be seen as a variant of CBOR-packed with

- Implicit table building
- Alternative referencing syntax and semantics

Semantics of *t*(*i*) for name construction:

- Start at n
  in CBOR object
- Stop if object is neither `tstr` nor tag *t* (i.e. not `comp`)
- Follow next *t* until there is nothing to consume

```
[ ["example", "org", 12, 1],
  [[3600, 2, "ns1", t(0)], [3600, 2, "ns2", t(0)]],
  [ t(2), 3600,
    h'20010db8000000000000000000000001'],
  [ t(5), 3600,
    h'20010db8000000000000000000000002'],
  [ t(6), 3600,
    h'20010db8000000000000000000003535']
] ]
```

[1] https://mailarchive.ietf.org/arch/msg/cbor/JOHCCBOzC46PrSq-61MMev--mpU/

Idea 2: CBOR-packed “Lite”

- More lightweight version of proposal by Lemogue et al.
  - Use array instead of dict for `name_ref`
  - Basically `application/cbor+dns;packed=1`
    but only take names into account for table building

Idea 2: CBOR-packed “Lite”

- More lightweight version of proposal by Lemogue et al.
  - Use array instead of dict for `name_ref`
  - Basically `application/cbor+dns;packed=1`
    but only take names into account for table building

```
[ ["example.org", "_coap._udp.local", 216("ns1."), 216("ns2.")]
  [⓪, 12, 1],
  [3600, ①]],
  [[3600, 2, ②], [3600, 2, ③]],
  [ [①, 3600, h'20010db8000000000000000000000001'],
    [②, 3600, h'20010db8000000000000000000000002'],
    [③, 3600, h'20010db8000000000000000000003535'] ] ]
```

(Reminder: Simple value ⓘ = Use table entry *i* as value; `216(tstr)` = Use table entry 0 as suffix for `tstr`)

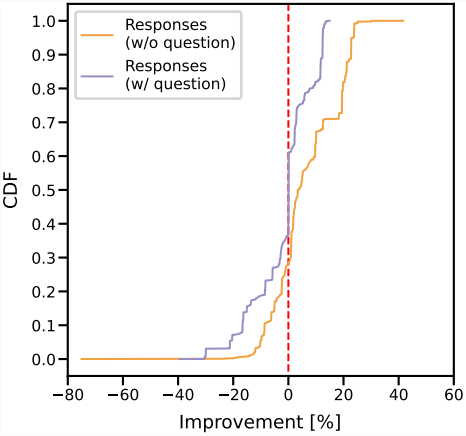
Idea 2: CBOR-packed “Lite”

- More lightweight version of proposal by Lemogue et al.
  - Use array instead of dict for `name_ref`
  - Basically `application/cbor+dns;packed=1`
    but only take names into account for table building

```
        0                1                2           3
[ ["example.org", "_coap._udp.local", 216("ns1."), 216("ns2.")]
  [⓪, 12, 1],
  [3600, ①]],
  [[3600, 2, ②], [3600, 2, ③]],
  [ [①, 3600, h'20010db8000000000000000000000001'],
    [②, 3600, h'20010db8000000000000000000000002'],
    [③, 3600, h'20010db8000000000000000000003535'] ] ]
```

(Reminder: Simple value ⓘ = Use table entry *i* as value; `216(tstr)` = Use table entry 0 as suffix for `tstr`)

Evaluation: Improvement of Packed Lite Over Component Referencing

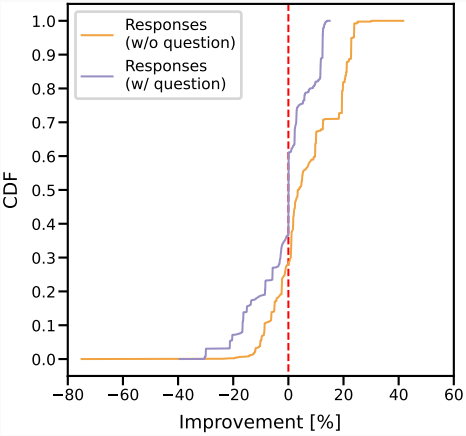


$$\frac{\text{len(comp. ref.) - len(packed lite)}}{\text{len(comp. ref.)}}$$

Using 1+0 tag for component referencing (comp. ref.)

- For $\approx 70\%$ of responses w/o question: $> 0\%$ improvement

Evaluation: Improvement of Packed Lite Over Component Referencing



$$\frac{\text{len(comp. ref.) - len(packed lite)}}{\text{len(comp. ref.)}}$$

Using 1+0 tag for component referencing (comp. ref.)

- For $\approx 70\%$ of responses w/o question: $> 0\%$ improvement
- For improvement $< 0\%$: Mostly overhead from extra list

Conclusion

Name compression leads compression ratio >100% for nearly all responses

Benefits of *component referencing*

- *Conciseness*
- *Simplicity*
- *Familiarity*

Benefits of *packed lite*

- *Consistency*
- *Efficiency*

Conclusion

Name compression leads compression ratio >100% for nearly all responses

Benefits of *component referencing*

- *Conciseness* No added byte overhead due to extra list
- *Simplicity*
- *Familiarity*

Benefits of *packed lite*

- *Consistency*
- *Efficiency*

Conclusion

Name compression leads compression ratio >100% for nearly all responses

Benefits of *component referencing*

- *Conciseness* No added byte overhead due to extra list
- *Simplicity* Encoding easy to implement
- *Familiarity*

Benefits of *packed lite*

- *Consistency*
- *Efficiency*

Conclusion

Name compression leads compression ratio >100% for nearly all responses

Benefits of *component referencing*

- *Conciseness* No added byte overhead due to extra list
- *Simplicity* Encoding easy to implement
- *Familiarity* More similar to classic DNS name compression

Benefits of *packed lite*

- *Consistency*
- *Efficiency*

Conclusion

Name compression leads compression ratio >100% for nearly all responses

Benefits of *component referencing*

- *Conciseness* No added byte overhead due to extra list
- *Simplicity* Encoding easy to implement
- *Familiarity* More similar to classic DNS name compression

Benefits of *packed lite*

- *Consistency* No new reference syntax, no new table building, just CBOR-packed
- *Efficiency*

Conclusion

Name compression leads compression ratio >100% for nearly all responses

Benefits of *component referencing*

- *Conciseness* No added byte overhead due to extra list
- *Simplicity* Encoding easy to implement
- *Familiarity* More similar to classic DNS name compression

Benefits of *packed lite*

- *Consistency* No new reference syntax, no new table building, just CBOR-packed
- *Efficiency* More responses made smaller compared to component referencing

Which way should we go?

Any other ideas?
Implicit tables for CBOR-packed in general?