New CoAP Block-Wise Transfer Options For Faster Transmission

draft-ietf-core-new-block-01

IETF CoRE Meeting, 22nd Oct 2020

Mohamed Boucadair

Jon Shallow

Agenda

- Requirements Reminder
- -01 Updates
- Implementation Observations
- Next Steps

Reminder

- Fast Transmission of data
  - Subject to Congestion Control
- Block transmission loss recovery
- Unidirectional NON Blocks support
- Handle unidirectional traffic loss
- Modelled on Block1 / Block2
- Addition to, not replacement for, Block1 / Block2

Updates in -01 (09/2020)

- Updated the Applicability Scope
- Removed the TBA3 (Missing Payloads), using 4.08 instead
- Renamed the options to Quick-Block1/2
  - The options are marked as unsafe
  - The caching behaviour is updated
- Moved the CC text to a (new) dedicated section
- Avoided the normative language for the usage of Tokens
  A new (short) section is added for the token discussion
- Other edits to enhance the readability of the document

Implementation Approach

- Using libcoap
  - PR #554 raised to move all Block1/2 handling into libcoap instead of being done in the application
  - Have additional libcoap code to support Quick-Block1/2 leveraging on #554
    - Code will become a PR at some point
- Some observations from the implementation work are discussed next

Mutual Support

- Support of both Quick-Block1/2
  - Currently *independently* supported
  - Makes tracking of which is supported more difficult
    - Especially if both of them are sent in a request

- RFC 7252 critical option reporting is unclear
  - CON 4.02 diagnostic payload (formatted how?)

    "*This response SHOULD include a diagnostic payload describing the unrecognized option(s)*“ (Section 5.4.1)
  - libcoap returns bad critical option as an option
  - NON returns RST

- **Suggestion**: Recommend that either both Quick-Block1/2 supported or neither
  - Thoughts?

Congestion Control

- MAX_PAYLOADS (default every 10 packets)
  - Default wait of ACK_TIMEOUT before proceeding
  - Use of CON every MAX_PAYLOAD for reduction of turnaround times
  - CON fails if unidirectional traffic loss
  - NON will wait for ACK_TIMEOUT before next packet sent
- **Issue**: NON reduction of turnaround times
  - Cannot assume MAX_PAYLOADS is same at both ends for trigger
- **Suggestion**: NON: Signal something in the MAX_PAYLOAD packet to indicate immediate acknowledge response required
  - if response fails to get through there still will be ACK_TIMEOUT wait which is OK
- **Question**: What to add?
  - Update the Quick-Block option format to "NUM **R** M SZX" where R bit set means:
    - Quick-Block1 – Respond with 2.31
    - Quick-Block2 – Issue GET for next block

Quick-Block2 Implementation

- Quick-Block2 was the easiest to implement
  - Size required for missing blocks (as options) difficult to compute
  - Dependent on ‘block.num’ value
  - Will they all fit into a request packet

- **Suggestion**: Limit the number of missing Quick-Block2 options to MAX_PAYLOADS. This also then ties in nicely with what a server can send at once
  - Thoughts?

Quick-Block1 implementation (1)

- CDDL for 4.08 response payload
  - Struggled to get this right for CBOR only
  - Now have how it should be defined (thanks Carsten)
- Had to add limited CBOR knowledge to libcoap

Quick-Block1 implementation (2)

- There are lot of Tokens to track
  - MAX_PAYLOAD of Tokens at a time
  - Last packet of MAX_PAYLOAD may not arrive
    - Which Token should be used for a failure response?

- **Suggestion**: There is an “associated response” a.k.a., Observer. Is it worth considering an “associated request” for Quick-Block1 where all the Tokens are the same?
  - Any request retry for missing Quick-Block1s would have a different Token

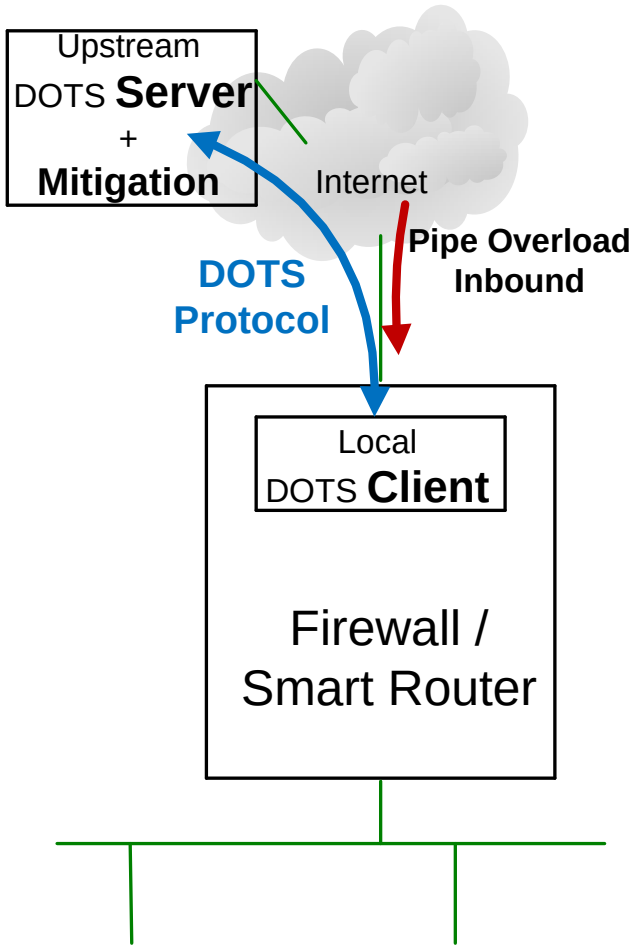
Next Steps

- Prepare -02 with the outcome of the discussion
  - Milestone: to be ready for IETF#109
- Update the implementation
- If no major issue, target a WGLC

- Please review and share comments:
  https://github.com/core-wg/new-block

Thank You

Sample Target Deployment



- DDoS Open Threat Signalling (DOTS)
- DOTS: App – CBOR – CoAP – DTLS – IP
- Client requests mitigation (NON)
- Server updates with simple DOTS mitigation status (NON)
- Inbound Pipe Overload
  - Clients can still request mitigations
  - Mitigation should be able to control pipe overload
- See RFC8782 for more details