Block wise option

제한된 환경이 이러한 제약을 요구한다.

3가지 인데,, (이 부분을 reducing 하지 않으면 그 CoRE 환경과 충돌이 된다.)

1. Code size 줄이기 위해 실행 복잡도를 감소
2. Fragments 수, transmission power, load를 줄이기 위해서 message size를 감소
3. ?

배경: fragmentation/reassembly process는 lower layer에게 conversation state를 줌으로써 짐을 지운다(burden). 이러한 conversation state는 application layer에서 더 잘 managed 된다. 그래서 adaptation layer, ip layer에서 fragmentation 하는 게 아니고 application layer에서 처리하도록 block 옵션이 생기게 되었다.

장점

1. Constrained network link layer packet보다 더 큰 패킷들을 smaller block으로 만들어서 transfer 할 때
2. Adaptation layer, IP layer fragmentation에서 생기는 관리하기 어려운 conversation state가 없다.
3. Block은 인지되어진다, 필요하다면(못 인지한 경우) retransmission 가능
4. 양 sides에서는 block size를 결정할 수 있다.
5. Resulting exchanges는 packet analyzer tools를 사용해서 이해하기 쉽고, 따라서 debugging 쉽다.
6. 필요 시, resource representation 내의 2의 승수 sized block에 random access 할 수 있다(?)

* Datagram size를 제한하는 것은 좋은 이유들이 있다
  + 최대 데이터 사이즈 (~64KiB for UDP)를 정한다
  + IP fragmentation을 피한다. (IPv6 1280 MTU)
  + Adaptation layer fragmentation을 피한다. (60-80 bytes for 6LoWPAN)
* Payload가 request나 response에 둘 다 존재할 수 있기 때문에 separate한 블락 옵션 존재한다.
  + 1번: request
  + 2번: response
* Term
  + Payload: 한 개 CoAP message의 actual content
  + Body: block-wise로 전송되는 전체 resource representation
* 대부분, 마지막 block을 제외하고는 같은 사이즈로 전송된다.
* Protocol에 의해 block size가 fix 되어있지 않다.
* 2\*\*4 ~ 2\*\*10 bytes 사이 이다.
* 마지막 block이 페이로드로 꽉 차지 않더라도 블락 사이즈는 똑같이 표시해줘야 한다.
* Request에 Block1이 있거나 Response에 Block2가 있다는 것은 의미한다.
  + Block-wise transfer 라는 것
  + Entire body를 어떻게 이 block-wise가 구성하는지 방법 (? – 순서인가)
* Block1이 response에 있거나 block2가 request에 있다는 것은 의미한다. (반대로)
  + 어떻게 payload가 구성되어 지는지
  + 어떻게 payload를 처리하는지

이 2가지에 대한 additional control를 제공한다.

Payload가 request에 달려있냐 response에 달려있냐에 따라 block 1, 2가 결정됨

(Block 1 으로 request보내면 block 1으로 응답..)

* Coap message에서 Block option은 processed 되어야만 한다. (아니면 reject해야함)
* 따라서 critical option임 (option 보는 것이 중요함)