New CoAP Block-Wise Transfer Options

draft-bosh-core-new-block

CoRE virtual interim 13th May 2020

Mohamed Boucadair

Jon Shallow
DOTS Use Case Example Environment

- DDoS Open Threat Signalling (DOTS)
- 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
DOITS General Operation

• Configuration
  – Confirmable
  – Peace Time

• Mitigation Requests / Responses
  – Non Confirmable
  – Single Packets contain all the information
  – Works with response packet loss
    • E.g. Request Mitigate traffic to IP W.X.Y.Z/32
    • Status updates may get lost

• Application Heartbeats
  – Non Confirmable
  – Initiated separately by Client and Server
  – Server can detect Client alive at all times
  – Client continues, even if no Server traffic seen
DOTS Telemetry

- DDoS Telemetry information both ways
  - (Smart) Client -> Server (PUT)
  - Server -> Client (GET)
- Data likely larger than Single Packet
- Without Packet Loss
  - BLOCK1 and BLOCK2 fine (Non Confirmable)
- With Packet Loss (usually Server -> Client)
  - Next BLOCK1 response lost
  - Next BLOCK2 packet request lost
  - All stalls – even when using Non Confirmable
Oversized Packet Handling

• Use IP Fragmentation
  – Requires large receipt buffers
  – Unable to recover missing fragments

• Application break up data into Chunks
  – YANG <anydata> requires chunk to be full JSON as per RFC7951
  – How to break data down to minimize no of chunks

• Use BLOCK1 and BLOCK2: Has limitations
  – Performance (symmetric traffic requires ‘ACK’ before next block is sent)
  – Handling lossy environments
CoAP Options BLOCK3 and BLOCK4

• Same as BLOCK1 and BLOCK2 with additions

• All Blocks sent before ‘ACK’ required
  – Similar to using fragmented IP packets
  – NSTART needs to be increased if CONfirmable

• Missing Blocks can be re-requested

• Each set of Blocks have same Block ID (BID) for re-assembly
  – Could use ETag for BID, but RFC7252 says:
    "An entity-tag is intended for use as a resource-local identifier for differentiating between representations of the same resource"
BLOCK1 vs. BLOCK3

• BLOCK1
  – If NON and no response, limited to PROBING_RATE (1 Byte/sec)

• BLOCK3
  – “Body” of data subject to PROBING_RATE
    • Higher transmit rate for “body” with multiple blocks as all sent with no waiting

• Both can utilize 4.08 for missing blocks

• 4.08 needs to be extended to include array of missing blocks in response (using repeat option with BLOCK3?)
BLOCK2 vs. BLOCK4

• BLOCK2
  – Server has to wait for next block request
  – Copy of “body” maintained for EXCHANGE_LIFETIME

• BLOCK4
  – Entire set of Blocks for “body” can be sent without waiting
  – Higher performance (negligible waits between blocks arriving at Client)
  – A Client can indicate multiple blocks are missing
  – Server can ‘delete’ “body” on successful receipt
  – Caches can keep data at Block and / or “body” level
BLOCK3 & BLOCK4 Tokens

• How should Tokens be handled
  – Set of Block4 responses (same BID) – tokens all the same?
  – Affect on Proxies

• RFC7252 5.4.1:
  “The Token is used to match a response with a request.”
  “A token is intended for use as a client-local identifier”

• RFC7641 4.2:
  “Each such notification response (including the initial response) MUST echo the token specified by the client in the GET request.”

• RFC7959 3.4:
  “requests for additional blocks cannot make use of the token of the Observation relationship”
Next Steps

• RFC 8613 OSCORE implications
• Further discussion

Thank You
Appendix
Example of Mitigation Status with Telemetry

```json
{
   "ietf-dots-signal-channel:mitigation-scope": {
      "scope": [
         {
            "mid": 12332,
            "mitigation-start": "1507818434",
            "alias-name": [
               "https1",
               "https2"
            ],
            "lifetime": 1600,
            "status": "attack-successfully-mitigated",
            "bytes-dropped": "134334555",
            "bps-dropped": "43344",
            "pkts-dropped": "333334444",
            "pps-dropped": "432432",
            "ietf-dots-telemetry:total-attack-traffic": [
               {
                  "ietf-dots-telemetry:unit": "megabit-ps",
                  "ietf-dots-telemetry:mid-percentile-g": "900"
               }
            ],
            "ietf-dots-telemetry:attack-detail": [
               {
                  "ietf-dots-telemetry:vendor-id": 1234,
                  "ietf-dots-telemetry:attack-id": 77,
                  "ietf-dots-telemetry:source-count": {
                     "ietf-dots-telemetry:peak-g": "10000"
                  }
               }
            ]
         }
      ]
   }
}```
Example of DOTS Telemetry

```json
{
    "ietf-dots-telemetry:telemetry": {
        "pre-or-ongoing-mitigation": [
            {
                "tmid": 123,
                "target": {
                    "target-prefix": [
                        "2001:db8::1/128"
                    ],
                    "target-protocol": [17]
                },
                "target-protocol": [17],
                "total-attack-traffic": [
                    {
                        "unit": "megabit-ps",
                        "mid-percentile-g": "900"
                    }
                ],
                "attack-detail": [
                    {
                        "vendor-id": 1234,
                        "attack-id": 77,
                        "start-time": "1957818434",
                        "attack-severity": "high"
                    }
                ]
            }
        ]
    }
}
```