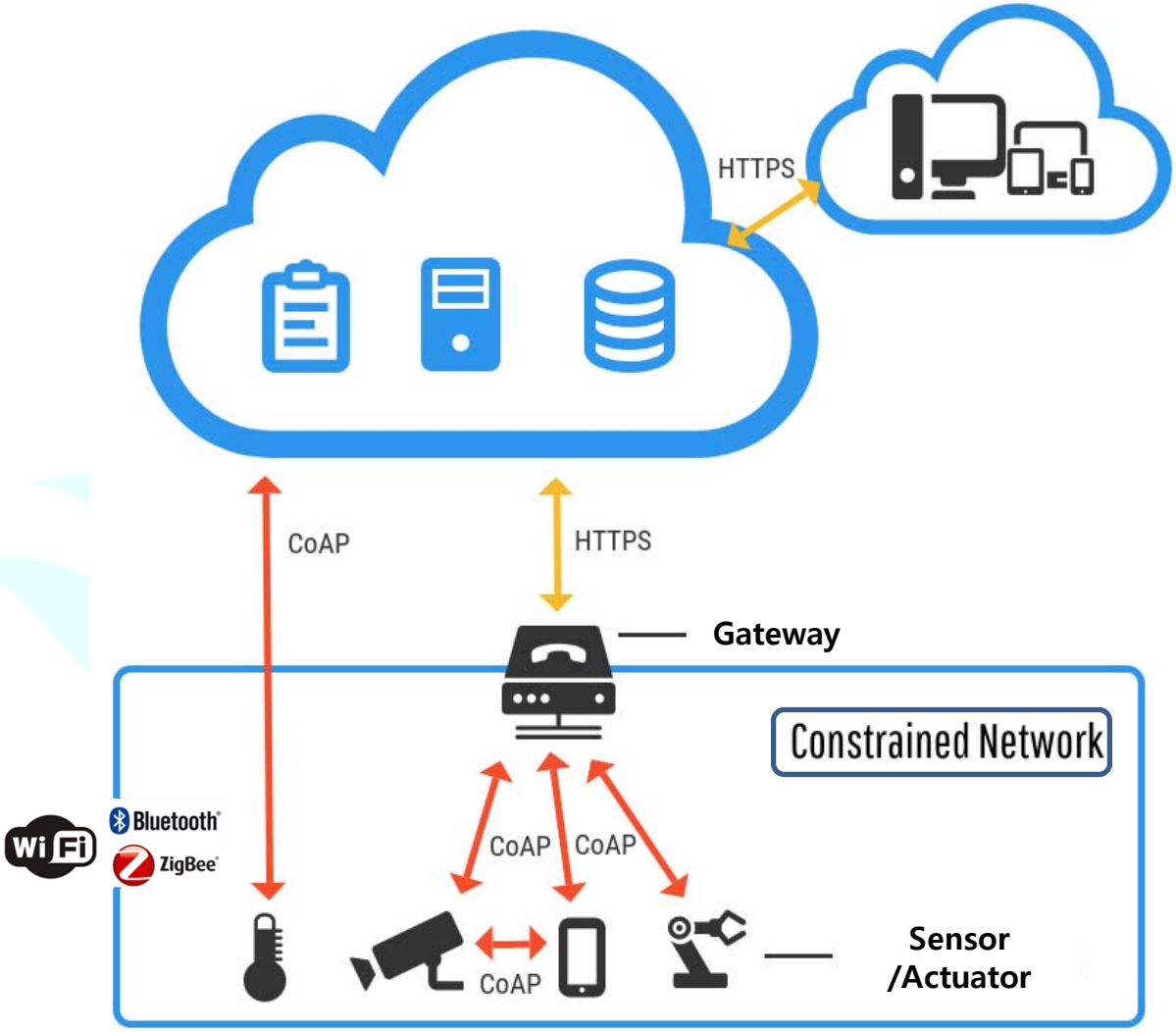
Constrained Application Protocol (1)

Kim, Eui-Jik

Contents

- IoT Network
- Constrained Application Protocol
- Messaging Model

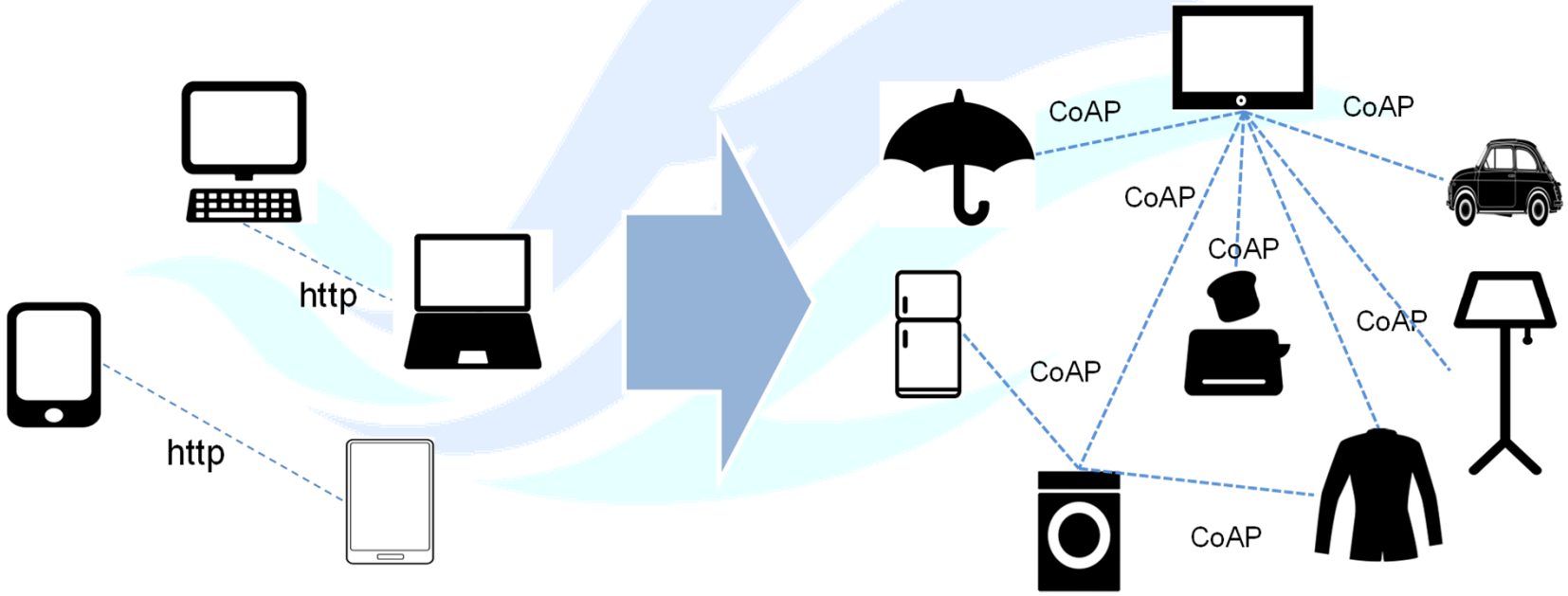
IoT Network



HALLYM UNIVERSITY

Constrained Application Protocol

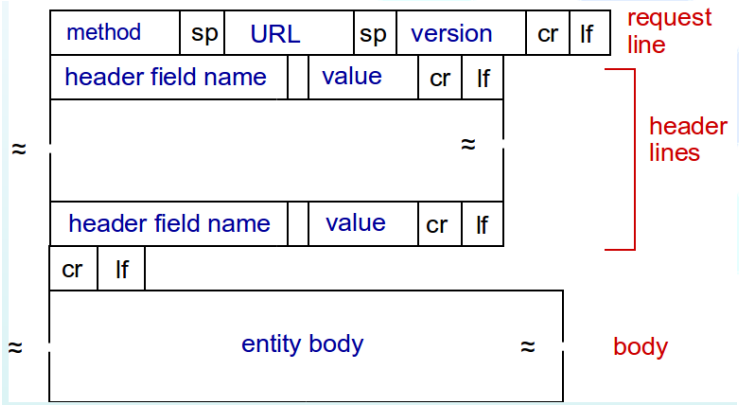
- 우리가 웹 공간에서 HTTP (Hypertext Transfer Protocol)라는 프로토콜로 정보를 교환할 수 있는 것처럼 사물 인터넷(Internet of Things) 시대에는 우리 주변의 다양한 사물들도 CoAP(Constrained Application Protocol)이라는 프로토콜을 통해 정보를 교환



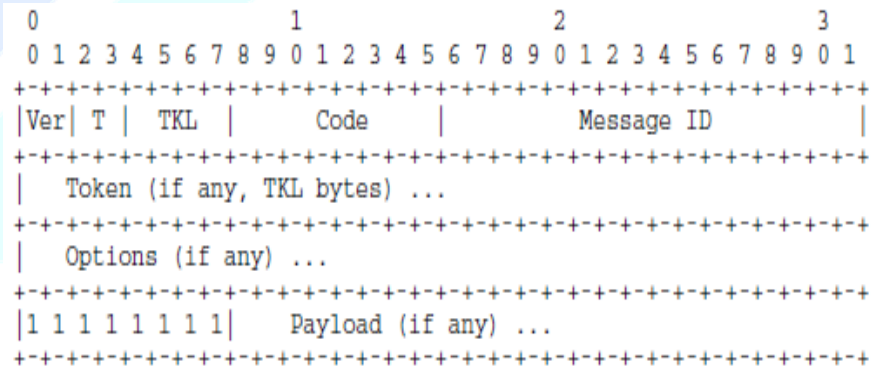
Constrained Application Protocol

- **왜 HTTP가 아닌 별도의 프로토콜이 필요할까?**
  - IoT 환경에서는 저비용, 저사양 센서 노드들의 사용을 고려하기 때문에, 노드들은 작은 용량의 메모리 및 낮은 처리량의 프로세서를 가짐
  - 속도가 빠른 Ethernet이나 Wi-Fi를 뿐 아니라 저속 (IEEE 802.15.4 등) 무선 통신장치나 잡음이 굉장히 심한 유선을 통해서도 통신

HTTP의 큰 헤더 사이즈와 TCP 기반 통신은 IoT환경에 적합하지 않음



HTTP Message format



CoAP Message format

Constrained Application Protocol

- **CoAP**
  - CoAP은 전송지연과 패킷 손실률이 높은 네트워크 환경에서 저사양의 하드웨어로 동작되는 센서 디바이스의 RESTful 웹 서비스를 지원하기 위한 경량 (Lightweight) 프로토콜 (RFC 7252)

REST: REpresentational State Transfer (표현 가능한 상태 전송)

- 자원을 이름(자원(resource)의 표현(representation))으로 구분하여, 해당 자원의 상태(정보)를 주고 받는 것을 의미, ROA(Resource Oriented Architecture) 웹 서비스 아키텍처
- HTTP URI(Uniform Resource Identifier)를 통해 자원을 명시하고, HTTP Method(POST, GET, PUT, DELETE)를 통해 해당 자원에 대한 CRUD operation을 적용함
  - Create: 생성(POST), Read: 조회(GET), Update: 수정(PUT), Delete: 삭제(DELETE) 메소드를 사용하여 데이터를 교환
- REST 원리를 따르는 시스템은 'RESTful'이란 용어로 지칭

Constrained Application Protocol

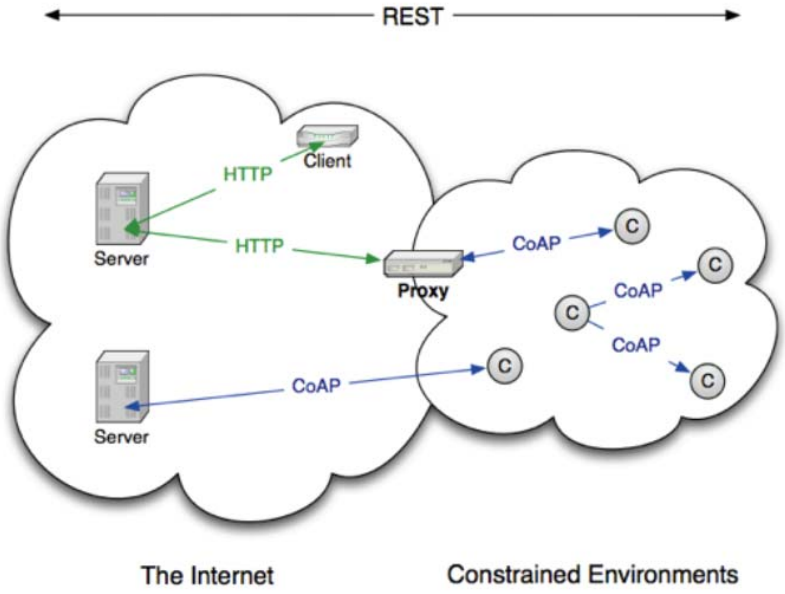
- **REST example**
  - REST는 크게 Resource, Method, Message 3가지 요소로 구성
    - 예를 들어서 "이름이 Terry인 사용자를 생성한다" 라는 호출이 있을 때
      - "사용자"는 생성되는 Resource
      - "생성한다" 라는 행위는 Method
      - '이름이 Terry인 사용자'는 Message
  - 이를 REST 형태로 표현해보면

```
HTTP POST , http://myweb/users/
{
   "users":{
      "name":"Terry"
   }
}
```

    - '사용자'라는 Resource는 http://myweb/users 라는 형태의 URI로 표현
    - '생성한다'의 의미를 갖는 Method는 HTTP Post Method
    - Message는 "name":"Terry" 라 표현

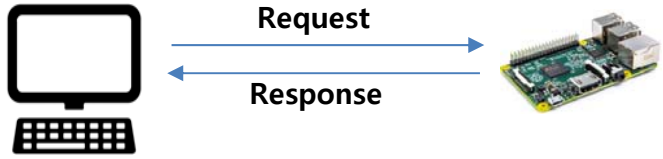
Constrained Application Protocol

- **CoAP의 특징**
  - 제한적인 환경에서의 사물간 통신 지원
  - 선택적으로 신뢰성이 있는 UDP 기반 Unicast 및 Multicast를 지원
  - 비동기적 Message 교환
  - 낮은 헤더의 오버헤드와 복잡도
  - URI 형식의 자원 표현
  - 간단한 Proxy와 Caching을 지원
  - HTTP와 CoAP간 쉬운 Protocol 전환

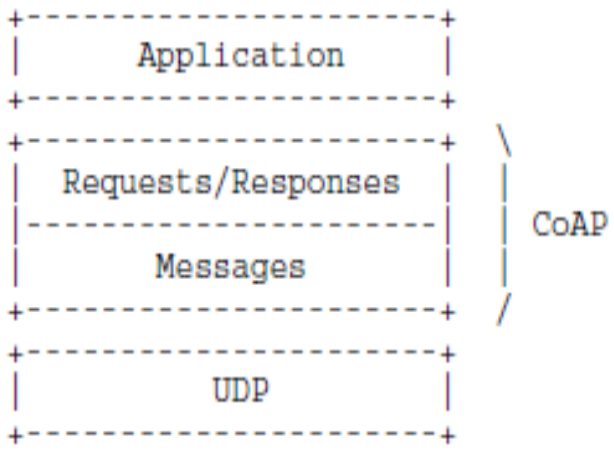


Constrained Application Protocol

- **CoAP 설계**
  - CoAP 모델
    - Client/Server 모델 (기존 HTTP 모델과 유사)



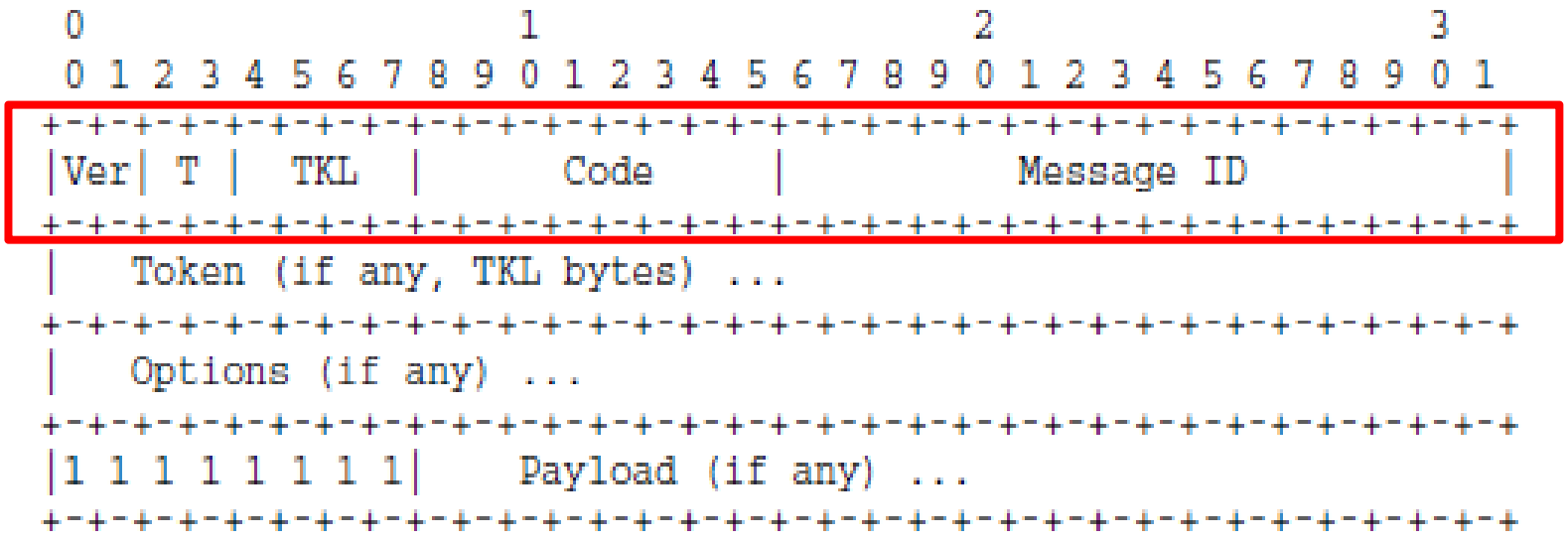
- **CoAP 계층 구조**
  - 논리적으로 두 개의 계층을 사용
    - Messages 계층
      - UDP 상에서 비동기식 정보 교환
      - CON, NON, ACK, RST
    - Request/Response
      - Method와 Code을 사용한 Request/Response 교환
  - CoAP은 Single protocol 이므로 두 계층의 내용이 CoAP 헤더에서 제공됨



Abstract Layering of CoAP

Messaging Model

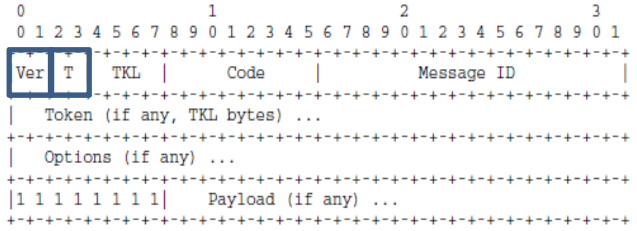
- **Message Format**
  - CoAP은 짧은 고정길이 이진 헤더(4 bytes)를 사용



Message format

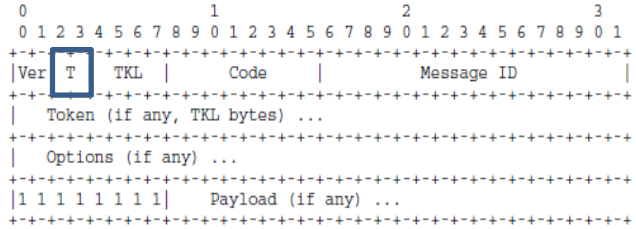
Messaging Model

- **Message Format**
  - Version (Ver)
    - 2bit의 양의 정수(unsigned integer)형태
    - CoAP의 버전 넘버를 표시
    - 알 수 없는 버전 번호의 Message는 자동으로 무시

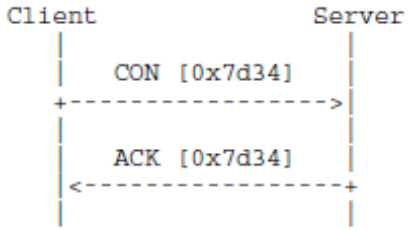


  - Type (T)
    - 2bit의 양의 정수(unsigned integer)형태
    - Message의 타입을 표시: CON(0), NON(1), ACK(2), Reset(3)
    - CoAP의 4가지 Message type
      - 확인형 (CON, Confirmable)
      - 비확인형 (NON, Non-confirmable)
      - 승인 (ACK, Acknowledgement)
      - 재설정 (RST, Reset)

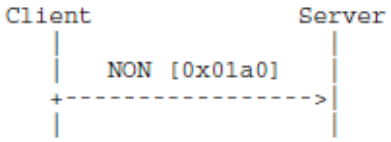
Messaging Model



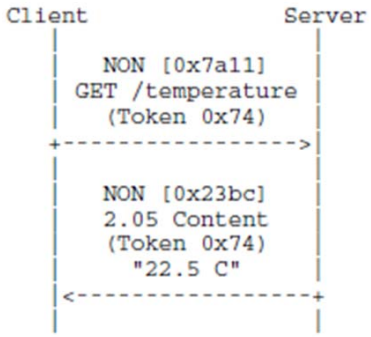
- **CoAP의 4가지 Message type**
  - 확인형 (CON, Confirmable)
    - CON Message는 요청에 대한 수신여부에 대한 응답으로 ACK을 받음으로써 정보교환의 신뢰성을 제공
    - 수신자가 받은 Message를 통해 동작을 수행할 수 없을 때 Reset를 보냄
  - 비 확인형 (NON, Non-confirmable)
    - NON Message는 센서 데이터 값을 얻어오는 것처럼 신뢰성 있는 전송이 요구되지 않을 때 사용
    - Client가 NON Message로 Request를 보내면 Server는 NON Message 타입으로 Response를 송신
  - 승인 (ACK, Acknowledgement)
    - CON Message에 대한 응답
  - 재설정 (RST, Reset)
    - 수신자가 받은 Message 처리할 수 없을 때의 응답



CON 예시



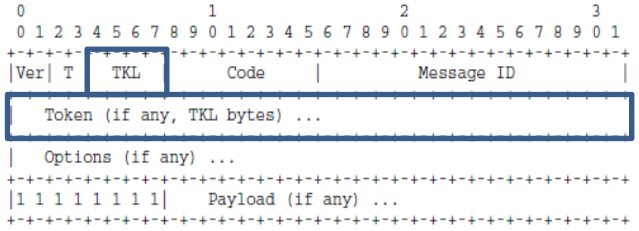
NON 예시

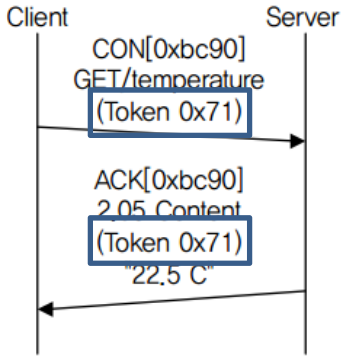


NON 에서의 Request/Response

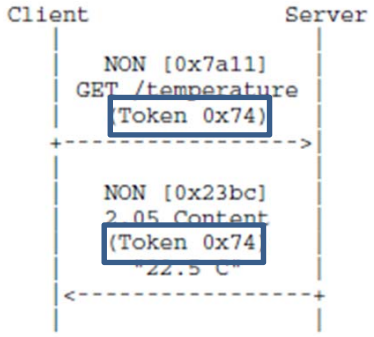
Messaging Model

- **Message Format**
  - Token Length (TKL)
    - 4bit 양의 정수(unsigned integer)형태
    - Token 필드의 길이를 표시
      - 0~8 bytes 까지의 값을 사용, 9~15 bytes (Reserved)
      - Token을 사용하지 않을 경우에는 TKL 값은 0
  - Token
    - 필요시 사용되는 필드
    - TKL 필드에 정의된 길이 만큼 Token 값을 기록함
    - CoAP 요청과 응답의 짝을 구분하기 위해 사용





CON Token

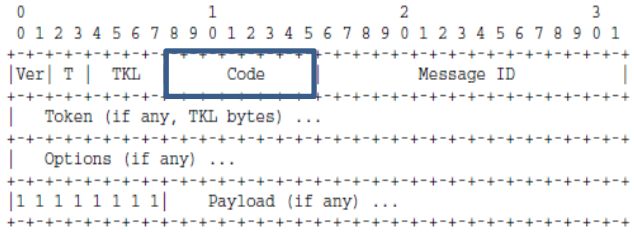


NON Token

HALLYM UNIVERSITY

Messaging Model

- **Message Format**
  - Code
    - 8bit 양의 정수(unsigned integer) 형태
      - 3비트는 클래스(Class)
      - 5비트는 자세한 내용(Detail)을 의미
      - CoAP 문서에서는 c.dd와 같은 형태로 표현
    - Class
      - 0: 요청, 2: 성공적인 응답, 4: Client 에러 응답, 5: Server 에러 응답
    - Detail
      - 요청의 경우, 0.01: GET, 0.02: POST, 0.03: PUT, 0.04: DELETE를 나타냄



Code	Name	Reference
0.01	GET	[RFC7252]
0.02	POST	[RFC7252]
0.03	PUT	[RFC7252]
0.04	DELETE	[RFC7252]

Method code

Method	의미
GET	Retrieve (인출)
POST	Create (생성)
PUT	Update (갱신)
DELETE	Delete (삭제)

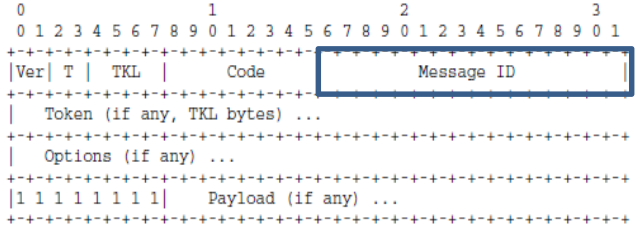
Code	Description	Reference
2.01	Created	[RFC7252]
2.02	Deleted	[RFC7252]
2.03	Valid	[RFC7252]
2.04	Changed	[RFC7252]
2.05	Content	[RFC7252]
4.00	Bad Request	[RFC7252]
4.01	Unauthorized	[RFC7252]
4.02	Bad Option	[RFC7252]
4.03	Forbidden	[RFC7252]
4.04	Not Found	[RFC7252]
4.05	Method Not Allowed	[RFC7252]
4.06	Not Acceptable	[RFC7252]
4.12	Precondition Failed	[RFC7252]
4.13	Request Entity Too Large	[RFC7252]
4.15	Unsupported Content-Format	[RFC7252]
5.00	Internal Server Error	[RFC7252]
5.01	Not Implemented	[RFC7252]
5.02	Bad Gateway	[RFC7252]
5.03	Service Unavailable	[RFC7252]
5.04	Gateway Timeout	[RFC7252]
5.05	Proxying Not Supported	[RFC7252]

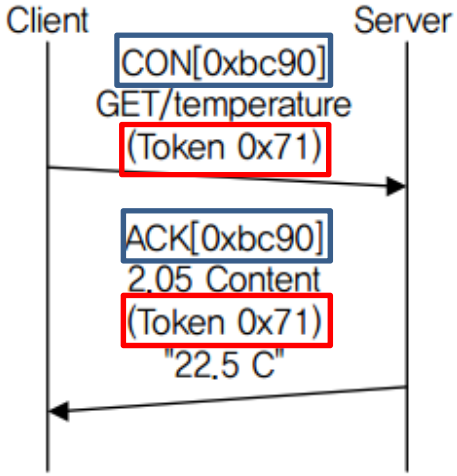
Response code

HALLYM UNIVERSITY

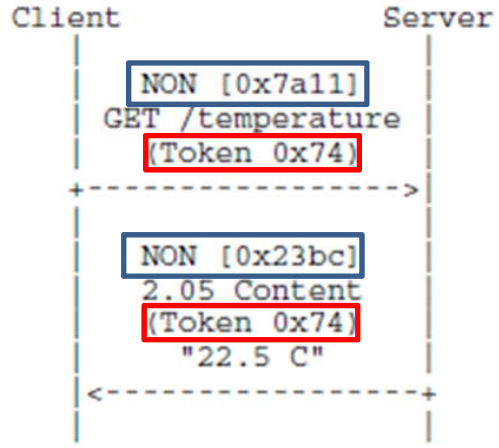
Messaging Model

- **Message Format**
  - Message ID
    - 2 Bytes의 양의 정수(unsigned integer) 형태
    - Message ID는 중복 확인 및 하나의 트랜잭션을 구분하는데 사용
      - 같은 Message가 중복으로 전송되었는지를 Message ID를 보고 판단
      - 하나의 트랜잭션을 구분 (CON: 요청/응답 같은 ID, NON: 매번 다른 ID)
      - Token vs. Message ID
        - Token은 요청에 대한 응답의 짝을 구분하기 위해 사용되고, Message ID는 하나의 트랜잭션을 구분하는데 사용





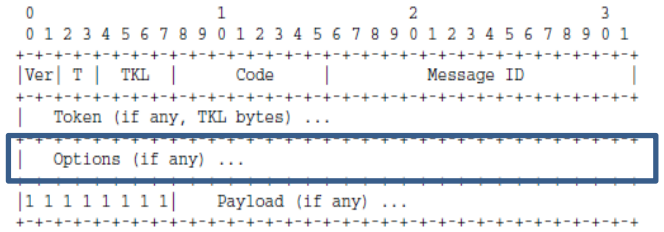
Token vs. Message ID (CON)

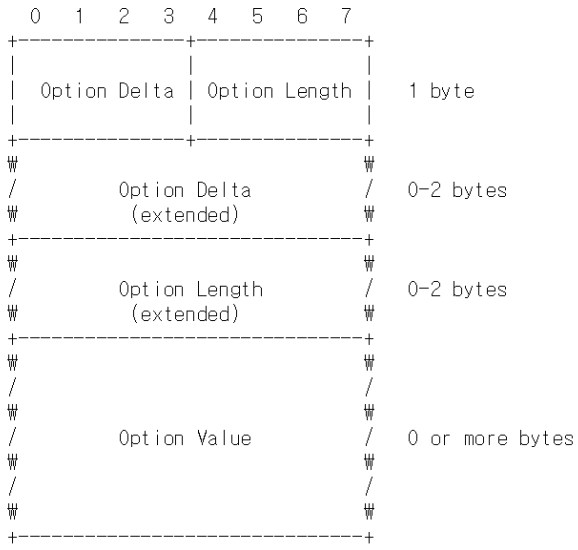


Token vs. Message ID (NON)

Messaging Model

- **Message Format**
  - Options
    - 필요시 사용되는 필드
    - CoAP에서 제공하는 Option을 정의
      - Option delta: Option 번호 결정
      - Option length: Option value 필드의 길이를 결정





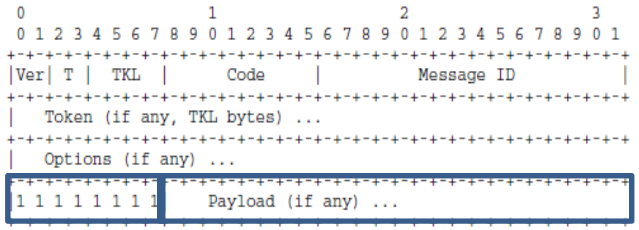
Option format

Number	Name	Reference
0	(Reserved)	[RFC7252]
1	If-Match	[RFC7252]
3	Uri-Host	[RFC7252]
4	ETag	[RFC7252]
5	If-None-Match	[RFC7252]
7	Uri-Port	[RFC7252]
8	Location-Path	[RFC7252]
11	Uri-Path	[RFC7252]
12	Content-Format	[RFC7252]
14	Max-Age	[RFC7252]
15	Uri-Query	[RFC7252]
17	Accept	[RFC7252]
20	Location-Query	[RFC7252]
35	Proxy-Uri	[RFC7252]
39	Proxy-Scheme	[RFC7252]
60	Size1	[RFC7252]
128	(Reserved)	[RFC7252]
132	(Reserved)	[RFC7252]
136	(Reserved)	[RFC7252]
140	(Reserved)	[RFC7252]

CoAP options

Messaging Model

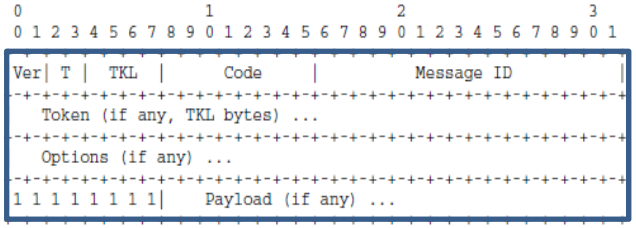
- **Message Format**
  - Payload Marker
    - Payload 구분자
      - Payload Marker 값: 11111111 (0xff)
      - Payload Marker로 더 이상 옵션이 없음을 알 수 있음

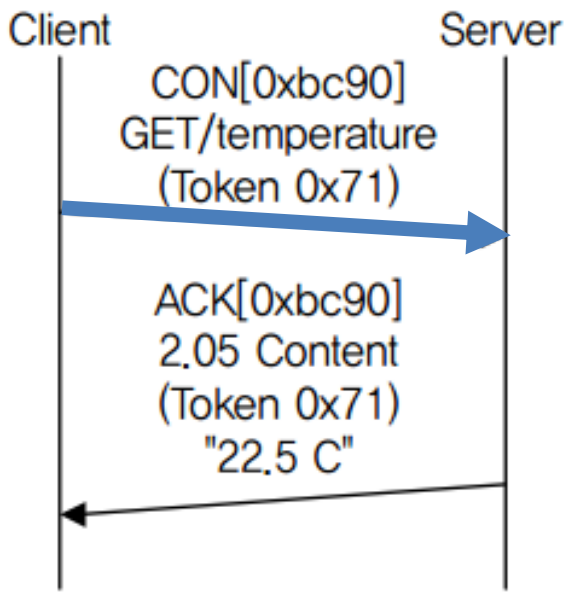


  - Payload
    - 필요시 사용되는 필드
    - Payload Marker 이후 위치하며 데이터 값을 정의

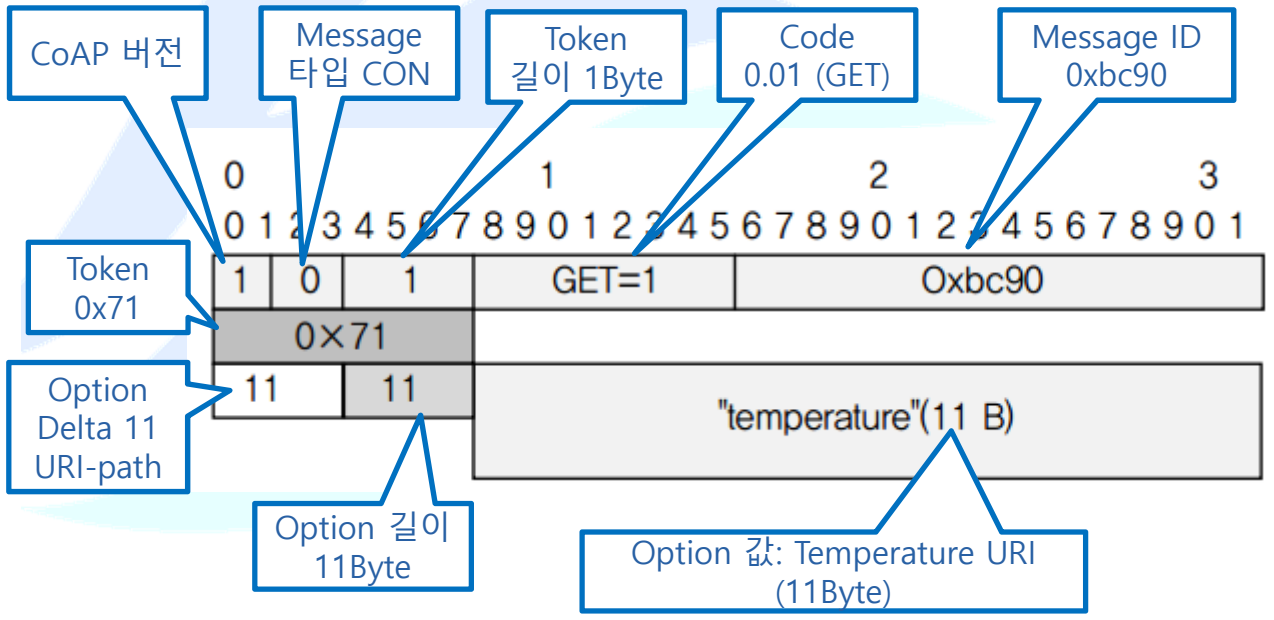
Messaging Model

- **CoAP Message 인코딩 예제**





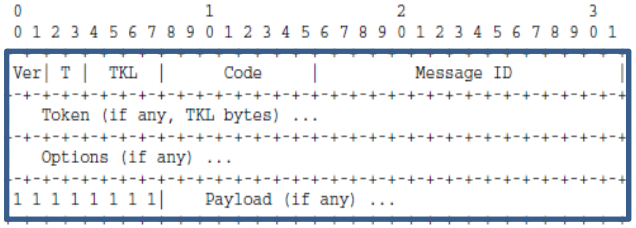
CON Message 요청/응답

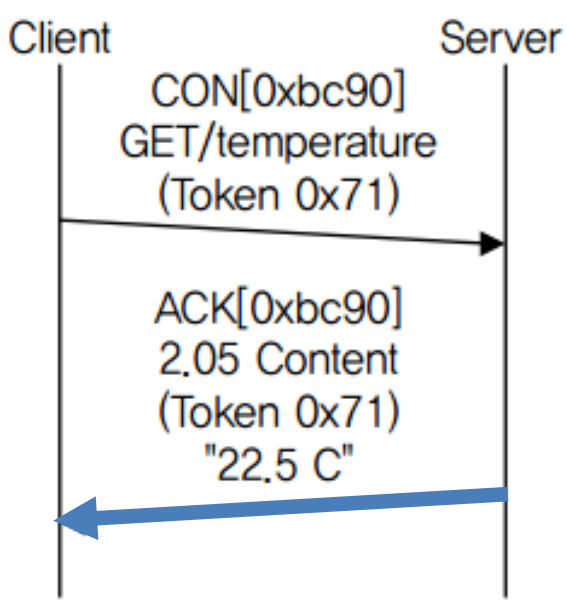


CoAP Message 인코딩 예제

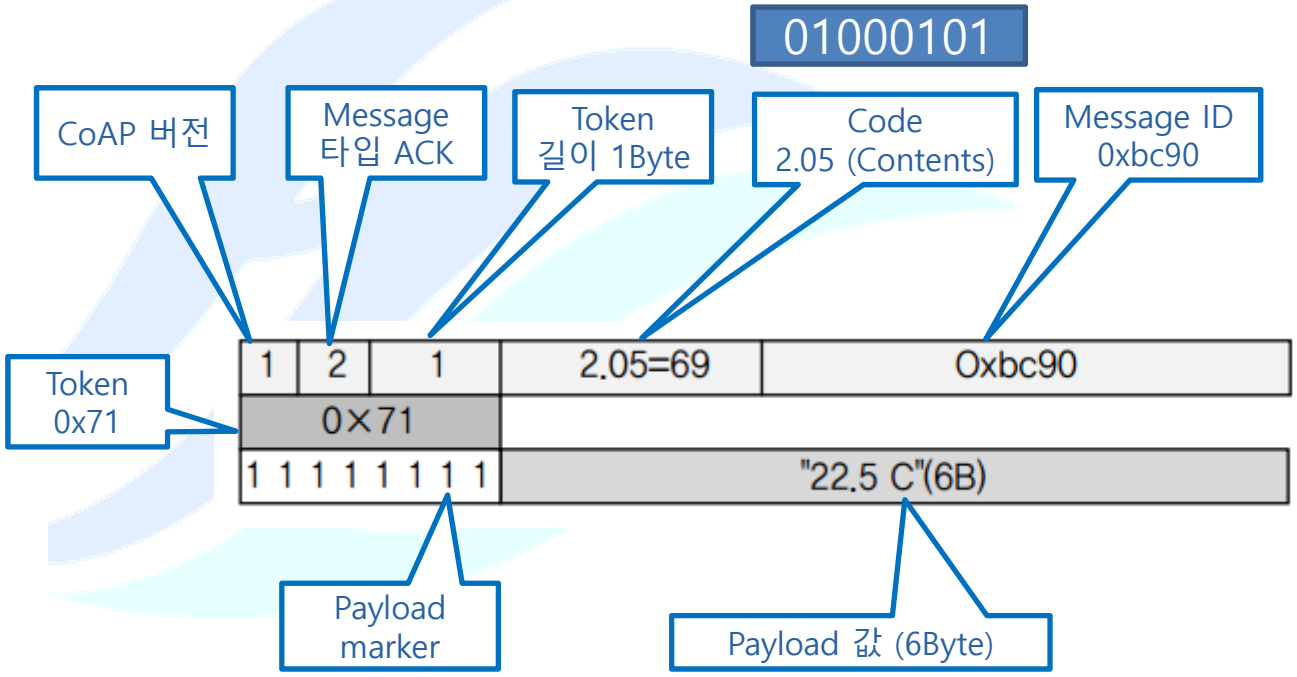
Messaging Model

- **CoAP Message 인코딩 예제**





CON Message 요청/응답



CoAP Message 인코딩 예제