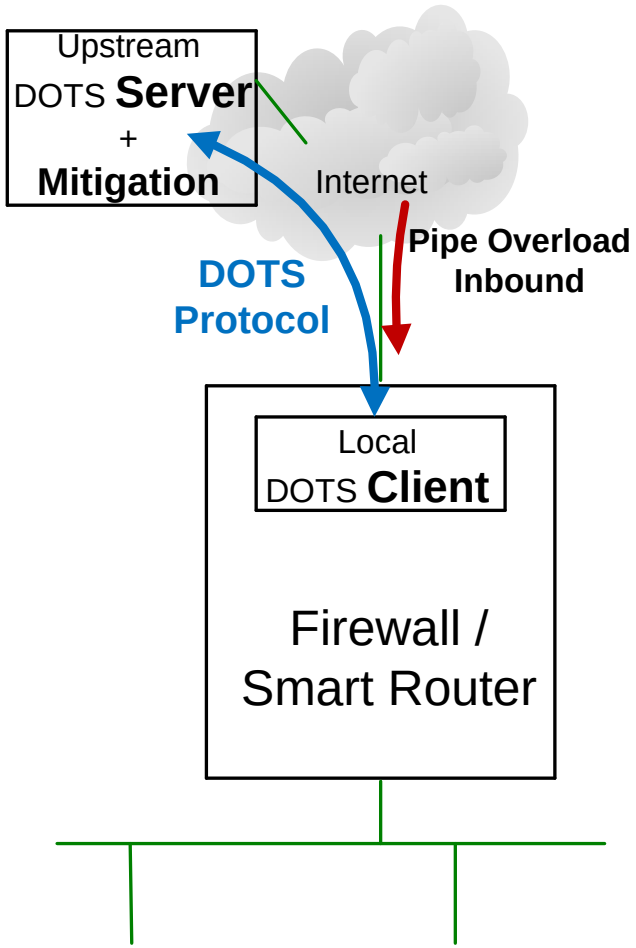
# New CoAP Block-Wise Transfer Options draft-bosh-core-new-block

CoRE virtual interim, 10th June 2020

**Mohamed Boucadair**

**Jon Shallow**

# 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

# Summary of Updates Since -00

Congestion Control

- Applicability Scope added
- Guards to prevent a CoAP agent from overloading the network
  - PROBING_RATE clarification
  - MAX_PAYLOADS defined, with a default value of 10

Protocol Machinery

- Detailed description of Block3/Block4 Option
  - Block3 no longer repeatable
  - New CoAP Response Code for missing blocks
- Block3/Block4 RFC8613 definitions included
- Text tidy up

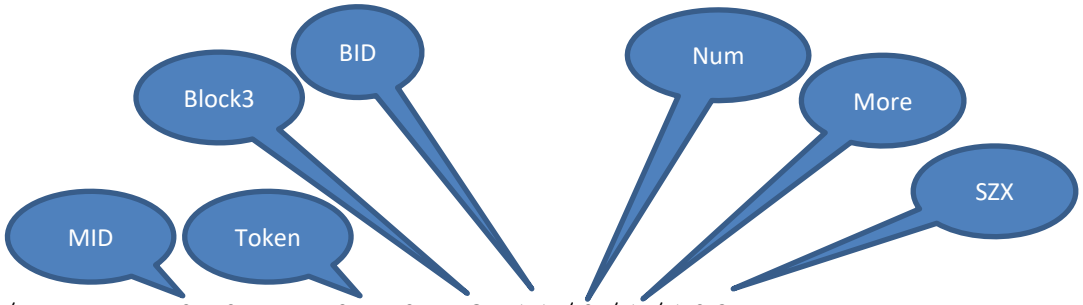
# Block3 Updates (1 of 2)

- ‘A’ Bit removed. Replaced by ‘U’ bit for future use
- Block ID (BID)
  - Need to differentiate between different PUTs (with different body) to same resource
    - To handle failure conditions
  - Unable to use Token here as alternative as each request must have a different token
  - Cannot have a value of 0 (Block4 special case)
- Tokens must not be empty
  - Each PUT has different Token
  - Error responses for particular block can be handled

# Block3 Updates (2 of 2)

- Partial body clean up comments
- 4.08 (Request Entity Not Complete) not recommended
  - Blocks may arrive out of order (no longer “lock stepping”)
- New TBA3 (4.18) (Missing Payloads) instead of 4.08
  - Indicates missing blocks in response payload
    - CBOR encoded – count + list of missing block numbers
- 2.31 (Continue) not used

# Block3: An Example

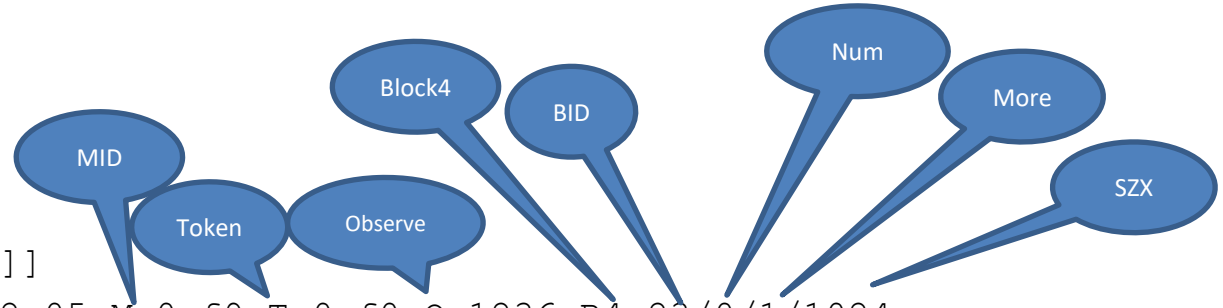



```
 CoAP          CoAP
Client        Server
  |             |
  +---------->| NON PUT /path M:0x05 T:0xe0 B3:11/0/1/1024
  +--->X      | NON PUT /path M:0x06 T:0xe1 B3:11/1/1/1024
  +--->X      | NON PUT /path M:0x07 T:0xe2 B3:11/2/1/1024
  +---------->| NON PUT /path M:0x08 T:0xe3 B3:11/3/0/1024
[[Server realizes missing blocks and indicates this]]
  |<----------+ NON 4.18 M:0xf2 T:0xe3 [Missing 1,2]
  +---------->| NON PUT /path M:0x09 T:0xe4 B3:11/1/1/1024
  +--->X      | NON PUT /path M:0x0a T:0xe5 B3:11/2/1/1024
[[Server requests final missing block]]
  |<----------+ NON 4.18 M:0xf3 T:0xe4 [Missing 2]
  +---------->| NON PUT /path M:0x0b T:0xe6 B3:11/2/1/1024
  |<----------+ NON 2.04 M:0xf4 T:0xe6
```

# Block4 Updates

- ‘A’ Bit removed. Replaced by ‘U’ bit for future use
- Block ID (BID)
  - Needed to indicate which “body” is missing some blocks
  - Cannot use ETag as an alternative here
    - 2.03 (Valid) response usage conflicts here
  - Each “body” response must have a non zero BID value
    - Random initial value – MUST be different per “body”
  - Can only have a value of 0 when requesting all blocks of (new) “body”

# Block4 Example




```
 CoAP          CoAP
Client        Server
  |             |
      ...
[[Observe triggered]]
  |<----------+ NON 2.05 M:0xf9 T:0xf0 O:1236 B4:23/0/1/1024
  |     X<---+ NON 2.05 M:0xfa T:0xf0 O:1236 B4:23/1/1/1024
  |     X<---+ NON 2.05 M:0xfb T:0xf0 O:1236 B4:23/2/1/1024
  |<----------+ NON 2.05 M:0xfc T:0xf0 O:1236 B4:23/3/0/1024
[[Client realizes blocks are missing and asks for the missing
    ones in one go]]
  +---------->| NON GET /path M:0x02 T:0xf1 B4:23/1/0/1024 \
  |           |                             B4:23/2/0/1024
  |     X<---+ NON 2.05 M:0xfd T:0xf1 B4:23/1/1/1024
  |<----------+ NON 2.05 M:0xfe T:0xf1 B4:23/2/1/1024
[[Client gets final missing block]]
  +---------->| NON GET /path M:0x03 T:0xf2 B4:23/1/0/1024
  |<----------+ NON 2.05 M:0xff T:0xf2 B4:23/1/1/1024
```

# Next Steps

- Further discussion
- Consider adopting as a WG Document

# Thank You