2025/09/13, 2025 네트워크 세미나

TCP/IP 프로토콜

- 7장 IPv4, 8장 ARP -

송 민 희(minhee@pel.sejong.ac.kr)

세종대학교 프로토콜공학연구실

목 차

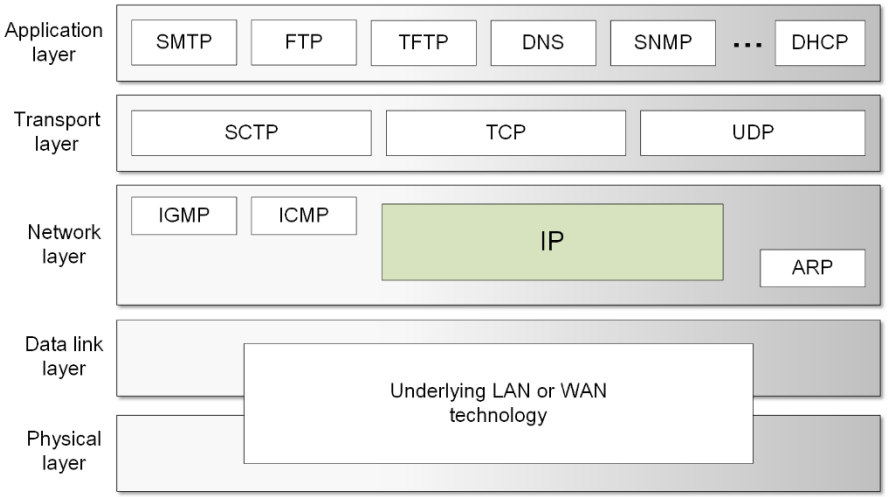
- IPv4
- ARP

목 차

- IPv4
- ARP

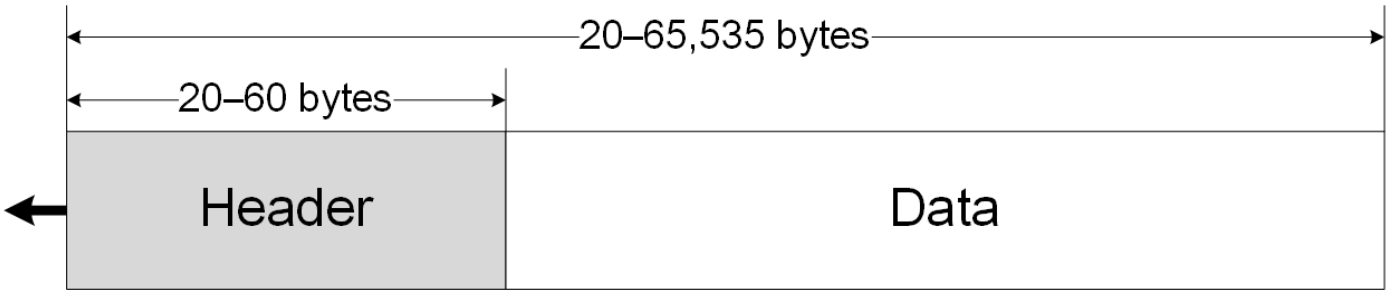
IPv4

- 인터넷 프로토콜(IP, Internet Protocol)
  - 정의
    - 네트워크 계층에서 TCP/IP 프로토콜이 사용하는 전송 메커니즘
  - 특징
    - 오류제어와 흐름제어 기능이 없는 데이터그램 프로토콜임
    - 비연결형(Connectionless) 데이터그램 방식임
    - 최선의 노력 서비스(Best Effort Service)임
      - 데이터의 전송을 위해 최대의 노력을 하지만 확실한 전송을 보장하지는 않음



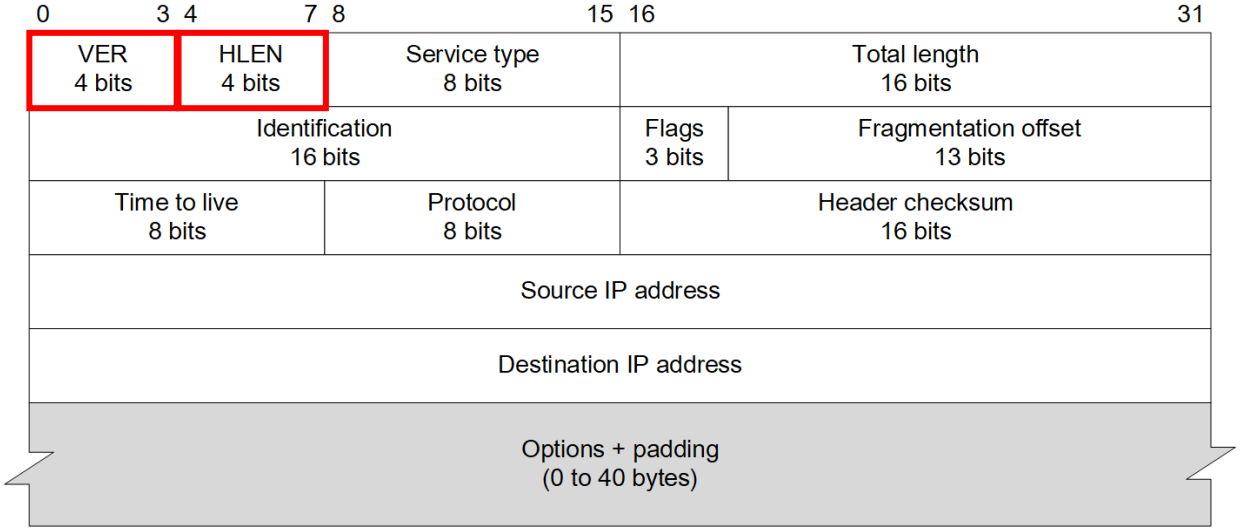
IPv4

- 데이터그램(Datagram) (1/13)
  - 정의
    - 패킷 교환에서 독립적으로 취급되는 네트워크 계층의 전송 단위
  - 특징
    - 가변 길이를 가짐
    - 헤더와 데이터로 구성됨
      - 헤더는 20~60바이트의 크기를 가짐
        - 20바이트의 고정 부분과 최대 40바이트의 가변 부분을 가짐
      - 데이터는 헤더의 길이에 따라 최대 65,515바이트의 크기를 가짐



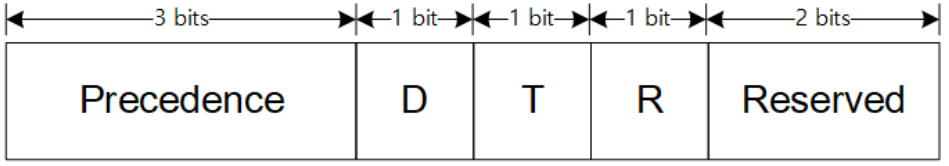
IPv4

- 데이터그램(Datagram) (2/13)
  - 구조 (1/8)
    - 버전(VER)
      - IP 프로토콜의 버전을 나타내는 4비트 필드
    - 헤더 길이(HLEN)
      - 데이터그램 헤더의 전체 길이를 4바이트 단위로 나타내는 4비트 필드
        - e.g., HLEN이 5의 값을 가지면 헤더의 길이는 20바이트임($5 \times 4 = 20$)

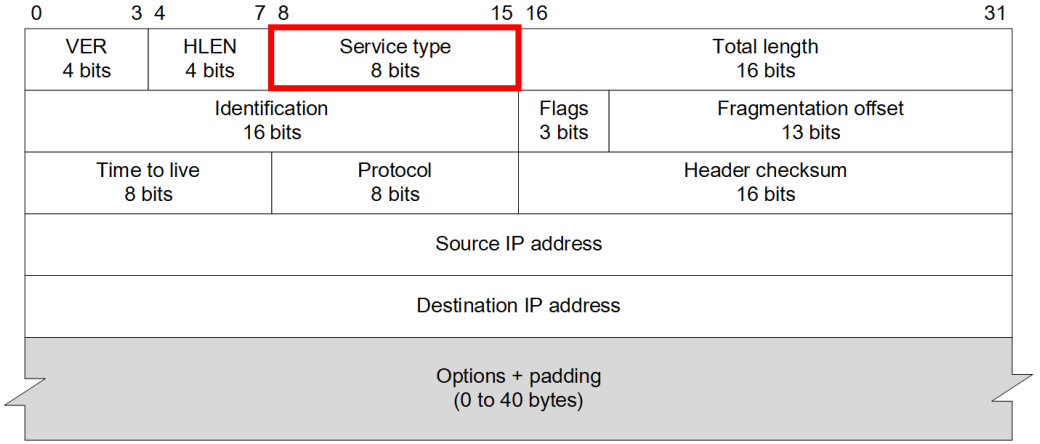


IPv4

- 데이터그램(Datagram) (3/13)
  - 구조 (2/8)
    - 서비스 유형(Service Type) (1/3)
      - 데이터그램이 처리되어야하는 방식을 정의하는 8비트 필드
      - ToS (Type of Service)
        - IETF RFC 791에서 정의한 IP 헤더의 초기 설계 구조에서의 서비스 유형 필드
        - 우선순위, 지연, 처리율, 신뢰성 등을 정의함
        - IETF RFC 2474부터 해당 필드를 DS Field(Differentiated Services Field)로 새롭게 정의함



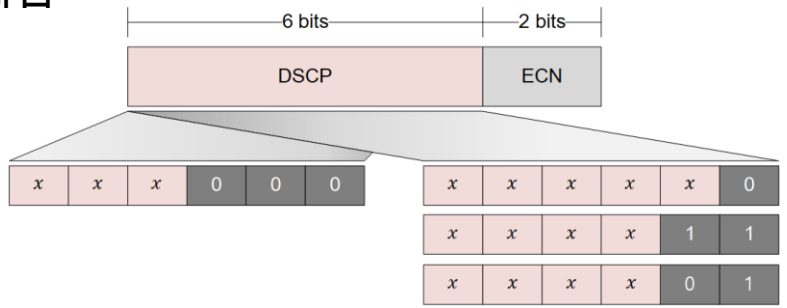
D : Delay
T : Throughput
R : Reliability



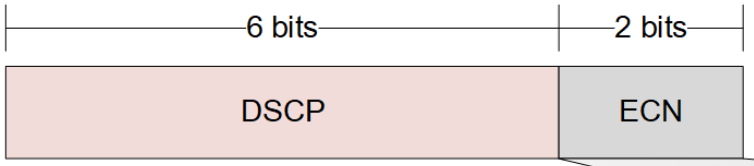
IPv4

- 데이터그램(Datagram) (4/13)
  - 구조 (3/8)
    - 서비스 유형 (2/3)
      - 차별화 서비스 필드(DS Field, Differentiated Services Field) (1/2)
        - IETF RFC 2474 부터 서비스 품질과 혼잡 알림에 대해 표현하기 위해 새롭게 정의된 서비스 유형 필드
        - DSCP (Differentiated Services Code Point)
          - 서비스 품질을 나타내기 위해 사용되는 6비트 필드
          - 오른쪽 세 비트가 0인 경우
            - 0~7 사이의 데이터그램 우선순위를 정의함
            - 라우터에 혼잡 발생 시 가장 낮은 우선순위의 데이터그램부터 삭제함
          - 오른쪽 세 비트가 0이 아닌 경우
            - 인터넷 당국(IETF), 지역당국, 실험 목적에 의해 범주가 구분됨
            - 우선순위에 따라 56가지의 서비스를 정의함
              - e.g., AF11(001010), AF32(011100), EF(101110)

Category	Codepoint	Assigning Authority
1	XXXXX0	Internet
2	XXXX11	Local
3	XXXX01	Temporary or experimental



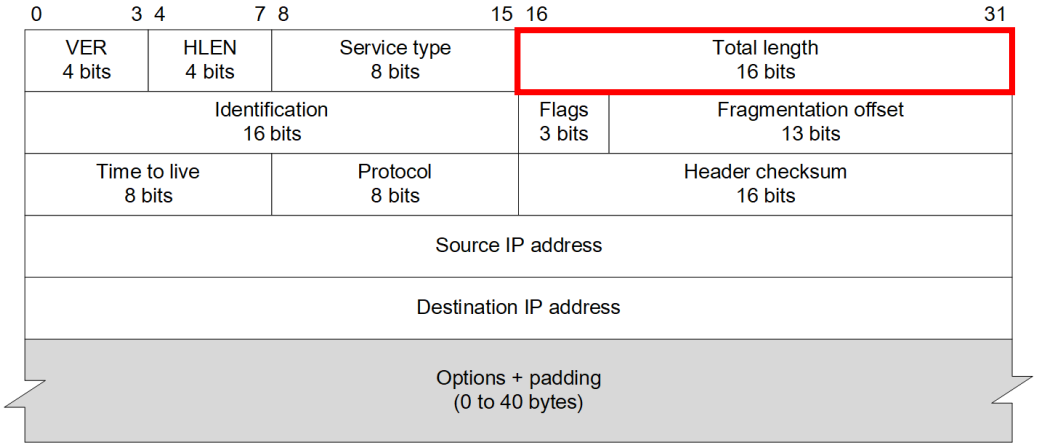
- 데이터그램(Datagram) (5/13)
  - 구조 (4/8)
    - 서비스 유형 (3/3)
      - 차별화 서비스 필드(DS Field, Differentiated Services Field) (2/2)
        - ECN (Explicit Congestion Notification)
          - 혼잡 알림을 위해 사용되는 2비트 필드
          - 00 : 패킷이 ECN 기능을 사용하지 않음
          - 01 / 10 : 패킷이 ECN 기능을 사용할 수 있음
          - 11 : 라우터가 혼잡을 감지하여 표시함

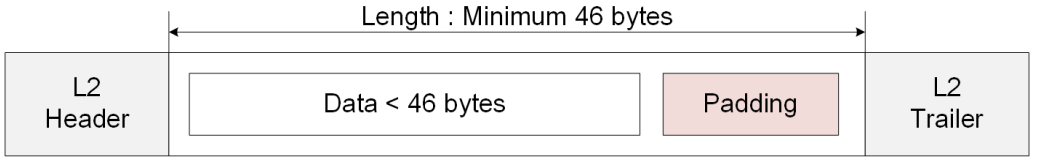


Binary	Keyword	Description
00	Not-ECT	Not ECN-Capable Transport
01	ECT(1)	ECN-Capable Transport(1)
10	ECT(0)	ECN-Capable Transport(0)
11	CE	Congestion Experienced

IPv4

- 데이터그램(Datagram) (6/13)
  - 구조 (5/8)
    - 전체 길이(Total Length)
      - 헤더와 데이터를 포함하는 IP 데이터그램의 전체 길이를 바이트 단위로 나타내는 16비트 필드
      - 데이터그램에 있는 데이터와 패딩을 구분하기 위하여 사용함
        - 데이터 길이 = 전체 길이 – 헤더 길이(HLEN × 4)

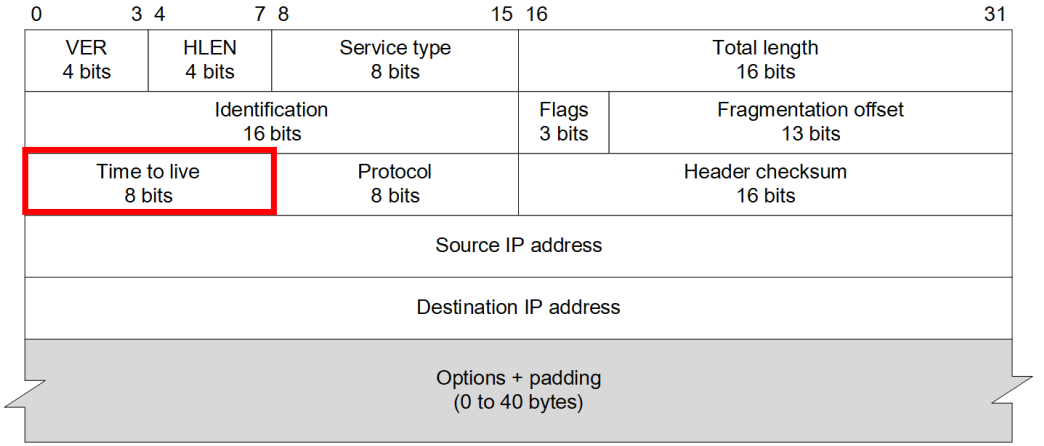




IPv4

- 데이터그램(Datagram) (7/13)
  - 구조 (6/8)
    - 수명(TTL, Time To Live)
      - 데이터그램이 방문할 수 있는 최대 *홉 수를 나타내는 8비트 필드
      - 홉을 경유할 때 마다 필드의 값이 1씩 감소되며, 필드의 값이 0이 되면 데이터그램이 폐기됨
      - RFC 6450에 의하면, TTL의 값은 최소 64로 설정하기를 권고하고 있음
        - 대부분의 네트워크에서 흔히 예상하는 홉 수(약 10~30)를 충분히 커버할 수 있는 수준임

*홉(Hop)
컴퓨터 네트워크에서 출발지와 목적지 사이에 위치한 경로의 한 부분



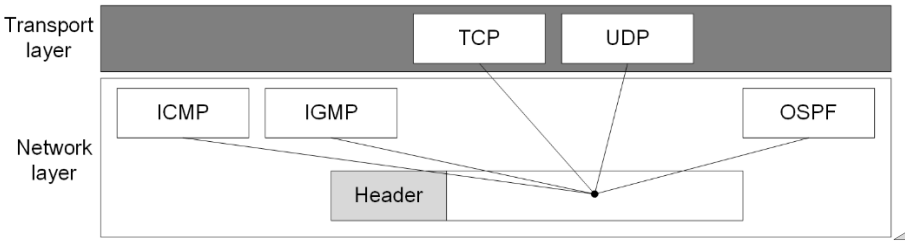
IPv4

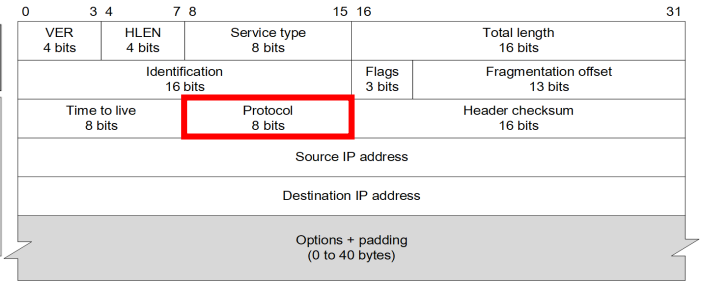
- 데이터그램(Datagram) (8/13)

*캡슐화(Encapsulation)
상위 계층의 데이터에 하위 계층의 헤더를 붙여 전송 가능한 형식으로 변환하는 과정

  - 구조 (7/8)
    - 프로토콜(Protocol)
      - IP 계층의 서비스를 사용하는 상위 계층 프로토콜의 종류를 식별하기 위한 8비트 필드
      - IP 프로토콜이 상위 계층 프로토콜로부터 오는 데이터를 다중화 및 역다중화 할 때 해당 필드 값을 활용함
        - 전송 계층에서의 다중화
          - 전송 계층의 세그먼트들을 IP 프로토콜을 통해 *캡슐화하여 네트워크 계층으로 전달하는 작업
        - 전송 계층에서의 역다중화
          - 네트워크 계층으로 도착한 데이터그램을 전송 계층의 올바른 소켓으로 전달하는 작업

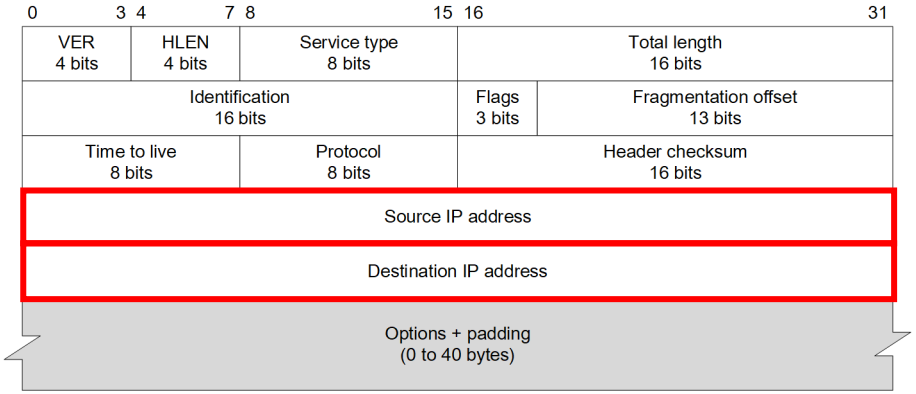
Value	Protocol
1	ICMP
2	IGMP
6	TCP
17	UDP
89	OSPF





IPv4

- 데이터그램(Datagram) (9/13)
  - 구조 (8/8)
    - 발신지 주소(Source Address)
      - 발신지의 IP 주소를 정의하는 32비트 필드
      - IP 데이터그램이 발신지에서 목적지까지 전달되는 동안 발신지 주소의 값이 변해서는 안됨
    - 목적지 주소(Destination Address)
      - 목적지의 IP 주소를 정의하는 32비트 필드
      - IP 데이터그램이 발신지에서 목적지까지 전달되는 동안 목적지 주소의 값이 변해서는 안됨



IPv4

- 데이터그램(Datagram) (10/13)
  - 예제 7.1

처음 8비트가 다음과 같은 IP 패킷이 도착하였다.

$$01000010$$

수신자가 해당 패킷을 폐기하는 이유를 설명하라.

- 왼쪽 4비트(0100)는 버전을 나타냄
  - $2^2 \times 1 = 4$ 로, IPv4를 의미함
    ∴ 올바른 값
- 오른쪽 4비트(0010)는 헤더의 길이를 나타냄
  - $2^1 \times 1 = 2, 2 \times 4 = 8$ 이지만, 헤더의 최소 바이트 수는 20임
    ∴ 잘못된 값

∴ 잘못된 값이 존재하므로 해당 패킷은 훼손되어 폐기함

IPv4

- 데이터그램(Datagram) (11/13)
  - 예제 7.2

IP 패킷에서 HLEN의 값이 이진수로 1000이다.
해당 패킷에는 옵션이 몇 바이트 있는가?

- HLEN의 길이는 $2^3 \times 1 = 8$임
- 헤더 내 바이트의 총 수는 $8 \times 4 = 32$바이트임
- 처음 20바이트는 헤더이므로 옵션은 $32 - 20 = 12$바이트임

$\therefore$ 12바이트

IPv4

- 데이터그램(Datagram) (12/13)
  - 예제 7.3

IP 패킷에서 HLEN의 값이 16진수로 5이고 전체 길이 필드의 값이 0x0028이다. 해당 패킷은 몇 바이트의 데이터를 전송하고 있는가?

- HLEN의 값이 5 이므로 $5 \times 4 = 20$바이트임
  - 처음 20바이트는 헤더이므로 옵션은 $20 - 20 = 0$바이트로 옵션은 존재하지 않음
- 전체 길이 필드의 값이 0x0028 이므로 패킷의 전체 길이는 $16^1 \times 2 + 16^0 \times 8 = 40$바이트임
  - $40 - 20 = 20$바이트의 데이터가 전송됨

∴ 20바이트

- 데이터그램(Datagram) (13/13)
  - 예제 7.4

처음 16진수 값이 다음과 같은 패킷이 도착하였다.

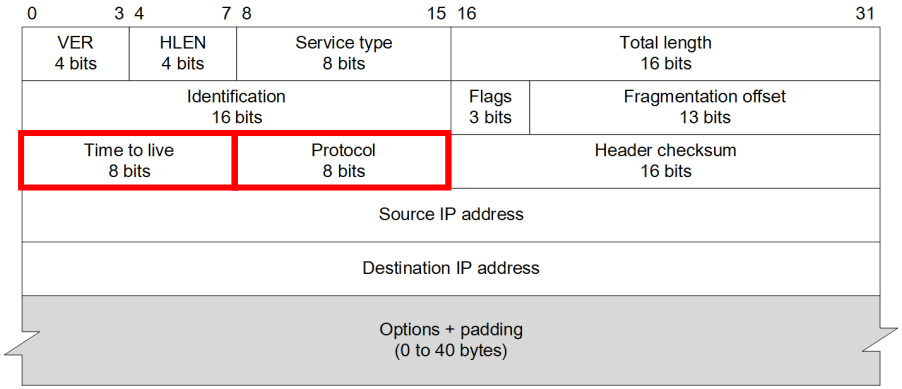
45 00 00 28 00 01 00 00 01 02 ...

해당 패킷은 폐기되기 전에 몇 홉을 지나갈 수 있는가?

해당 패킷은 어떤 상위계층 프로토콜 데이터를 전송하고 있는가?

- 수명 필드는 9번째 바이트에 위치하므로 01의 값을 가짐
  ∴ 패킷이 한 홉만 갈 수 있음
- 프로토콜 필드는 10번째 바이트에 위치하므로 02의 값을 가짐
  ∴ 상위 계층 프로토콜이 IGMP임

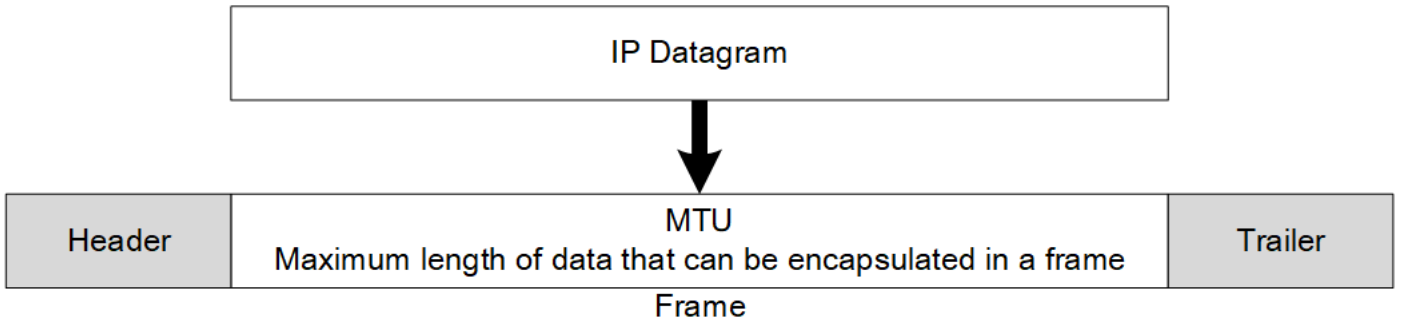
Value	Protocol	Value	Protocol
1	ICMP	17	UDP
2	IGMP	89	OSPF
6	TCP		



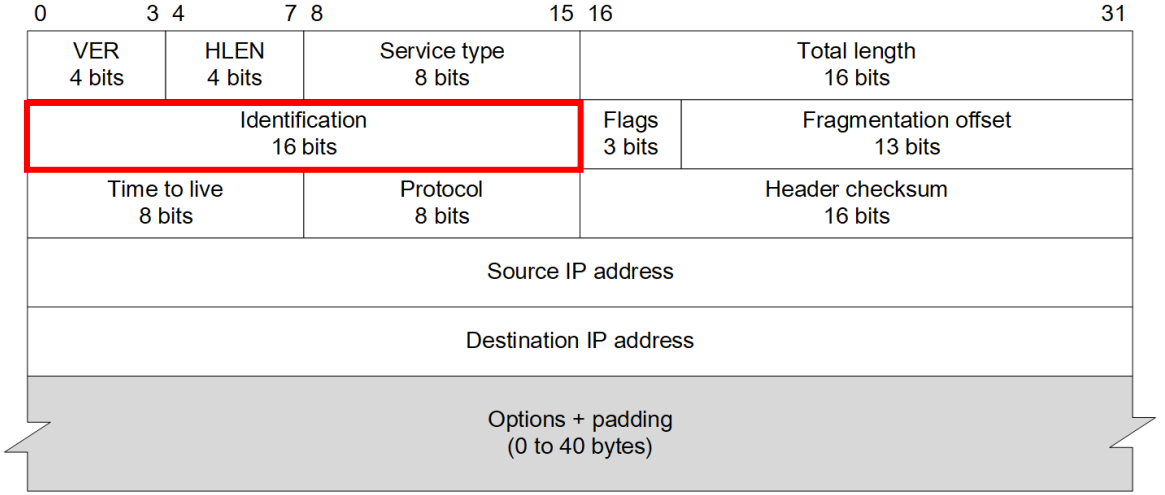
IPv4

- IP 단편화(IP Fragmentation) (1/12)
  - 정의
    - 하나의 IP 데이터그램이 네트워크의 최대 전달 단위(MTU, Maximum Transfer Unit)보다 클 경우, 이를 여러 개의 작은 단편으로 나누어 전송하는 과정
  - 특징
    - 단편화
      - 데이터그램에서 헤더를 제외한 데이터만 단편화 됨
      - 데이터그램은 최종 목적지 도착 전까지 여러지점에서 단편화 됨
    - 전송 과정
      - 단편화 된 데이터그램은 서로 다른 경로로 전달될 수 있음
      - 동일 데이터그램의 단편은 모두 목적지 호스트에 도착해야 재조립됨
    - 재조립
      - 단편화의 재조립은 최종 목적지에서만 수행됨
      - 단편의 IP 헤더는 플래그, 단편화 옵션, 전체 길이를 제외한 나머지 필드를 공통으로 사용함

- IP 단편화 (2/12)
  - 최대 전달 단위(MTU, Maximum Transfer Unit)
    - 정의
      - 하나의 IP 데이터그램이 단편화 없이 전송될 수 있는 최대 크기
    - 특징
      - 데이터그램이 프레임 속에 캡슐화될 때 데이터그램의 크기는 MTU보다 작아야 함
      - MTU는 네트워크 내에서 사용되는 데이터 링크 프로토콜에 따라 결정됨
        - e.g., Ethernet LAN은 1500바이트, FDDI LAN의 경우에는 4352바이트, PPP의 경우에는 296바이트의 MTU를 가짐



- IP 단편화 (3/12)
  - 관련 필드 (1/3)
    - 식별(Identification)
      - 발신지 호스트로부터 나온 데이터그램을 유일하게 식별하는 16비트 필드
      - 데이터그램을 보낼 때 마다 카운터의 현재 값을 식별에 복사하고 카운터 값을 1씩 증가시킴
        - 식별의 값이 16비트를 넘어가면 0부터 다시 시작함
      - 하나의 데이터그램에서 단편은 같은 식별 값을 가짐

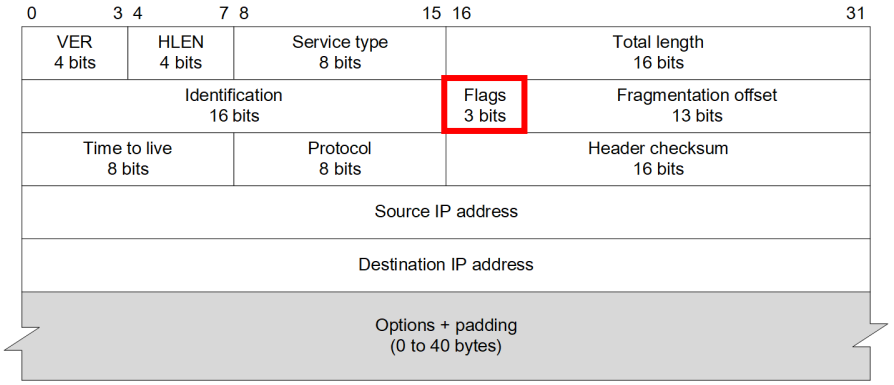


- IP 단편화 (4/12)
  - 관련 필드 (2/3)
    - 플래그(Flag)
      - IP 데이터그램의 단편화 관련 상태를 정의하는 3비트 필드
      - 첫 번째 비트(R)는 사용하지 않음
      - 두 번째 비트(D)는 Do not fragment 비트로, 값이 1이면 단편화해서는 안됨
      - 세 번째 비트(M)는 More fragment 비트로, 값이 1이면 데이터그램이 마지막 단편이 아님을 알려줌

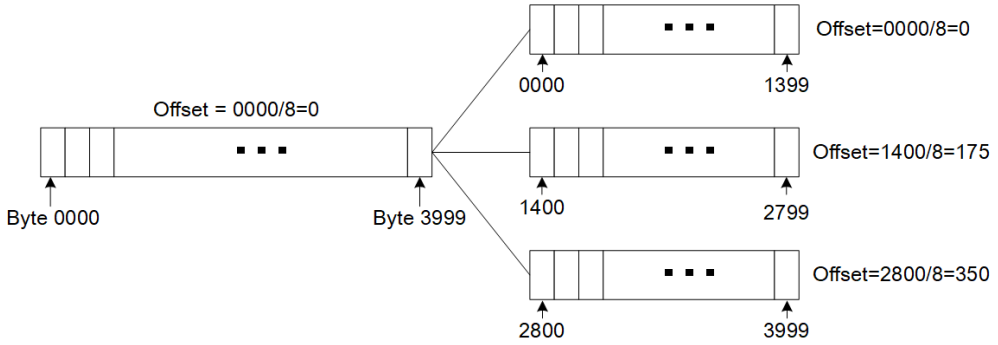
R : Reserved Flag
D : Do not fragment Flag
M : More fragment Flag



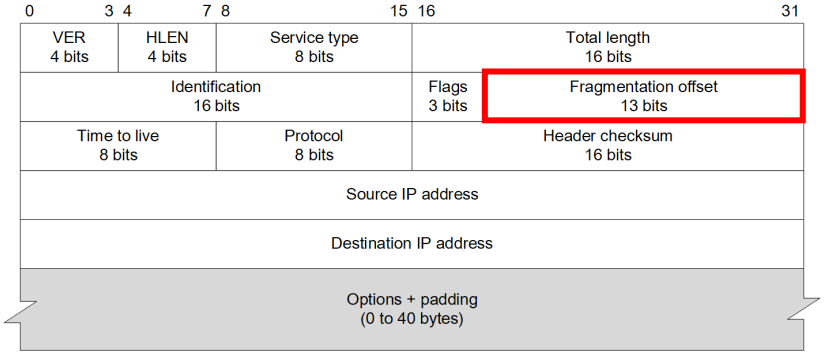
Value	D	M
0	단편화 수행 가능	마지막 단편 혹은 유일한 단편
1	단편화 불가능	마지막 단편이 아님



- IP 단편화 (5/12)
  - 관련 필드 (3/3)
    - 단편화 옵셋(Fragmentation Offset)
      - 데이터그램 내에서 단편의 상대적 위치를 8바이트 단위로 나타내는 13비트 필드
      - 필드 크기가 13비트이므로 8,191보다 큰 값을 표현할 수 없음
      - 각 단편의 첫번째 데이터의 인덱스는 8로 나눠지는 숫자여야 함
        - 원본 데이터그램 내에서의 데이터의 옵셋을 8바이트 단위로 나타냄

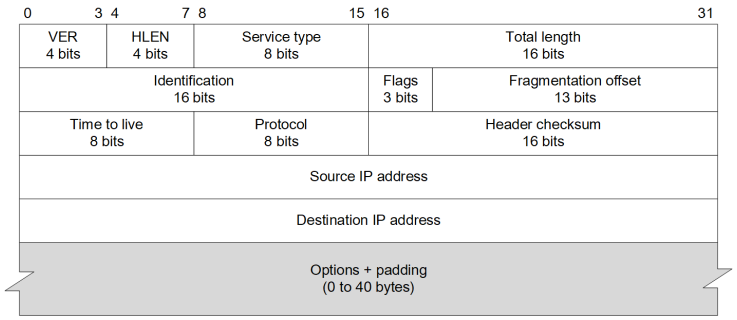


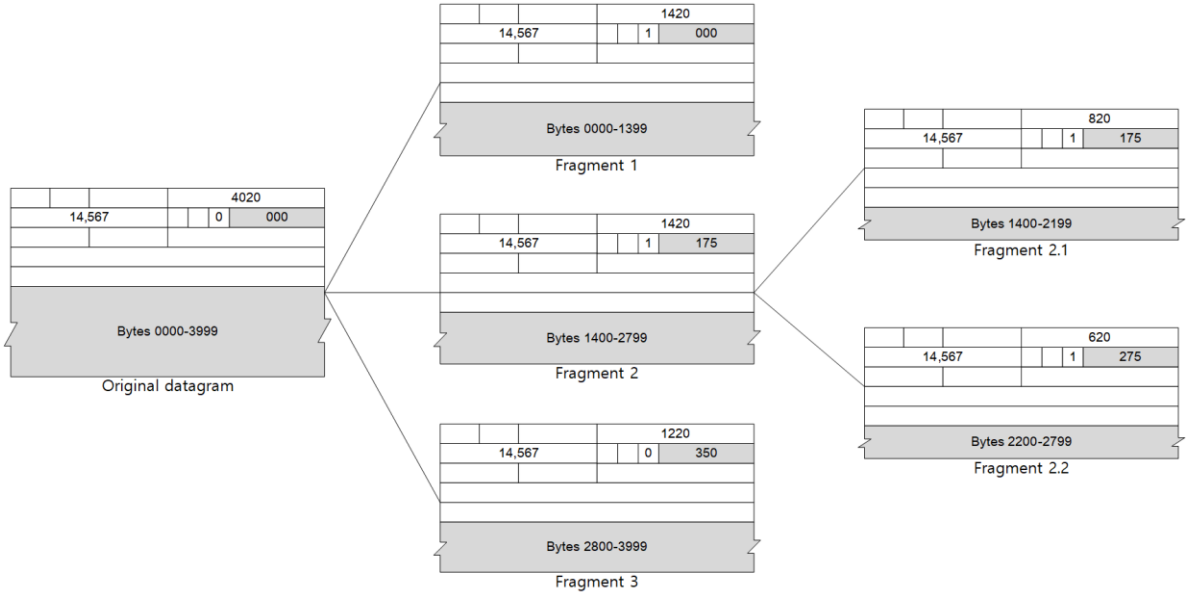
e.g., 데이터 크기가 4,000바이트인 데이터그램이 세 개의 단편으로 나누어진 경우



IPv4

- IP 단편화 (6/12)
  - 4020바이트 데이터그램의 단편화 예시
    - 식별 값이 14,567로 모두 동일함
    - 원본 데이터그램과 마지막 단편은 M비트가 0으로 설정 됨
    - 모든 단편의 크기는 헤더의 기본길이를 제외하고 8로 나누어 떨어짐
      - 8로 나눈 값이 다음 단편의 단편화 옵셋 값이 됨





IPv4

- IP 단편화 (7/12)
  - 재조립 과정
    1. 첫 번째 단편의 옵셋 값이 0인지 확인
    2. 첫 번째 단편의 길이를 8로 나누고, 두 번째 단편의 옵셋 값과 동일한지 확인
    3. 첫 번째 단편과 두 번째 단편의 식별 값이 동일한지 확인
    4. 첫 번째와 두 번째 단편의 길이의 합을 8로 나누고, 세 번째 단편의 옵셋 값과 동일한지 확인
    5. 세 번째 단편과 이전 단편들의 식별 값이 동일한지 확인
    6. 위와 같은 과정을 more fragment 값이 0이 될 때까지 반복

- IP 단편화 (8/12)
  - 예제 7.5

M비트 값이 0인 패킷이 도착하였다. 해당 패킷은 첫 번째 단편인가, 마지막 단편인가, 또는 중간 단편인가? 패킷이 단편화 되었는지 알 수 있는가?

  - M비트 0이면 뒤에 이어지는 단편이 없음
    - 유일한 단편이거나 마지막 단편임
  - 원래의 패킷이 단편화 되었는지 알 수 없음
    - 이를 알기 위해서는 단편화 옵셋의 정보가 필요함

IPv4

- IP 단편화 (9/12)
  - 예제 7.6

M비트 값이 1인 패킷이 도착하였다. 해당 패킷은 첫 번째 단편인가, 마지막 단편인가 또는 중간 단편인가? 패킷이 단편화 되었는지 알 수 있는가?

- M비트가 1이면 적어도 1개의 단편이 추가로 존재함
  - 위의 패킷은 첫 번째 단편이거나 중간 단편임
- 원래의 패킷이 단편화 되었는지 알 수 없음
  - 이를 알기 위해서는 단편화 옵셋의 정보가 필요함

- IP 단편화 (10/12)
  - 예제 7.7

M비트가 1이고 단편화 옵셋이 0인 패킷이 도착하였다. 해당 패킷은 첫 번째 단편인가, 마지막 단편인가, 또는 중간 단편인가?

    - M비트가 1이면 적어도 1개의 단편이 추가로 존재함
      - 위의 패킷은 첫 번째 단편이거나 중간 단편임
    - 옵셋이 0이므로 첫 번째 단편임

- IP 단편화 (11/12)
  - 예제 7.8

옵셋 값이 100인 패킷이 도착하였다. 첫 번째 바이트의 번호는 무엇인가? 마지막 바이트의 번호를 알 수 있는가?

  - 첫 번째 바이트의 번호 : $100 \times 8 = 800$
  - 데이터의 길이가 주어지지 않았기 때문에 마지막 바이트의 번호를 알 수 없음

IPv4

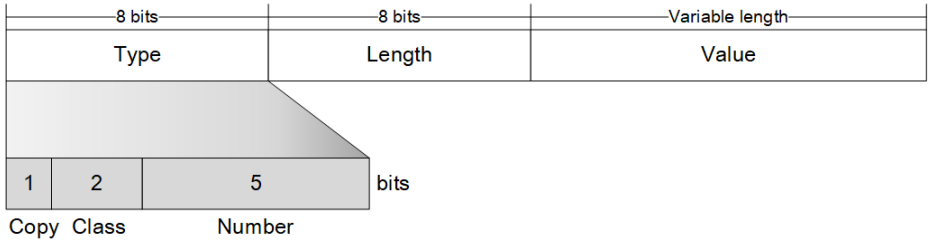
- IP 단편화 (12/12)
  - 예제 7.9

옵셋 값이 100이고 HLEN이 5이며 전체 길이가 100인 패킷이 도착하였다. 첫 번째와 마지막 바이트의 번호를 구하라.

- 첫 번째 바이트의 번호 : $100 \times 8 = 800$
- 전체 길이는 100바이트이고, 헤더 길이는 20바이트 이므로 데이터그램 내에는 $100 - 20 = 80$바이트의 데이터가 있음
- 첫 번째 바이트 번호가 800이므로 마지막 바이트의 번호는 $800 + 80 - 1 = 879$가 됨

IPv4

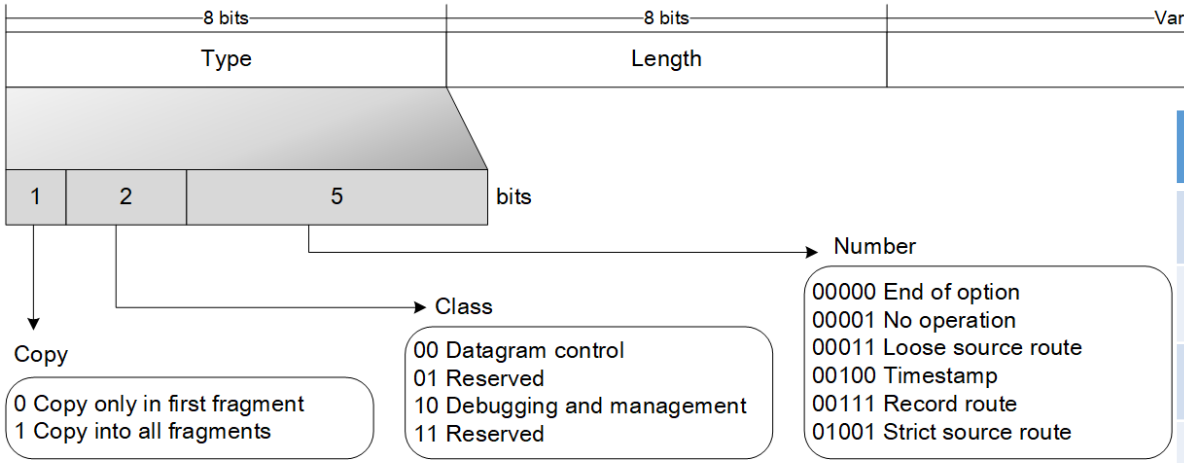
- 옵션 필드(Option Field) (1/3)
  - 정의
    - 특수한 기능이나 추가 정보를 담기 위해 사용되는 선택적 필드
  - 구조(Format) (1/3)
    - 유형(Type) (1/2)
      - 복사(Copy)
        - 단편화에 옵션을 포함시킬 것인지를 나타내는 1비트 필드
      - 클래스(Class)
        - 옵션의 일반적인 목적을 정의하는 2비트 필드



복사 필드	의미
0	옵션이 첫 번째 단편에만 복사됨
1	옵션이 모든 단편에 복사됨

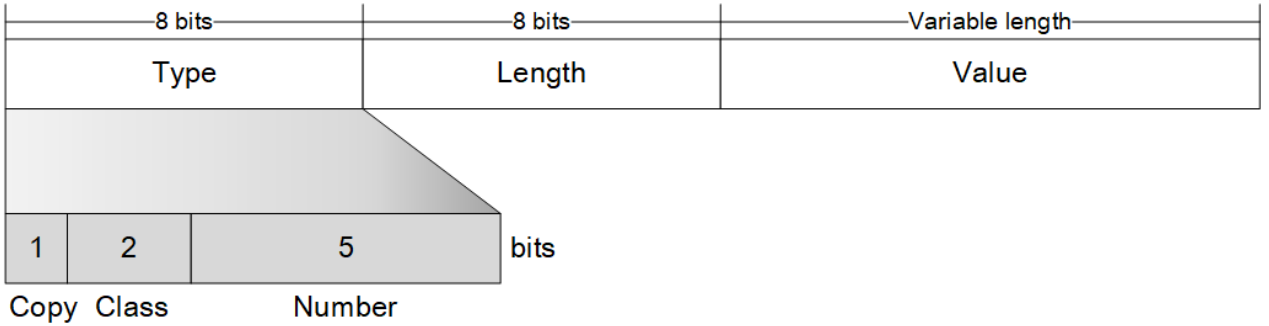
클래스 필드	의미
00	옵션이 데이터그램의 제어에 사용됨
10	옵션이 디버그나 관리 목적에 사용됨
01 / 11	정의되지 않음

- 옵션 필드(Option Field) (2/3)
  - 구조(Format) (2/3)
    - 유형(Type) (2/2)
      - 번호(Number)
        - 옵션의 유형을 정의하는 5비트 필드
        - $2^5 = 32$로 32개의 유형을 정의할 수 있으나, 현재는 6개의 유형만 정의됨



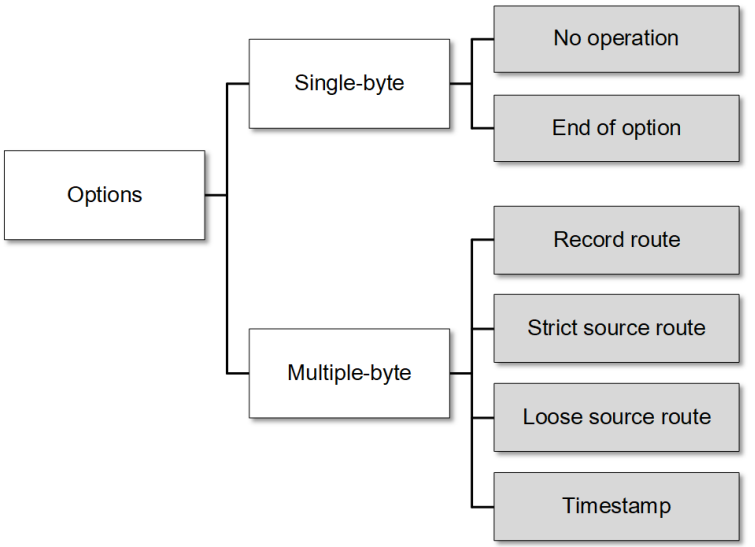
숫자 필드	옵션 유형
00000	End of option
00001	No operation
00011	Loose source route
00100	Timestamp
00111	Record route
01001	Strict source route

- 옵션 필드(Option Field) (3/3)
  - 구조(Format) (3/3)
    - 길이(Length)
      - 옵션의 전체 길의를 정의하는 8비트 필드
      - 모든 옵션 유형에 존재하는 것은 아님
    - 값(Value)
      - 특정 옵션이 필요로 하는 데이터를 포함하고 있는 가변적인 크기의 필드
      - 모든 옵션 유형에 존재하는 것은 아님

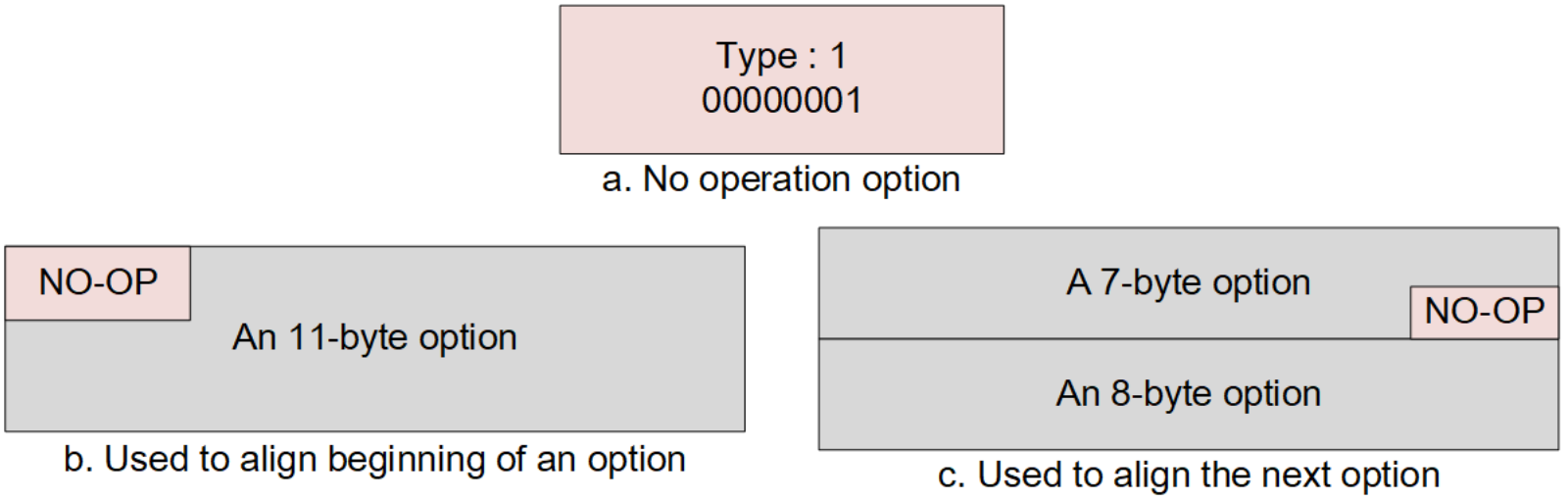


IPv4

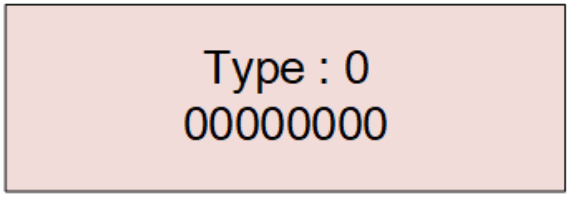
- 옵션 유형(Option Type) (1/14)
  - Single-byte 옵션
    - 1바이트로 구성되는 옵션
    - 추가적인 데이터 없이 형식(Type) 필드만 사용함
    - 주로 패딩과 같은 간단한 기능을 수행함
  - Multiple-byte 옵션
    - 2바이트 이상으로 구성되는 옵션
    - Type, Length, Data 필드로 구성됨
    - 라우트 기록, 타임스탬프 등 복잡한 기능을 수행함



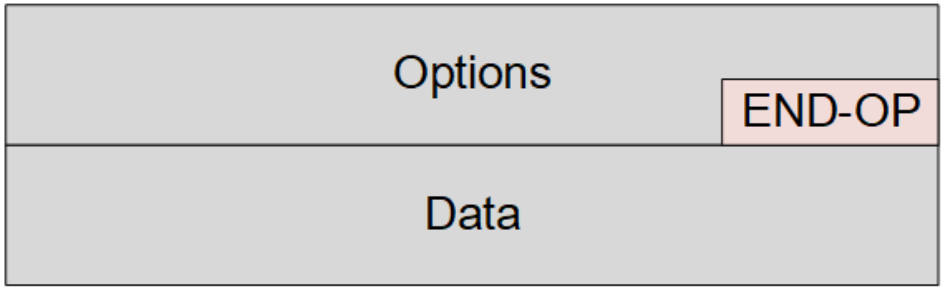
- 옵션 유형(Option Type) (2/14)
  - Single-byte 옵션 (1/2)
    - 무연산(No Operation) 옵션
      - 1바이트 옵션으로, 옵션들 사이의 여백을 채우기 위해 사용함



- 옵션 유형(Option Type) (3/14)
  - Single-byte 옵션 (2/2)
    - 옵션 종료(End of Option) 옵션
      - 옵션의 필드 끝의 패딩 목적으로 사용함
      - 마지막 옵션으로만 사용할 수 있으며, 하나의 옵션 종료만 사용할 수 있음
        - 옵션 필드의 경계를 맞추기 위해 한 바이트 이상이 필요하다면 무연산 옵션이 사용된 후 마지막에 옵션 종료 옵션이 사용되어야 함

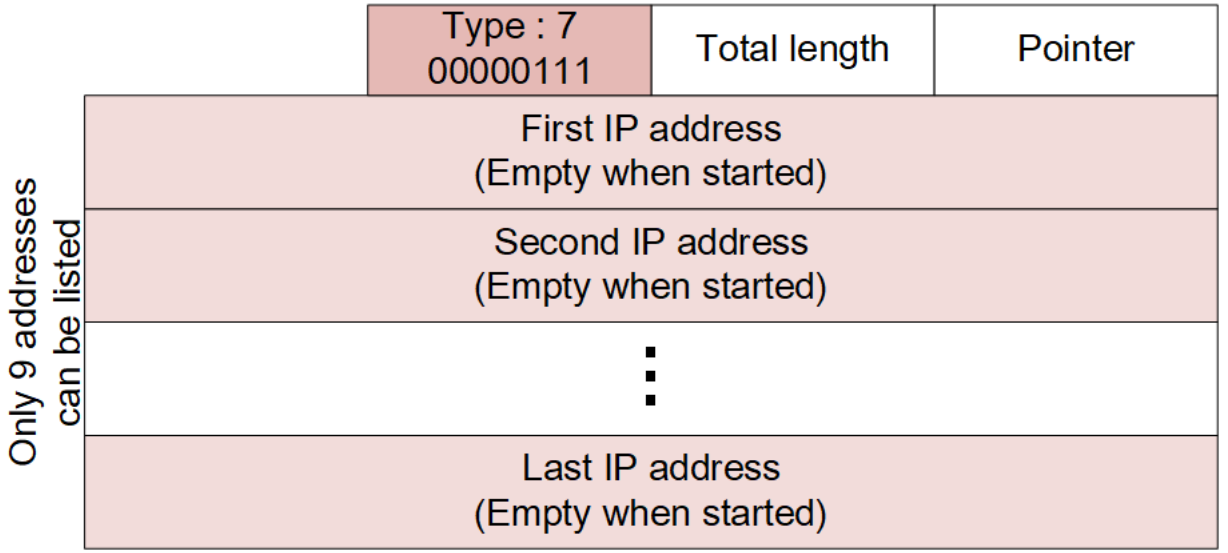


a. End of option

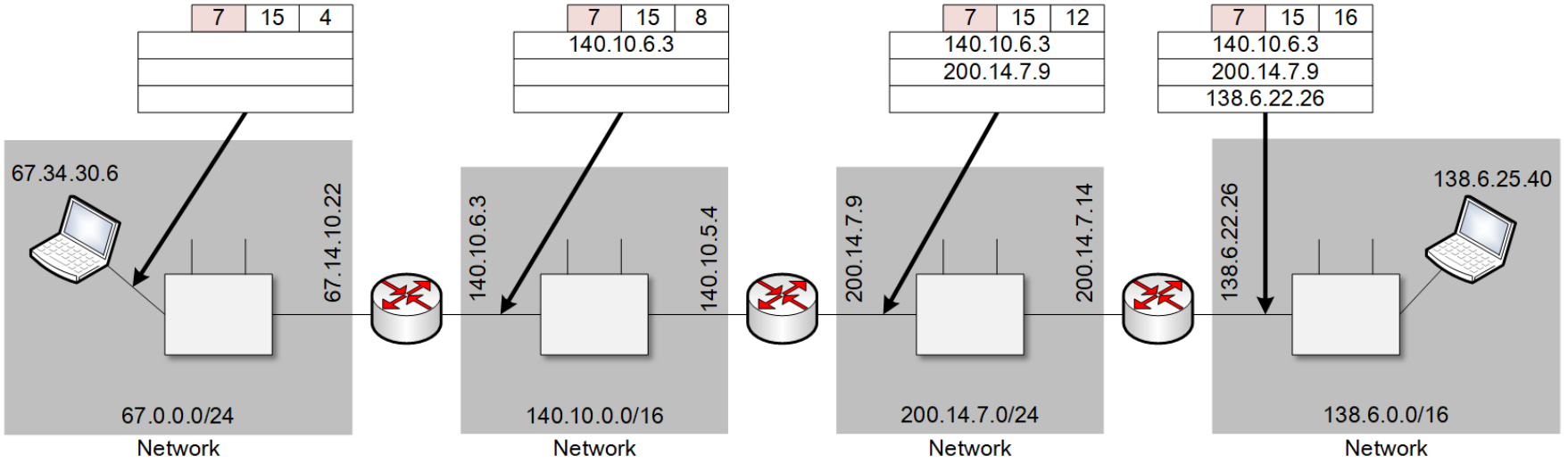


b. Used for padding

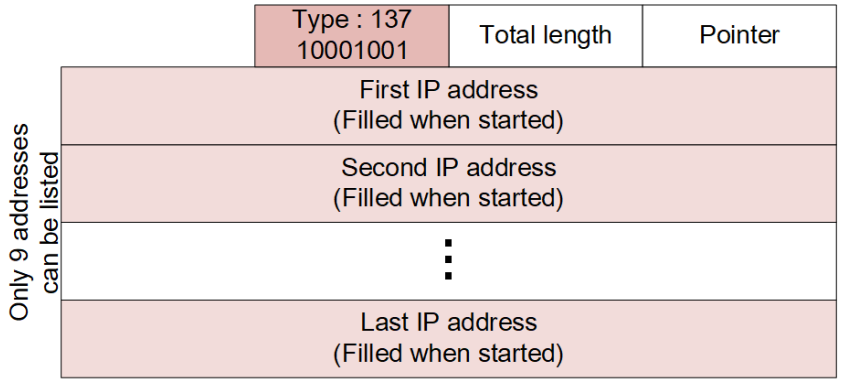
- 옵션 유형(Option Type) (4/14)
  - Multiple-byte 옵션 (1/9)
    - 경로 기록(Record Route) 옵션 (1/2)
      - 데이터그램을 처리한 인터넷 라우터들을 기록하기 위해 사용함
      - 최대 9개의 IP 주소를 기록할 수 있음



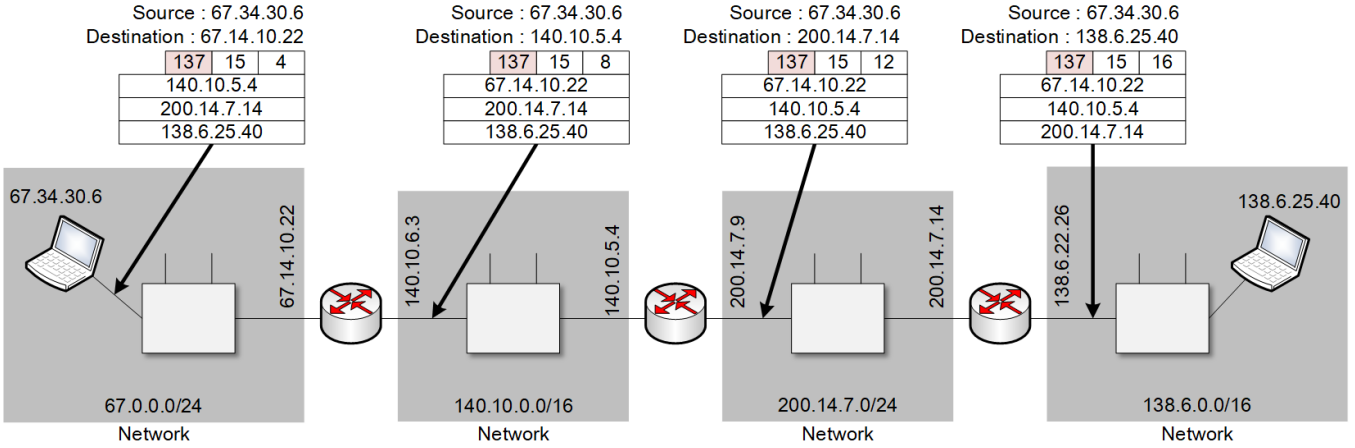
- 옵션 유형(Option Type) (5/14)
  - Multiple-byte 옵션 (2/9)
    - 경로 기록(Record Route) 옵션 (2/2)
      - 경로 기록 예시
        1. 데이터그램을 처리하는 라우터는 포인터 필드의 값과 길이 필드의 값을 비교함
        2. 포인터 값이 길이 값보다 크지 않으면, 라우터는 자신의 IP 주소를 다음 빈 필드에 기록함
        3. 포인터 값을 4만큼 증가시키고 다음 경로로 이동 시킴



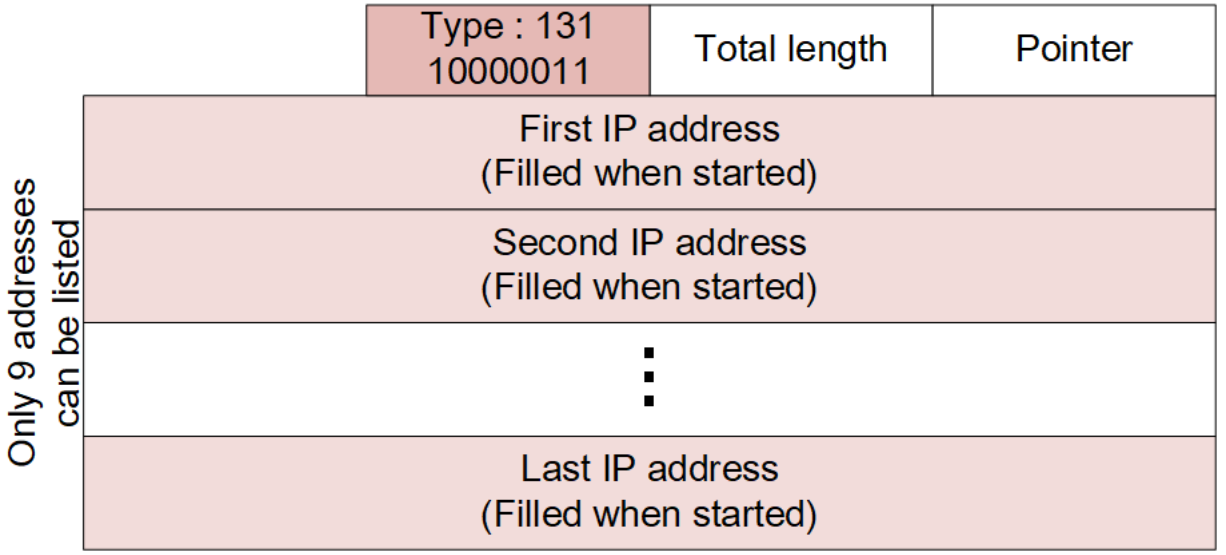
- 옵션 유형(Option Type) (6/14)
  - Multiple-byte 옵션 (3/9)
    - 엄격한 발신지 경로(Strict Source Route) 옵션 (1/2)
      - 데이터그램이 인터넷에서 거쳐야 할 경로를 미리 지정하기 위해 사용함
      - 경로를 미리 지정하여 최소 지연, 최대 처리율, 신뢰성 등을 가지는 경로를 선택할 수 있음
        - e.g., 송신자의 데이터그램이 공격자의 네트워크를 통과하지 않도록 함
      - 데이터그램이 리스트에 없는 주소를 방문하거나, 리스트의 주소를 방문하지 않은 경우에는 폐기되며 오류 메시지가 발송됨



- 옵션 유형(Option Type) (7/14)
  - Multiple-byte 옵션 (4/9)
    - 엄격한 발신지 경로(Strict Source Route) 옵션 (2/2)
      - 엄격한 발신지 경로 예시
        1. 라우터는 포인터 필드의 값과 길이 필드의 값을 비교함
           - 포인터 값보다 길이 값이 크면, 미리 저장된 라우터를 모두 방문했음
        2. 포인터 값이 길이 값보다 크지 않으면, 포인터가 가리키고 있는 IP 주소와 라우터에서 데이터그램이 들어온 IP 주소를 비교함
        3. 두 주소가 같으면, 현재의 IP 주소를 나가는 IP 주소로 대체함
        4. 포인터 값을 4만큼 증가시키고 다음 경로로 이동 시킴



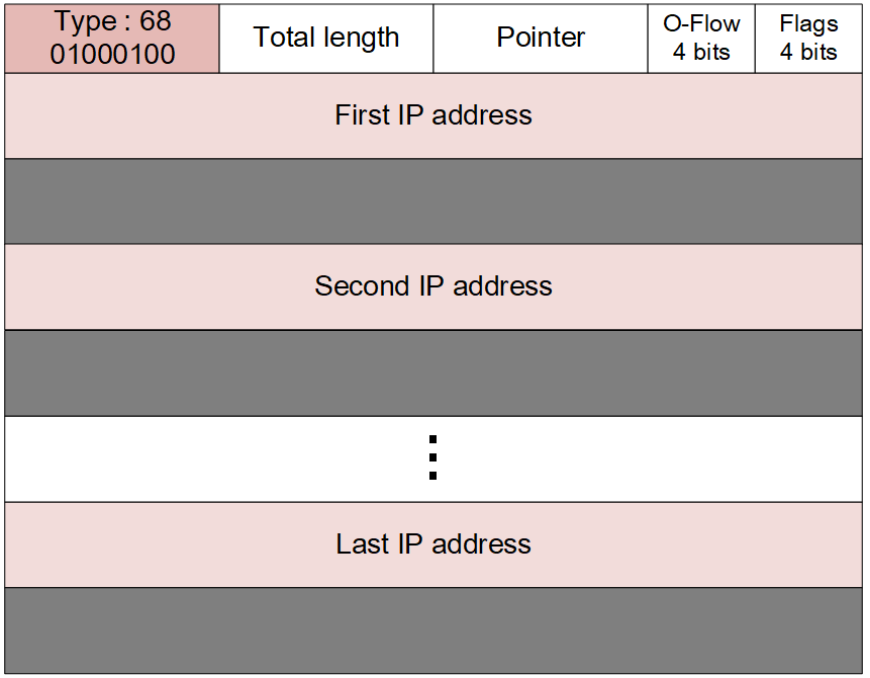
- 옵션 유형(Option Type) (8/14)
  - Multiple-byte 옵션 (5/9)
    - 느슨한 발신지 경로(Loose Source Route) 옵션
      - 데이터그램이 인터넷에서 거쳐야 할 경로를 미리 지정하기 위해 사용함
      - 엄격한 발신지 경로와 달리 리스트에 없는 라우터도 방문할 수 있음
        - 옵션 필드에 적힌 경로들 외의 중간 경로는 유동적으로 선택될 수 있음



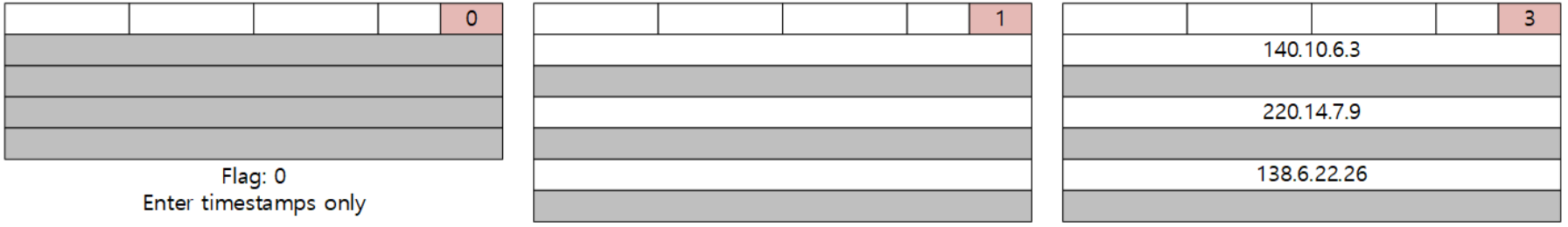
IPv4

- 옵션 유형(Option Type) (9/14)
  - Multiple-byte 옵션 (6/9)
    - 타임스탬프(Timestamp) 옵션 (1/4)
      - 개요
        - 타임스탬프
          - 네트워크 패킷이 라우터를 통과하는 등의 사건이 발생한 시점을 기록한 값
        - 장점
          - 네트워크 지연/경로 추적을 통해 이상 징후를 탐지하여 가용성을 확보할 수 있음
          - 경유 라우터 기록을 통해 경로 무결성 검증이 가능함
        - 단점
          - 경로/시간 정보가 노출되므로 공격자가 이를 기반으로 사용자 활동패턴을 추적할 수 있음
          - 타임스탬프 값이 변조될 경우 잘못된 경로/지연 정보로 신뢰성이 저하됨

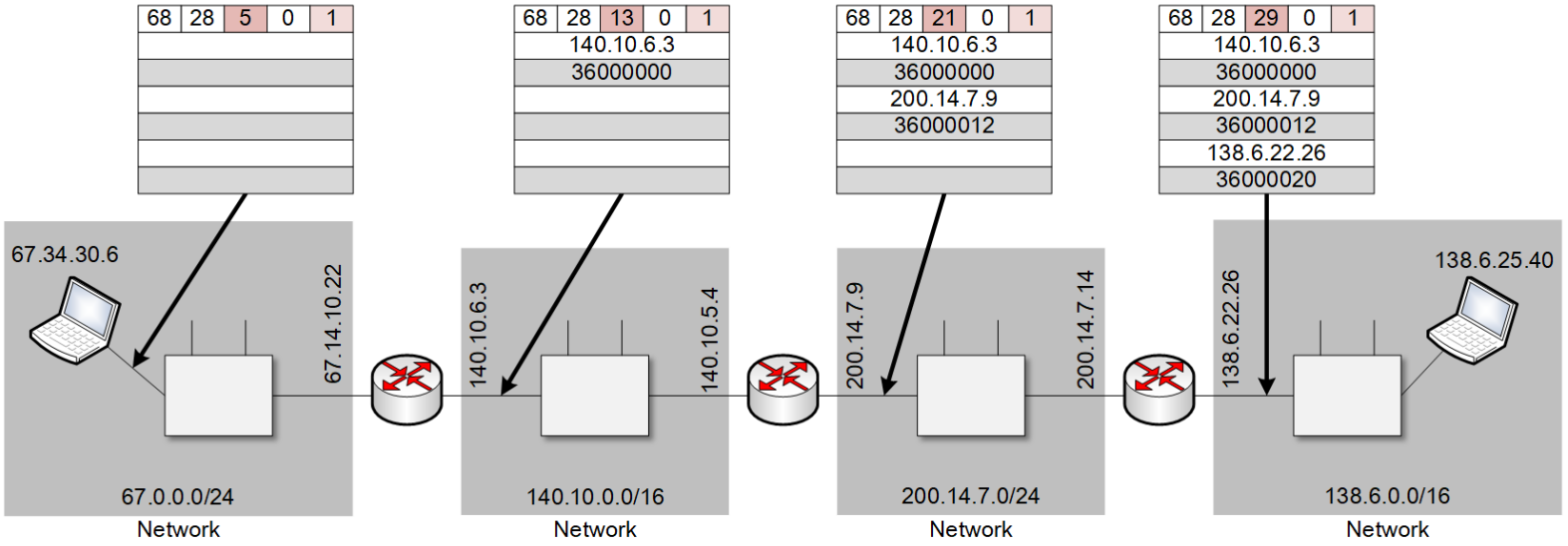
- 옵션 유형(Option Type) (10/14)
  - Multiple-byte 옵션 (7/9)
    - 타임스탬프(Timestamp) 옵션 (2/4)
      - 라우터가 데이터그램을 처리하는 시간을 기록하기 위해 사용함
      - 세계 표준시(Universal Time) 자정으로부터의 millisecond 단위로 표기함



- 옵션 유형(Option Type) (11/14)
  - Multiple-byte 옵션 (8/9)
    - 타임스탬프(Timestamp) 옵션 (3/4)
      - 구조
        - 오버플로우(O-Flow) 필드
          - 필드가 더 이상 없어 타임스탬프를 기록하지 못한 라우터의 수를 기록함
        - 플래그(Flags) 필드
          - 방문한 라우터가 데이터그램에 수행할 동작을 정의함
          - 0이면 각 라우터는 주어진 필드에 타임스탬프만 추가함
          - 1이면 라우터는 출력 인터페이스 IP 주소와 타임스탬프를 기록함
          - 3이면 라우터는 주어진 IP 주소와 입력 IP 인터페이스 주소를 비교하고, 두 주소가 같으면 IP 주소에 출력 인터페이스 주소를 덮어쓰고 타임스탬프를 추가함



- 옵션 유형(Option Type) (12/14)
  - Multiple-byte 옵션 (9/9)
    - 타임스탬프(Timestamp) 옵션 (4/4)
      - 타임스탬프 예시
        1. 라우터는 플래그 값을 확인함
           - 플래그 값이 0이라면 포인터가 가리키는 필드에 타임스탬프 값만 기록함
        2. 플래그 값이 1이면, 포인터가 가리키는 필드에 IP 주소와 타임스탬프 값을 기록함
        3. 포인터 값을 8만큼 증가시키고 데이터그램을 다음 경로로 이동 시킴



- 옵션 유형(Option Type) (13/14)
  - 예제 7.10

6개의 옵션 중 어느 것이 각 단편에 복사되어야 하는가?

- 각 옵션 필드의 왼쪽 첫 번째 비트를 통해 알 수 있음
  - 엄격한 발신지 경로와 느슨한 발신지 경로만 각 단편에 복사됨
  - 나머지 옵션은 복사되지 않음

옵션 유형	옵션 값
무연산	00000001
옵션 종료	00000000
경로 기록	00000111
엄격한 발신지 경로	10001001
느슨한 발신지 경로	10000011
타임스탬프	01000100

IETF RFC 791에 정의된 옵션 값

- 옵션 유형(Option Type) (14/14)
  - 예제 7.11

6개의 옵션 중 어떤 것이 데이터그램 제어에 사용되고, 어떤 것이 디버깅과 관리에 사용되는가??

- 각 옵션 필드의 왼쪽에서 두 번째와 세 번째 비트를 통해 알 수 있음
  - 10 값을 갖는 타임스탬프는 디버깅과 관리에 사용 됨
  - 00 값을 갖는 나머지 다섯가지 유형은 데이터그램 제어에 사용 됨

옵션 유형	옵션 값
무연산	00000001
옵션 종료	00000000
경로 기록	00000111
엄격한 발신지 경로	10001001
느슨한 발신지 경로	10000011
타임스탬프	01000100

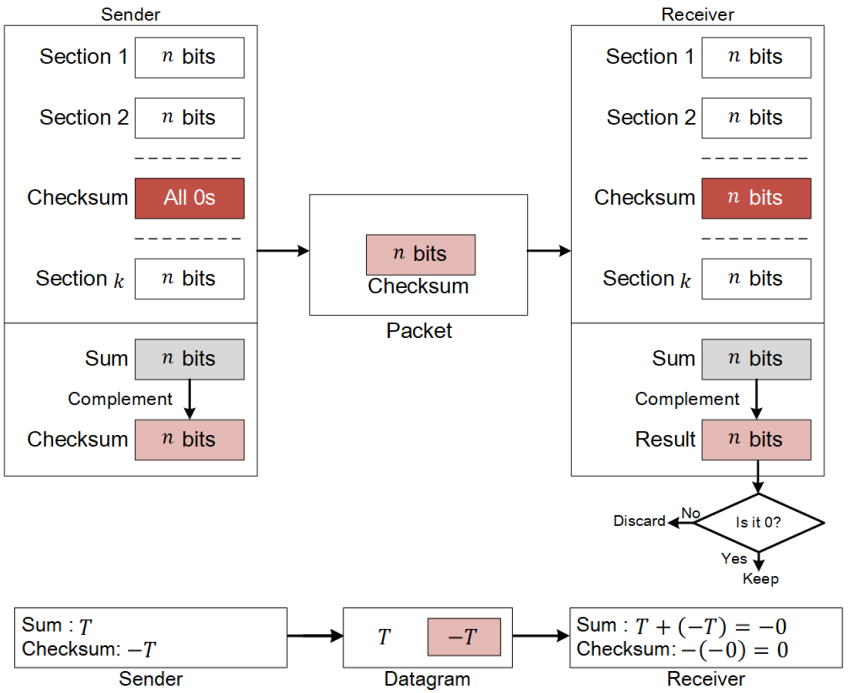
IETF RFC 791에 정의된 옵션 값

IPv4

- 검사합(Checksum) (1/5)
  - 정의
    - 패킷의 전달 중에 발생할 수 있는 오류를 검출하기 위해 사용하는 값
  - 검증 과정
    1. 송신자는 검사합을 계산해 패킷과 함께 전송함
    2. 수신자는 검사합을 포함하고 있는 전체 패킷에 대해 같은 계산을 반복함
       - 두 결과가 일치하면 패킷을 받아들임
       - 두 결과가 일치하지 않으면 패킷을 폐기함

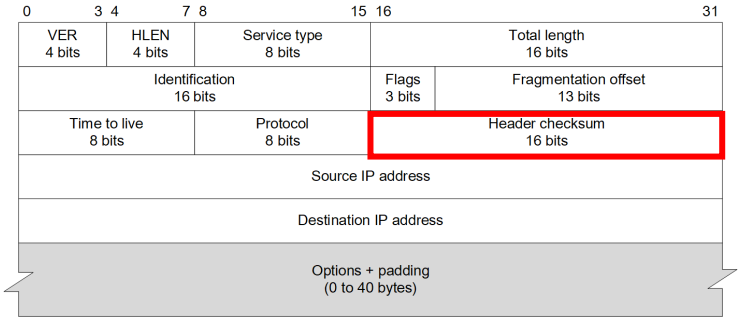
- 검사합(Checksum) (2/5)
  - 검사합 계산
    - 송신자
      1. 송신 할 패킷을 k개의 n비트 단편으로 나눔
      2. 모든 단편을 1의 보수 덧셈을 사용하여 합함
      3. 합에 대한 1의 보수가 검사합이 됨
    - 수신자
      1. 수신된 패킷을 k개의 단편으로 나눔
      2. 모든 단편을 전부 1의 보수 덧셈으로 합함
      3. 합에 대한 1의 보수를 구함
         - 결과가 0이면 패킷 수신
         - 결과가 0이 아니면 패킷 수신 거부

*1의 보수 덧셈
이진수를 더할 때 자릿수를 넘긴 자리올림(carry-out)을 다시 하위 비트에 더해주는 덧셈 방식



IPv4

- 검사합(Checksum) (3/5)
  - IP 패킷의 검사합
    - 검사합 필드
      - 데이터그램의 전송 중 발생할 수 있는 오류를 검출하기 위한 16비트 필드
    - 과정
      1. 헤더를 16비트의 단편으로 나눔
      2. 모든 단편의 합을 구함
      3. 합에 대한 보수를 구해 검사합 필드에 삽입



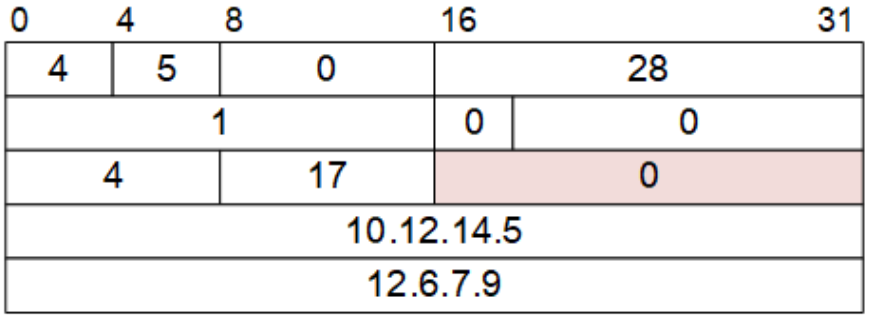
    - 특징
      - IP 패킷 내의 헤더만 포함하고, 데이터는 포함하지 않음
        - IP 데이터그램 내의 데이터를 캡슐화하는 상위 계층의 프로토콜은 데이터를 포함한 전체 패킷에 대한 검사합을 가지고 있음
        - 헤더는 라우터를 방문할 때마다 변경되지만 데이터는 변경되지 않음

- 검사합(Checksum) (4/5)
  - 예제 7.17

다음 IP 데이터그램의 송신지에서의 검사합을 계산하라.

- 16비트로 나누어 계산

4, 5 / 0	0100 0101	0000 0000
28	0000 0000	0001 1100
1	0000 0000	0000 0001
0 / 0	0000 0000	0000 0000
4 / 17	0000 0100	0001 0001
0	0000 0000	0000 0000
10.12	0000 1010	0000 1100
14.5	0000 1110	0000 0101
12.6	0000 1100	0000 0110
7.9	0000 0111	0000 1001
sum	0111 0100	0100 1110
Checksum	1000 1011	1011 0001
	(= 35761)	

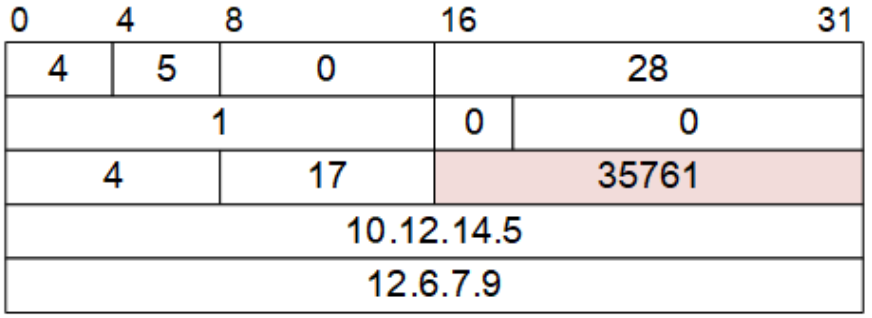


- 검사합(Checksum) (5/5)
  - 예제 7.18

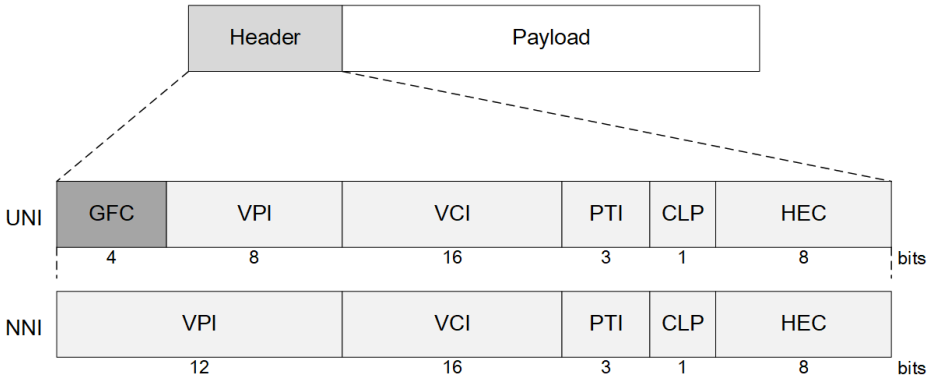
다음 IP 데이터그램의 수신지에서의 검사합을 검증하라.

- 16비트로 나누어 계산

4, 5 / 0	0100 0101	0000 0000
28	0000 0000	0001 1100
1	0000 0000	0000 0001
0 / 0	0000 0000	0000 0000
4 / 17	0000 0100	0001 0001
Checksum Value (35761)	1000 1011	1011 0001
10.12	0000 1010	0000 1100
14.5	0000 1110	0000 0101
12.6	0000 1100	0000 0110
7.9	0000 0111	0000 1001
sum	1111 1111	1111 1111
Result	0000 0000	0000 0000



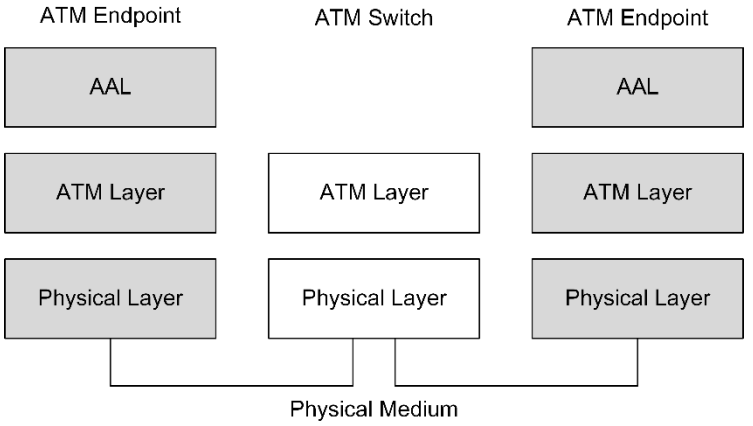
- ATM 상 IP (1/6)
  - ATM 셀
    - 정의
      - ATM 네트워크 상에서 데이터를 전달하는 최소 기본 단위
    - 구조
      - 5바이트의 헤더와 48바이트의 데이터로 구성됨
      - 헤더는 UNI(User-Network Interface)와 NNI(Network-Network Interface)환경에 따라 차이가 존재함



name	size	Description
GFC (Generic Flow Control)	4bits	UNI 에서만 사용 하며, 사용자와 네트워크 사이 흐름을 제어하는데 사용
VPI (Virtual Path Identifier)	8~12bits	가상 경로를 식별하는 번호
VCI (Virtual Channel Identifier)	16bits	가상 채널을 식별하는 번호
PT (Payload Type)	3bits	페이로드가 어떤 데이터인지 구분
CLP (Cell Loss Priority)	1bits	셀 손실 우선순위 표시
HEC (Header Error Control)	8bits	ATM 헤더 전용 오류 검출/수정 코드

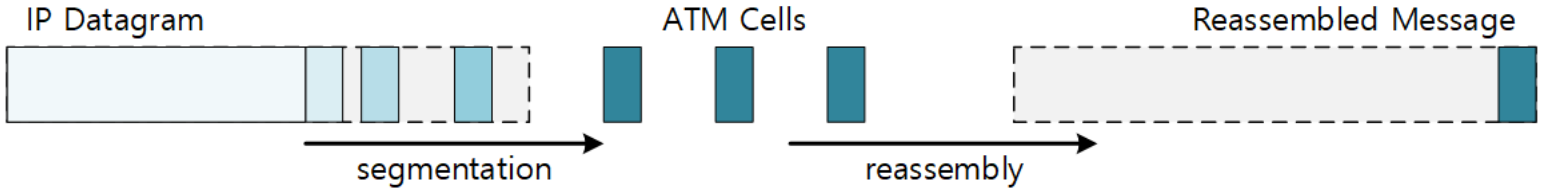
IPv4

- ATM 상 IP (2/6)
  - AAL(Application Adaptation Layer)
    - 개요
      - ATM에서 사용하는 응용 적응 계층
      - 상위 계층으로부터 전송을 받아 고정된 크기의 ATM 셀로 매핑함



    - 종류
      - AAL1
        - 고정 비트율 실시간 서비스(음성, 회선)를 지원하며, 타이밍 복원 기능을 제공함
      - AAL2
        - 가변 비트율 실시간 서비스(압축 음성, 영상)에 적합하며, 작은 단위 다중화로 대역폭 효율을 높임
      - AAL3, 4
        - 데이터 통신용으로 오류 제어, 다중화 기능은 풍부하나, 오버헤드가 커 비효율적임
      - AAL5
        - IP 등 데이터 전송에 사용되며, 구조가 단순하고 오버헤드가 최소화됨

- ATM 상 IP (3/6)
  - AAL5 (1/2)
    - 정의
      - IP 데이터그램을 ATM(Asynchronous Transfer Mode) 네트워크 상에서 전송하기 위해 사용하는 ATM 적응 계층 프로토콜
    - 특징
      - 인터넷에서 사용되는 유일한 AAL임
      - 하나의 IP 데이터그램을 여러 ATM 셀로 분할하더라도, 수신 측에서 셀들을 모아 하나의 메시지로 처리함
        - 연결 지향 방식을 사용하여 주소 정보나 순서와 같은 다른 헤더 정보를 제공하지 않고 셀 순서를 보장함
        - 패딩과 트레일러만 IP 데이터그램에 추가됨

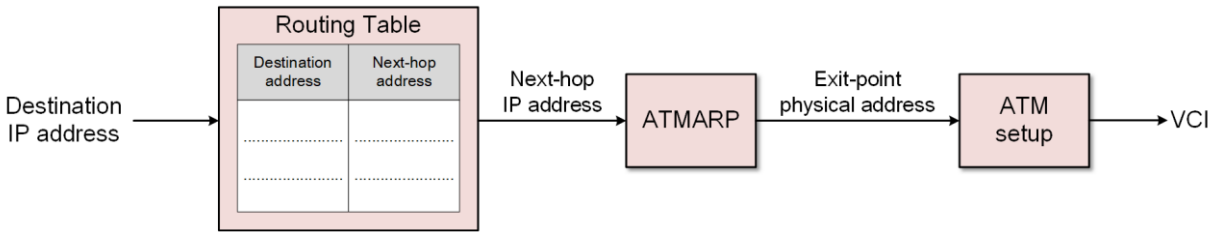


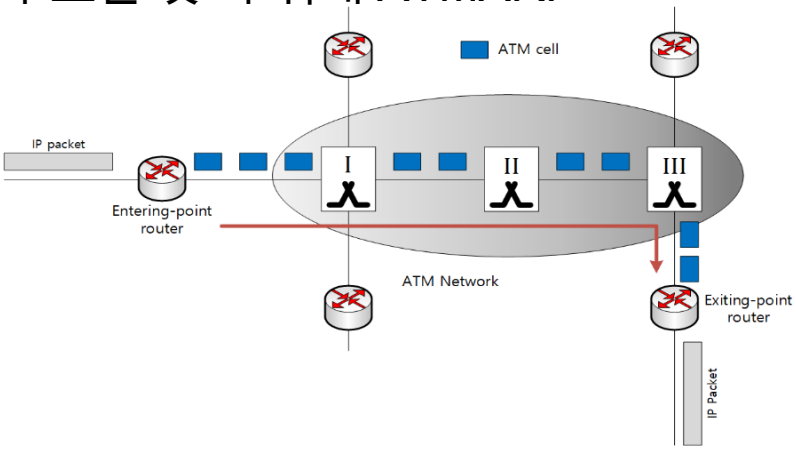
- ATM 상 IP (4/6)
  - AAL5 (2/2)
    - 동작 과정
      1. 65,536바이트 이하의 IP 패킷만을 받아들임
      2. 패딩과 8바이트의 트레일러를 추가함
         - 트레일러의 위치가 마지막 셀의 마지막 8바이트가 되도록 패딩을 추가함
      3. 메시지를 48바이트의 단편으로 만들어 ATM 계층으로 전달함
    - 사용 이유
      - 헤더가 집중된 하나의 ATM 셀의 효율성은 28/53 $\approx 52\%$ 로 낮음
        - IP 헤더와 ATM 헤더를 한 셀에 담으면 데이터는 28바이트만 전송됨

          $\because 53 - 20 - 5 = 28$
      - AAL5는 IP 데이터그램의 헤더를 여러 개의 셀로 분산시켜 전송하므로, 모든 셀의 데이터 활용도가 균등해져 전송 효율이 향상됨

- ATM 상 IP (5/6)
  - 셀 라우팅 (1/2)
    - 정의
      - ATM 셀이 목적지까지 전달되도록 경로를 설정하는 과정
    - 주소
      - IP 주소
        - IP 계층에서 라우터의 주소를 정의
        - ATM 네트워크와 라우터에 정의된 IP 주소는 무관함
      - 물리 주소
        - ATM 네트워크 관리자에 의해 정의된 20바이트의 주소
        - ATM 네트워크와 연관되어 있으며 인터넷과는 무관함
        - 연결 설정 동안 사용됨
      - 가상 회선 식별자(VCI, Virtual Circuit Identifier)
        - ATM 네트워크의 교환기가 라우팅에 사용하는 주소
        - 데이터 전송에 사용됨

- ATM 상 IP (6/6)
  - 셀 라우팅 (2/2)
    - 주소 결합(Address Binding)
      - 목적
        - ATM 네트워크에서 셀을 라우팅하기 위한 VCI를 목적지 IP 주소로부터 결정하기 위함
      - 과정
        1. 진입점 라우터는 IP 데이터그램을 수신함
           - 진출점 라우터인 다음 라우터 IP 주소를 찾기 위해 목적지 주소와 라우팅 테이블을 사용함
        2. 진입점 라우터는 진출점 라우터의 물리 주소를 찾기 위해 ATMARP 프로토콜의 서비스를 사용함
        3. 가상 회선 식별자가 물리 주소에 결합됨

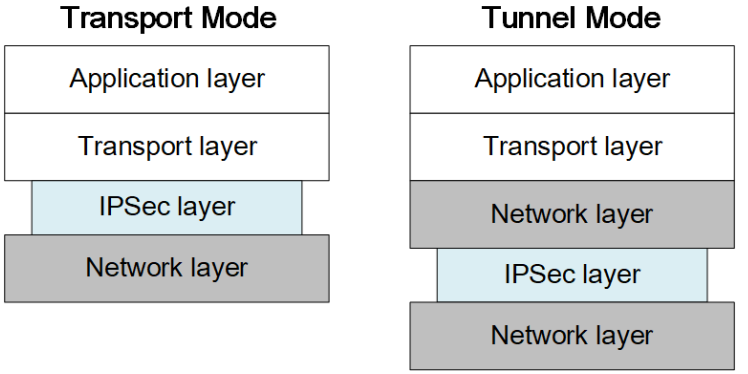




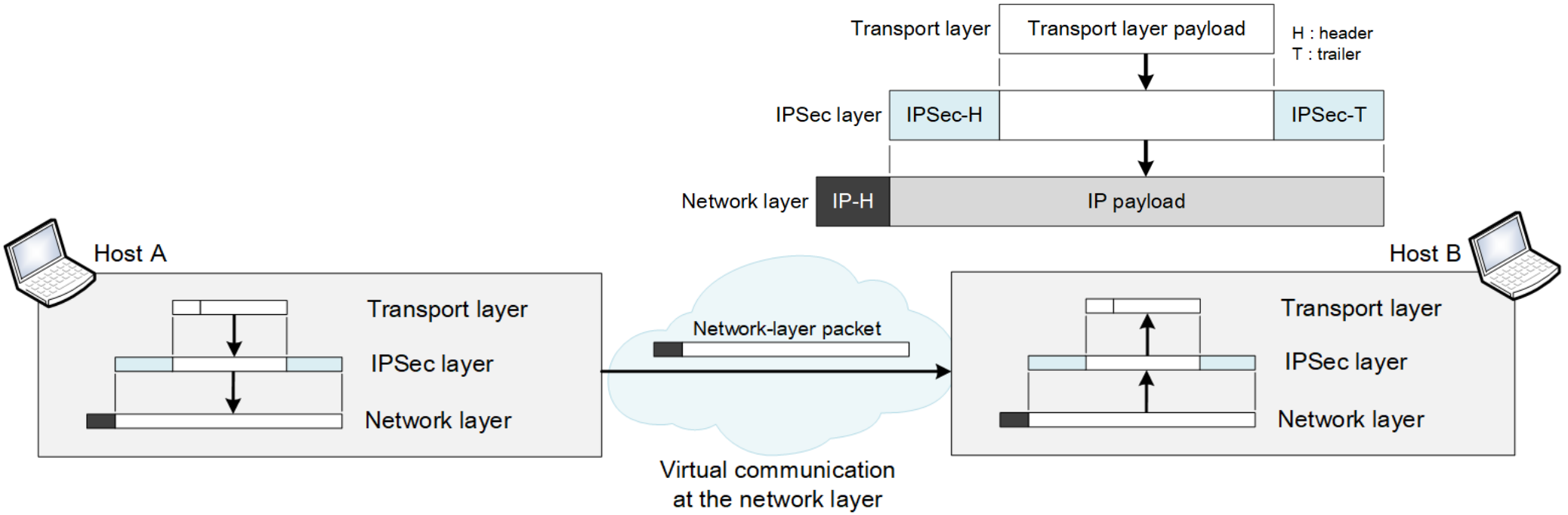
IPv4

- 보안 (1/15)
  - 위협
    - 패킷 스니핑(Packet Sniffing)
      - 침입자가 IP 패킷을 도청하여 복사본을 만들 수 있음
      - 패킷이 복사되었음을 송신자와 수신자가 탐지하기 어려움
        - 수동적인 공격으로 공격자가 패킷 내용을 수정하지 않기 때문
      - 패킷 암호화를 통해 침입자가 내용을 알아낼 수 없도록 무력화 가능
    - 패킷 수정(Packet Modification)
      - 공격자가 패킷을 가로채고 내용을 고친 뒤 새로운 패킷을 수신자에게 보낼 수 있음
      - 데이터 무결성 메커니즘을 사용하여 탐지할 수 있음
    - IP 스푸핑(IP Spoofing)
      - 공격자가 다른 사람으로 위장하여 다른 컴퓨터의 발신지 주소를 포함한 IP 패킷을 생성할 수 있음
      - 발신지 인증(Origin Authentication) 메커니즘을 사용하여 방지할 수 있음

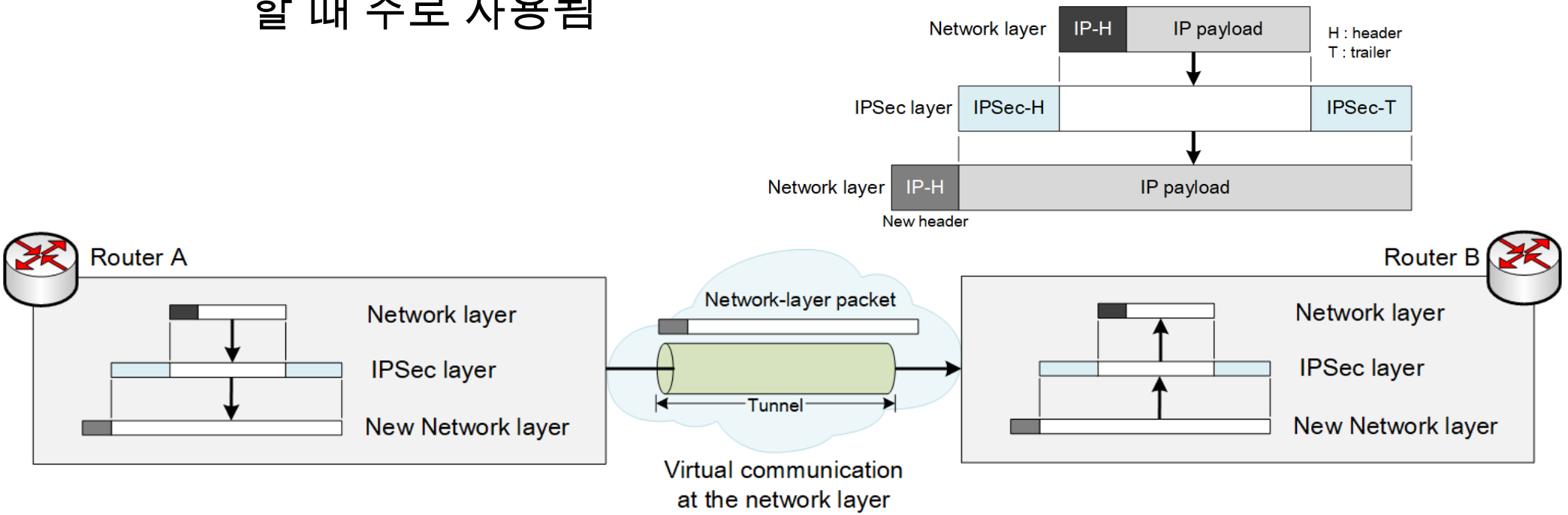
- 보안 (2/15)
  - IPSec (IP Security) (1/4)
    - 정의
      - 네트워크 계층에서 보안을 위한 인증과 기밀성을 갖는 패킷을 생성하도록 하는 프로토콜 집합
    - 필요성
      - IP는 전송되는 데이터에 대해 암호화를 제공하지 않으므로 패킷의 기밀성을 보장할 수 있는 기능이 필요함
      - IP는 무결성 검증 수단과 인증 기능이 존재하지 않으므로 데이터의 무결성이나 송/수신자의 신원을 인증할 수 있는 보안 기능이 필요함
    - 모드
      - 전송 모드(Transport Mode)
      - 터널 모드(Tunnel Mode)



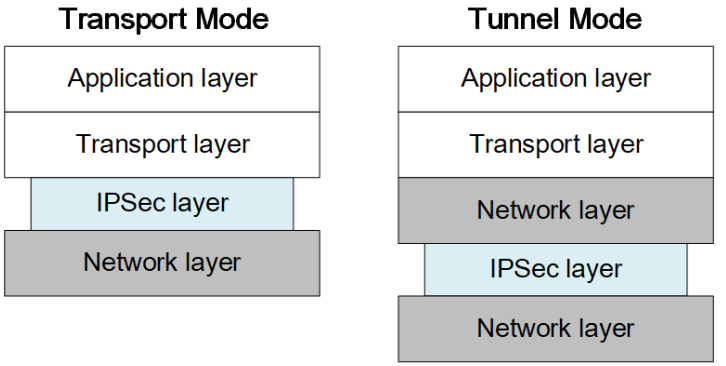
- 보안 (3/15)
  - IPSec (IP Security) (2/4)
    - 모드 - 전송 모드
      - 전송 계층에서 네트워크 계층으로 전달되는 모든 것을 보호함
        - IP 헤더는 보호하지 않고, 전송 계층에서 온 정보만 보호함
      - 호스트-호스트 간에 데이터 보호를 필요로 할 때 주로 사용됨



- 보안 (4/15)
  - IPSec (IP Security) (3/4)
    - 모드 - 터널 모드
      - 전체 IP 패킷을 보호함
        - IP 패킷 전체에 새로운 IP 헤더를 추가하여 패킷을 보호함
      - 라우터-라우터 혹은 라우터-호스트 간에 데이터 보호를 필요로 할 때 주로 사용됨

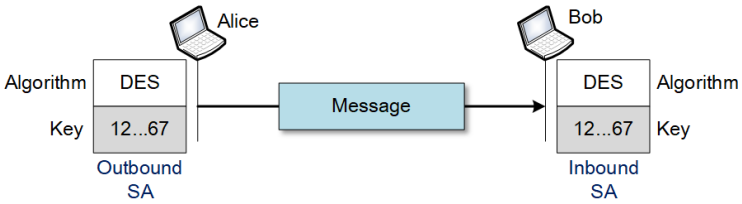


- 보안 (5/15)
  - IPSec (IP Security) (4/4)
    - 모드 – 비교
      - 전송 모드
        - 전송 계층과 네트워크 계층 사이에 존재
      - 터널 모드
        - 두 네트워크 계층 사이에 존재



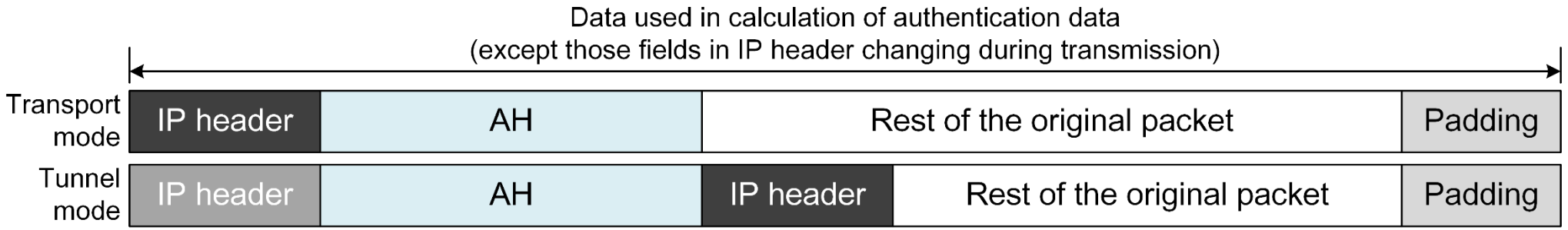
    - 프로토콜
      - 종류
        - 인증 헤더(AH, Authentication Header) 프로토콜
        - 보안 페이로드 캡슐화(ESP, Encapsulating Security Payload) 프로토콜

- 보안 (6/15)
  - IPSec 보안 구성 요소
    - 보안 연관(SA, Security Association)
      - 개요
        - IPSec에서 송신자와 수신자 간의 통신 보안을 위해 설정되는 논리적 연결
        - 단방향으로 정의되므로 양방향 통신을 위해서는 한 쌍의 SA가 필요함
        - 일반적으로 IKE(Internet Key Exchange) 프로토콜을 사용하여 자동으로 생성
      - 종류
        - Outbound SA
          - 송신자가 메시지를 보낼 때 적용되는 보안 연관
        - Inbound SA
          - 수신자가 메시지를 받을 때 적용되는 보안 연관



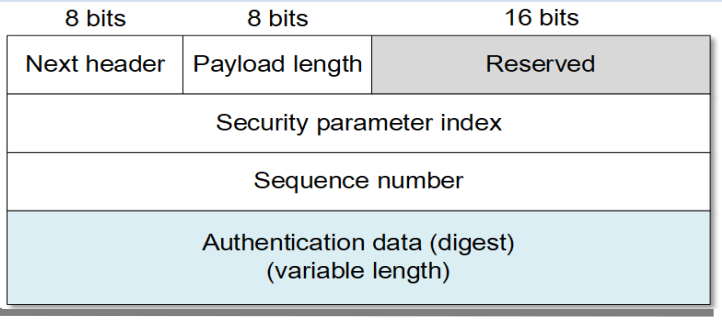
    - 보안 정책(SP, Security Policy)
      - 개요
        - 패킷이 송신되거나 도착될 때 적용되는 보안 유형
        - SA 데이터베이스를 사용하기 전, 호스트가 미리 적용할 정책을 설정해야 함

- 보안 (7/15)
  - IPSec (IP Security) (1/9)
    - 프로토콜 – 인증 헤더(AH) 프로토콜 (1/3)
      - 필요성
        - 발신지 호스트를 인증하기 위함
        - IP 패킷으로 전달되는 페이로드의 무결성을 보장하기 위함
      - 특징
        - 메시지 다이제스트(인증 데이터)를 생성하기 위해 해시함수와 대칭키를 사용함
        - 인증 데이터는 인증 헤더(AH)로 삽입 됨
        - IP 헤더의 프로토콜 필드의 값을 51로 설정함

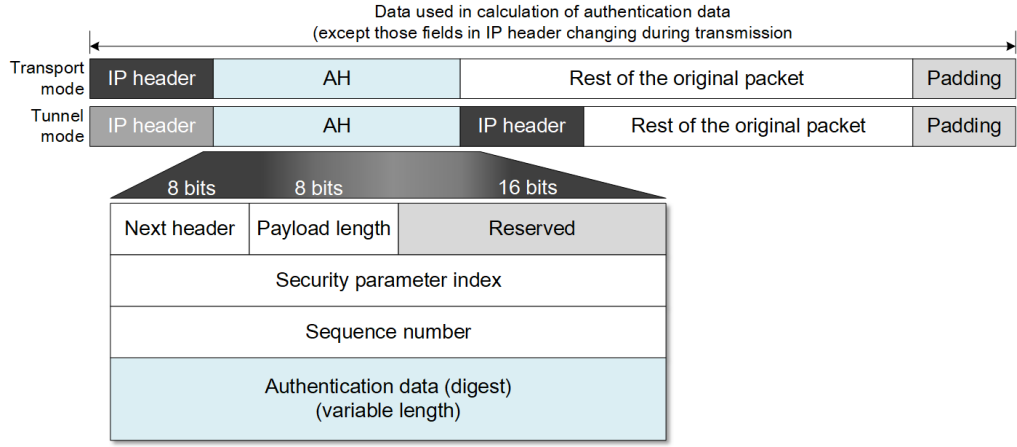


- 보안 (8/15)
  - IPSec (IP Security) (2/9)
    - 프로토콜 – 인증 헤더(AH) 프로토콜 (2/3)
      - 구조

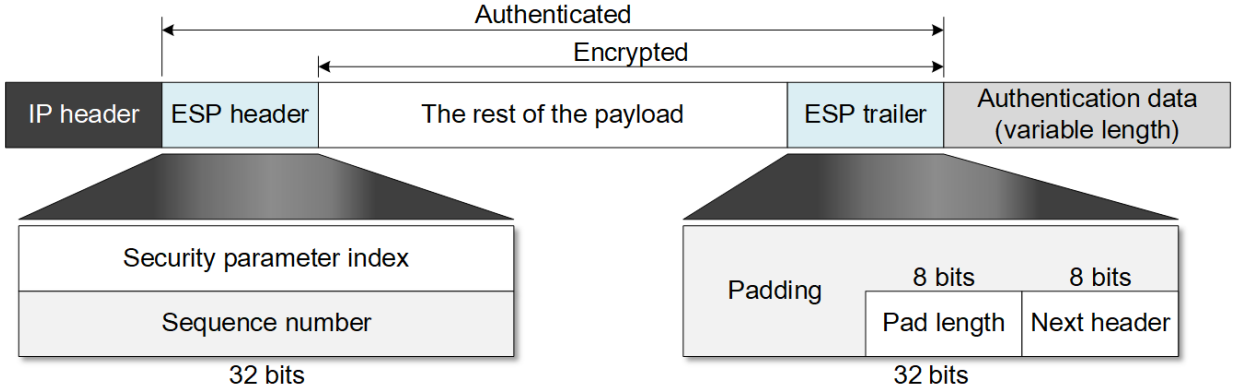
Field name	Description
다음 헤더(Next Header)	• IP 데이터그램에 의해 전달되는 페이로드의 유형을 정의하는 8비트 길이의 필드 • 캡슐화 이전의 IP 헤더의 프로토콜 필드와 동일한 유형을 가짐
페이로드 길이(Payload Length)	• AH의 길이를 4바이트 배수로 정의하는 8비트 길이의 필드
보안 매개변수 색인 (SPI, Security Parameter Index)	• VCI의 역할을 하는 32비트 길이의 필드 • 동일한 보안 연관 동안 보내진 모든 패킷에 대해 동일한 값을 가짐
순서 번호(Sequence Number)	• 데이터그램의 순서를 위한 순서 정보를 제공하는 32비트 길이의 필드 • 순서 번호는 2^{32}에 도달한 이후에는 발생하지 않으며, 새로운 연결이 설정되어야 함
인증 데이터(Authnetication data)	• 전송 도중에 변경된 필드(TTL)를 제외한 완전한 IP 데이터그램에 해시함수를 적용한 메시지 다이제스트를 가지는 가변길이의 필드



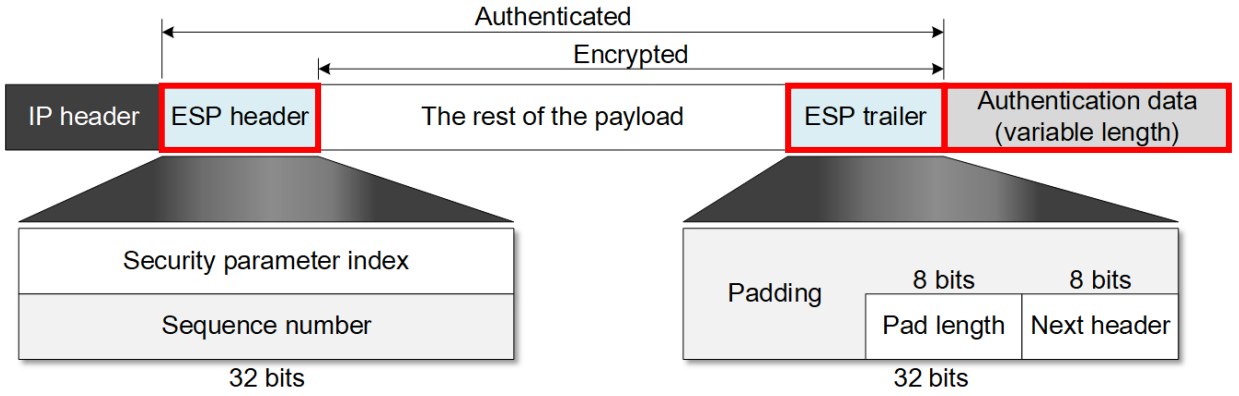
- 보안 (9/15)
  - IPSec (IP Security) (3/9)
    - 프로토콜 – 인증 헤더(AH) 프로토콜(3/3)
      - 인증 헤더 추가 과정
        1. 0으로 설정된 인증 데이터 필드를 가진 페이로드에 인증 헤더가 추가 됨
        2. 특별한 해싱 알고리즘에 대해 전체 길이를 일정하게 만들기 위해 패딩이 추가 될 수 있음
        3. 해싱은 전체 패킷을 기반으로 함
        4. 인증 데이터는 인증 헤더에 삽입 됨
           - 단, 전송 도중에 값이 변경되는 IP 헤더의 필드는 메시지 다이제스트(인증 데이터)의 계산에서 제외함
        5. IP헤더는 프로토콜 필드의 값을 51로 바꾼 뒤에 추가 됨



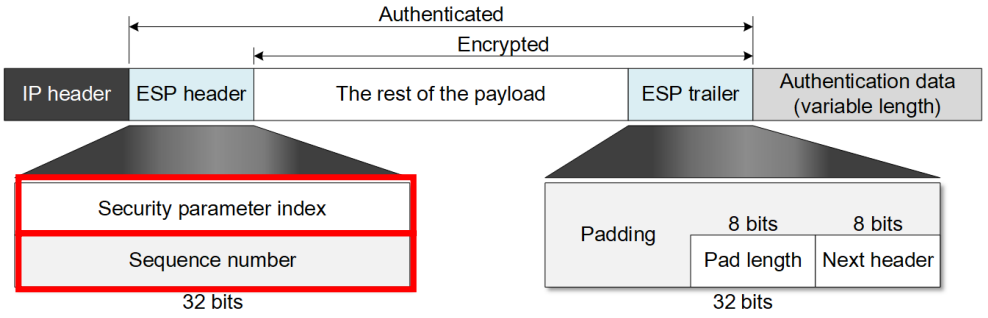
- 보안 (10/15)
  - IPSec (IP Security) (4/9)
    - 프로토콜 – 보안 페이로드 캡슐화(ESP) 프로토콜 (1/4)
      - 필요성
        - 발신지 인증, 무결성, 프라이버시를 제공하기 위함
      - 특징
        - 메시지 다이제스트(인증 데이터)를 생성하기 위해 대칭키 암호화와 선택적으로 인증을 사용함
        - 헤더와 트레일러를 추가하며, 인증 데이터는 패킷의 끝에 추가됨
        - IP 헤더의 프로토콜 필드의 값을 50으로 설정함



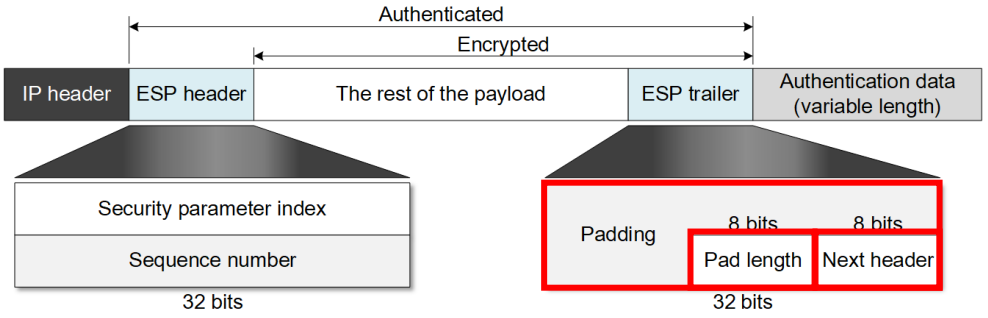
- 보안 (11/15)
  - IPSec (IP Security) (5/9)
    - 프로토콜 – 보안 페이로드 캡슐화(ESP) 프로토콜 (2/4)
      - 구조
        - 인증 데이터(Authentication Data)
          - 인증 구조를 데이터그램의 한 부분으로 적용한 가변 길이의 필드
          - IP헤더를 인증 데이터 계산에 포함하지 않음
        - ESP 헤더(ESP Header)
          - 보안 매개변수 색인(SPI)과 순서 번호를 갖는 필드
        - ESP 트레일러(ESP Trailer)
          - 패드 길이와 다음 헤더를 정의하는 필드



- 보안 (12/15)
  - IPSec (IP Security) (6/9)
    - 프로토콜 – 보안 페이로드 캡슐화(ESP) 프로토콜 (3/4)
      - 구조 – ESP 헤더
        - 보안 매개변수 색인(SPI, Security Parameter Index)
          - VCI의 역할을 하는 32비트 길이의 필드
          - 보안 연관이라는 연결 동안 보내진 모든 패킷에 대해 동일한 값을 가짐
        - 순서 번호(Sequence Number)
          - 데이터그램의 순서를 위한 순서 정보를 제공하는 32비트 길이의 필드
          - 순서 번호는 재전송(Playback)을 방지하기 위해 사용됨
          - 패킷이 재전송되더라도 순서 번호는 반복되지 않음
          - 순서 번호는 2^{32}에 도달한 이후에는 발생하지 않으며, 새로운 연결이 설정되어야 함

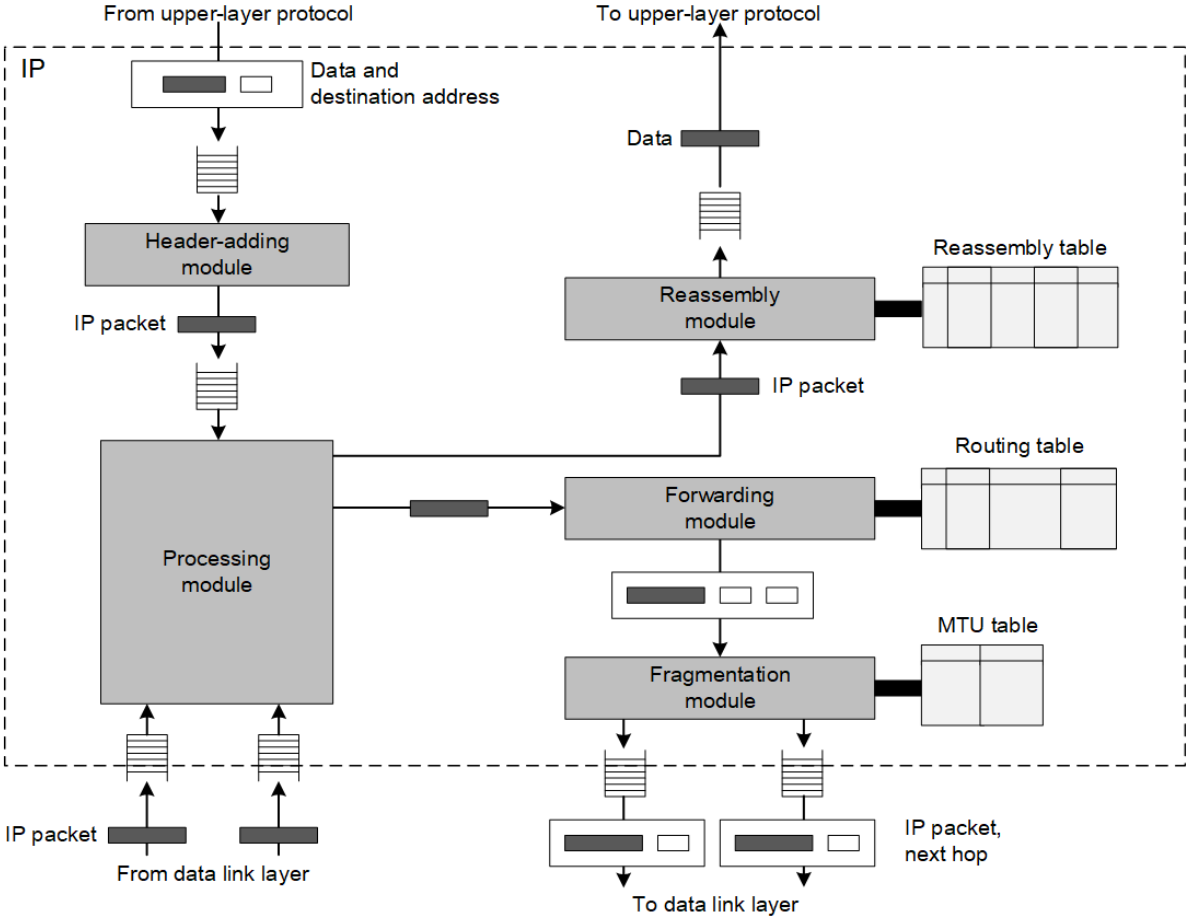


- 보안 (13/15)
  - IPSec (IP Security) (7/9)
    - 프로토콜 – 보안 페이로드 캡슐화(ESP) 프로토콜 (4/4)
      - 구조 – ESP 트레일러
        - 패딩(Padding)
          - 0으로만 구성되며 최대 255바이트의 길이를 갖는 필드
        - 패드 길이(Pad Length)
          - 패딩 바이트 수를 정의하는 8비트 길이의 필드
          - 0부터 255 사이의 값을 가짐
        - 다음 헤더(Next Header)
          - IP 데이터그램에 의해 전달되는 페이로드의 유형을 정의하는 8비트 길이의 필드
          - 캡슐화 이전의 IP 헤더의 프로토콜 필드와 동일한 유형을 가짐
          - IP 데이터그램의 프로토콜 필드 값을 해당 필드에 복사하고, 기존 IP 헤더의 프로토콜 필드에는 ESP를 의미하는 50이 설정됨



IPv4

- IP 패키지
  - 정의
    - IP 패킷을 다음 계층으로 전달하기 위한 처리 과정 전체
  - 구성요소
    - 큐
    - 헤더 추가 모듈
    - 처리 모듈
    - 라우팅 테이블
    - 포워딩 모듈
    - MTU 테이블
    - 단편화 모듈
    - 재조립 모듈
    - 재조립 테이블



IPv4

- IP 패키지 – 구성요소 (1/10)
  - 큐(Queue)
    - 입력 큐
      - 데이터 링크 계층이나 상위 계층 프로토콜로부터 온 데이터그램을 저장함
      - 처리 모듈은 입력 큐로부터 데이터그램을 가져옴
    - 출력 큐
      - 데이터 링크 계층이나 상위 계층 프로토콜로 가는 데이터그램을 저장함
      - 단편화 모듈과 재조립 모듈은 출력 큐에 데이터그램을 삽입함

- IP 패키지 – 구성요소 (2/10)
  - 헤더 추가 모듈(Header-Adding Module)
    - 동작
      - 상위 계층 프로토콜로부터 데이터와 목적지 IP를 받음
      - IP 데이터그램에 있는 데이터를 캡슐화함
      - 검사합을 계산하고 검사합 필드에 추가함
      - 목적에 맞는 큐에 완성된 IP 데이터그램을 전달함
    - 의사코드

```
IP_Adding_Module(data, destination_address)
{
    Encapsulate data in an IP datagram
    Calculate ckecksum and insert it in the checksum field
    Send data to the corresponding queue
    Return
}
```

- IP 패키지 – 구성요소 (3/10)
  - 처리 모듈(Processing Module) (1/2)
    - 동작
      - 입력 큐에서 데이터그램을 제거함
      - 현재 장치의 IP 주소와 패킷의 목적지 주소 필드를 비교하여 패킷이 최종 목적지에 도착하였는지 확인함
        - 최종 목적지인 경우 재조립 모듈로 데이터그램을 보내고 종료함
      - 장치가 라우터인지 확인함
        - 라우터이면 TTL을 1만큼 감소시킴
      - TTL이 0이하인지 확인함
        - 0이하이면 데이터그램을 폐기하고 ICMP 오류 메시지를 전송한 뒤 종료함
      - 데이터그램을 포워딩 모듈에 보냄

- IP 패키지 – 구성요소 (4/10)
  - 처리 모듈(Processing Module) (2/2)
    - 의사코드

```
IP_Processing_Module(datagram)
{
    Remove one datagram from one of the input queues.
    if (destination address matches a local address)    {
        send the datagram to the reassembly module.
        Return.
    }
    if (machine is a router)    {
        Decrement TTL.
    }
    if (TTL less than or equal to zero)    {
        Discard the datagram.
        Send an ICMP error message.
        Return.
    }
    Send the datagram to the forwarding module.
    Return.
}
```

IPv4

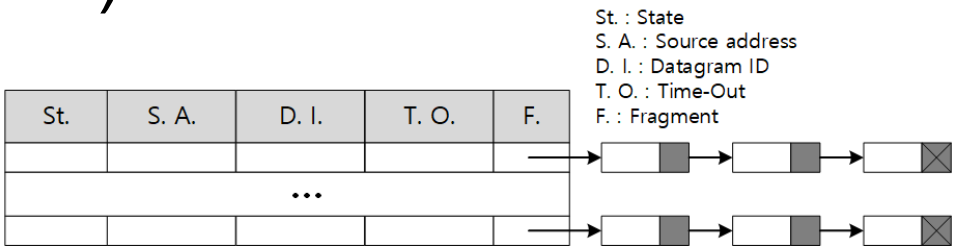
- IP 패키지 – 구성요소 (5/10)
  - 라우팅 테이블(Routing Table)
    - 패킷의 다음 홉 주소를 결정하기 위해 포워딩 모듈에서 사용함
  - 포워딩 모듈(Forwarding Module)
    - 처리 모듈로부터 전달되어야 하는 IP 패킷을 받아 처리함
      - 보내져야할 노드의 주소와 패킷이 보내져야하는 인터페이스의 번호를 찾아서 패킷과 함께 단편화 모듈로 보냄
  - MTU 테이블
    - 단편화 모듈이 특정 인터페이스의 MTU를 찾기 위해 사용
    - 인터페이스와 MTU의 두 열로 구성됨

- IP 패키지 – 구성요소 (6/10)
  - 단편화 모듈(Fragmentation Module) (1/2)
    - 동작
      - 포워딩 모듈로부터 IP 데이터그램을 받음
      - 데이터그램의 크기를 추출하고 네트워크의 MTU보다 큰지 확인함
        - 크기가 MTU보다 큰 경우 D비트 가 설정 되어있는지 확인함
          - D비트가 설정되어 있으면 데이터그램을 폐기하고 ICMP 에러메시지를 보낸 후 종료함
        - D비트가 설정되어 있지 않으면 최대 크기를 계산하고 단편으로 분할함
        - 각 단편에 헤더와 필요한 옵션을 추가하고 전송한 뒤 종료함
      - 데이터그램을 전송하고 종료함

- IP 패키지 – 구성요소 (7/10)
  - 단편화 모듈(Fragmentation Module) (2/2)
    - 의사코드

```
IP_Fragmentation_Module(datagram)
{
    Extract the size of datagram
    if  (size > MTU of the corresponding network) {
        if (D bit is set ) {
            Discard datagram
            Send an ICMP error message
            return
        }
        else {
            Calculate maximum size
            Divide the  segment into fragments
            Add header to each fragment
            Add required options to each fragment
            Send fragment
            return
        }
    }
    else {
        Send the datagram
    }
    Return.
}
```

- IP 패키지 – 구성요소 (8/10)
  - 재조립 테이블(Reassembly table)
    - 재조립 모듈에 의하여 사용됨
    - 필드
      - 상태(State)
        - 재조립 테이블의 사용 상태를 정의함
        - 비어있을 때는 FREE, 사용 중일 때는 IN-USE의 값을 가짐
      - 발신지 IP 주소(Source Address)
        - 데이터그램의 발신지 IP 주소를 정의함
      - 데이터그램 ID(Datagram ID)
        - 데이터그램과 데이터그램에 속하는 단편들을 유일하게 정의하는 번호임
      - 타임아웃(Time-Out)
        - 사전에 설정한 시간 내에 모든 단편이 도착 해야함
      - 단편(Fragments)
        - 재조립 대상인 각 단편들의 정보를 연결 리스트 형태로 관리하는 포인터 필드



- IP 패키지 – 구성요소 (9/10)
  - 재조립 모듈(Reassembly Module) (1/2)
    - 동작
      - 처리 모듈로부터 최종 목적지에 도착한 데이터그램 단편을 받음
      - 단편의 옵셋 값이 0이면서 M비트가 0인지 확인함
        - 두 값이 0이면 데이터그램을 적절한 큐에 보내고 종료함
      - 재조립 테이블에서 항목에 대해 검색함
        - 해당 항목을 찾지 못하면 새로운 항목을 생성함
      - 데이터그램을 연결 리스트에 삽입하고 모든 단편이 도착했는지 확인함
        - 모든 단편이 도착했으면 단편을 재조립하고 상위 계층 프로토콜에 단편을 전송하고 종료함
      - 단편들이 모두 도착하지 않았고, 타임아웃이 발생했는지 확인함
        - 타임아웃이 발생했다면 모든 단편을 폐기하고 ICMP 오류 메시지를 전송하고 종료함

- IP 패키지 – 구성요소 (10/10)
  - 재조립 모듈(Reassembly Module) (2/2)
    - 의사코드

```
IP_Reassembly_Module(datagram)
{
    if  (offset value = 0 AND M = 0) {
        Send datagram to the appropriate queue
        Return
    }
    Search the reassembly table for the entry
    if (entry not found) {
        Create a new entry
    }
    Insert datagram into the linked list
    if (all fragments have arrived) {
        Reassemble the fragment
        Deliver the fragment to upper-layer protocol
        return
    }
    else{
        if (time-out expired) {
            Discard all fragments
            Send an ICMP error message
        }
    }
    Return.
}
```

목 차

- IPv4
- ARP

ARP

- 주소의 종류

*인터 네트워크
여러 개의 서로 다른 네트워크를 연결한 것

  - 논리 주소(Logical Address)
    - 모든 곳에서 유효하고 유일한 *인터 네트워크 주소
    - 네트워크를 연결하는 모든 프로토콜은 논리 주소를 필요로 함
    - 소프트웨어 상에 구현되므로 논리 주소라고 불림
    - e.g., TCP/IP 프로토콜들에서의 논리 주소는 IP 주소라고 부르며 32비트의 길이를 가짐

*NIC(Network Interface Card)
컴퓨터나 기타 장치가 네트워크에 연결되도록 해주는 하드웨어 장치

  - 로컬 주소(Local Address)
    - 로컬 네트워크 내에서만 유효한 주소
    - 물리적인 레벨에서 호스트와 라우터를 인식하기 위해 필요함
    - 하드웨어 내에 구현되므로 물리 주소라고도 불림
    - e.g., 이더넷이나 토큰링의 물리 주소는 호스트나 라우터 내에 설치된 *NIC에 들어있으며 48비트의 길이를 가짐

ARP

- 주소 변환
  - 필요성
    - 하나의 물리적인 네트워크상에 두 개의 다른 네트워크 계층 프로토콜이 동시에 사용될 때 두 종류의 주소가 모두 필요함
    - 두 종류를 함께 사용하기 위해 두 주소간 변환이 필요함
  - 종류
    - 정적 변환(Stactic Mapping)
    - 동적 변환(Dynamic Mapping)

ARP

- 주소 변환 – 종류 (1/2)
  - 정적 변환
    - 논리 주소와 물리 주소를 연관시키는 테이블을 사용하여 변환함
      - 테이블은 네트워크 상의 각 장치 내에 저장되어 있음
    - 한계점
      1. 장치의 NIC를 교체하여 새 물리 주소를 갖는 경우
      2. LocalTalk와 같은 LAN에서 컴퓨터가 부팅될 때마다 물리 주소가 변하는 경우
      3. 이동 컴퓨터와 같이 하나의 물리적 네트워크에서 다른 네트워크로 이동하여 물리 주소가 변하는 경우
      - 물리 주소가 변하는 한계를 극복하기 위해 정적 변환 테이블은 주기적으로 갱신되어야 하는데, 이는 네트워크에 매우 큰 오버헤드가 됨

ARP

- 주소 변환 – 종류 (2/2)
  - 동적 변환
    - 장치가 다른 장치의 논리 주소를 알고 있는 경우 프로토콜을 사용하여 물리 주소를 알아냄
    - 관련 프로토콜 종류
      - 주소 변환 프로토콜(ARP, Address Resolution Protocol)
        - 논리 주소를 물리 주소로 동적 변환함
      - 역 주소 변환 프로토콜(RARP, Reverse Address Resolution Protocol)
        - 물리 주소를 논리 주소로 동적 변환함
        - 현재는 *DHCP로 대체되어 사용되지 않음

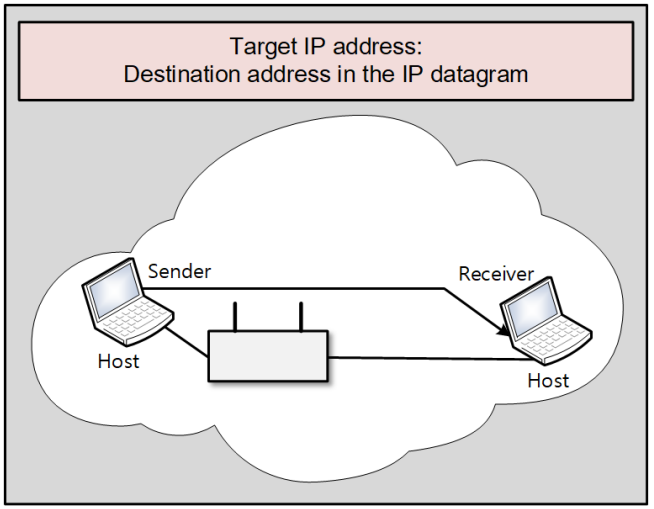
*DHCP(Dynamic Host Configuration Protocol) 네트워크에 연결된 장치에게 자동으로 IP 주소를 할당해주는 프로토콜

ARP

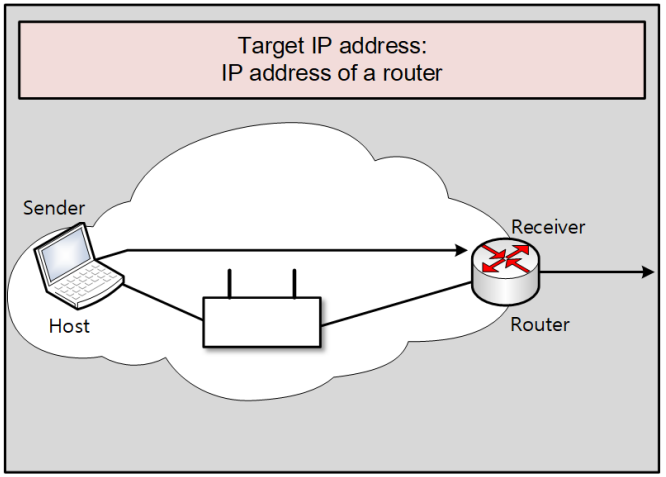
- 주소 변환 프로토콜(ARP) (1/11)
  - 개요
    - IP 주소와 MAC 주소 사이의 매핑을 수행하는 프로토콜
    - 호스트가 목적지 IP 주소만 알고 있을 때, 해당 IP의 MAC 주소를 요청/응답으로 획득함
  - 필요성
    - 패킷 전달을 위해서는 논리적 주소인 IP 주소가 아니라 물리적 주소인 MAC 주소가 반드시 필요함
    - 송신자가 IP 주소를 알고 있을 때, 해당 주소를 MAC 주소로 변환하는 과정이 필요함
  - 특징
    - IPv4 환경에서만 사용됨
    - 브로드캐스트를 기반으로 동작함

ARP

- 주소 변환 프로토콜(ARP) (2/11)
  - ARP가 사용되는 경우 (1/2)
    - 한 호스트가 같은 네트워크 상에 있는 다른 호스트에게 패킷을 전송하는 경우
      - 데이터그램 헤더 내의 목적지 IP 주소를 물리 주소로 변환해야 함

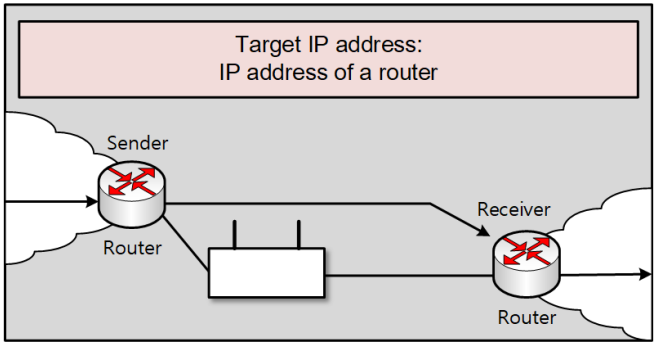


    - 한 호스트가 다른 네트워크 상에 있는 다른 호스트에게 패킷을 전송하는 경우
      - 라우팅 테이블의 다음 홉의 IP 주소 또는 디폴트 라우터의 IP 주소 값을 물리 주소로 변환해야 함
      - 호스트는 라우팅 테이블을 참조하여 목적지를 위한 다음 홉의 IP 주소를 찾음
      - 라우팅 테이블이 없으면 디폴트 라우터의 IP 주소를 찾음

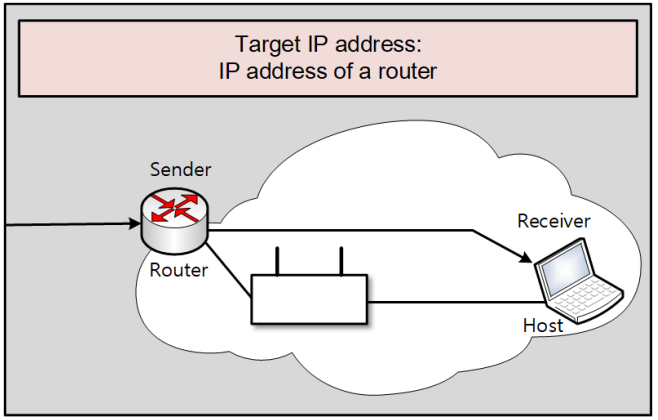


ARP

- 주소 변환 프로토콜(ARP) (3/11)
  - ARP가 사용되는 경우 (2/2)
    - 다른 네트워크로 전송 중인 패킷을 라우터가 중간에서 수신한 경우
      - 다음 홉 라우터의 IP 주소를 물리 주소로 변환해야 함
      - 라우터는 라우팅 테이블을 참조하여 다음 홉 라우터의 IP 주소를 찾음

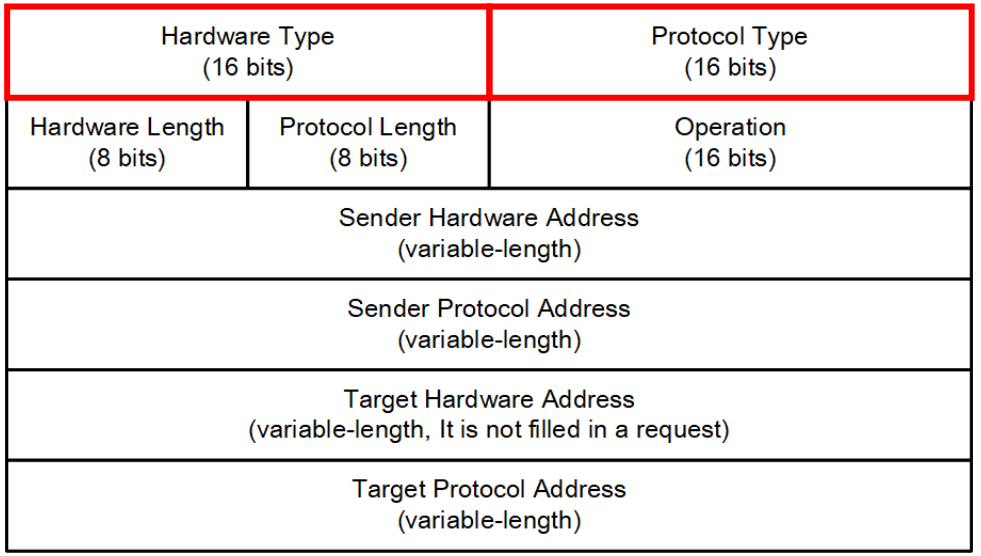


    - 라우터가 자신과 같은 네트워크에 있는 호스트로 가는 패킷을 수신한 경우
      - 데이터그램의 목적지 IP 주소가 물리 주소를 변환해야 함



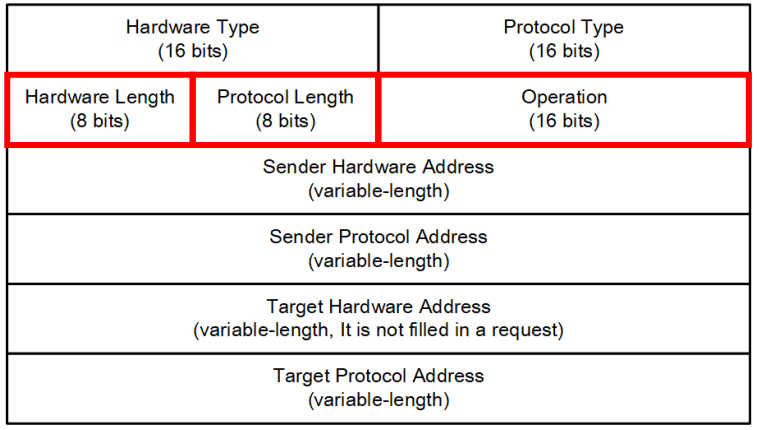
ARP

- 주소 변환 프로토콜(ARP) (4/11)
  - 패킷 구조 (1/4)
    - 하드웨어 유형(Hardware Type) 필드
      - ARP가 수행되고 있는 네트워크 유형을 정의하는 16비트 필드
        - e.g., 이더넷의 경우에 1을 값으로 가짐
    - 프로토콜 유형(Protocol Type) 필드
      - 프로토콜을 정의하는 16비트 필드
        - e.g., IPv4의 경우에 0800_{16}을 값으로 가짐

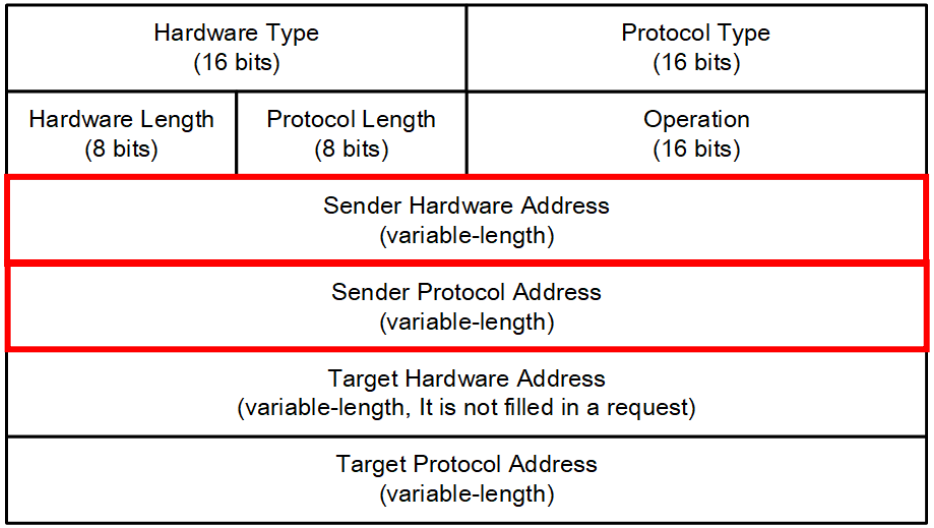


ARP

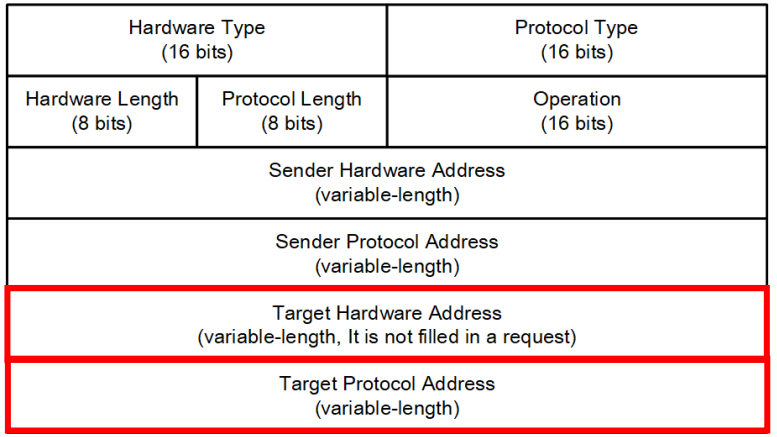
- 주소 변환 프로토콜(ARP) (5/11)
  - 패킷 구조 (2/4)
    - 하드웨어 길이(Hardware Length) 필드
      - 물리 주소의 길이를 바이트 단위로 정의하는 8비트 필드
        - e.g., 이더넷의 경우에는 6를 값으로 가짐
    - 프로토콜 길이(Protocol Length) 필드
      - 논리 주소의 길이를 바이트 단위로 정의하는 8비트 필드
        - e.g., IPv4의 경우에는 4를 값으로 가짐
    - 동작(Operation) 필드
      - 패킷의 유형을 정의하는 16비트 필드
        - ARP 요청(1)과 ARP 응답(2)의 두 가지 유형이 정의되어 있음



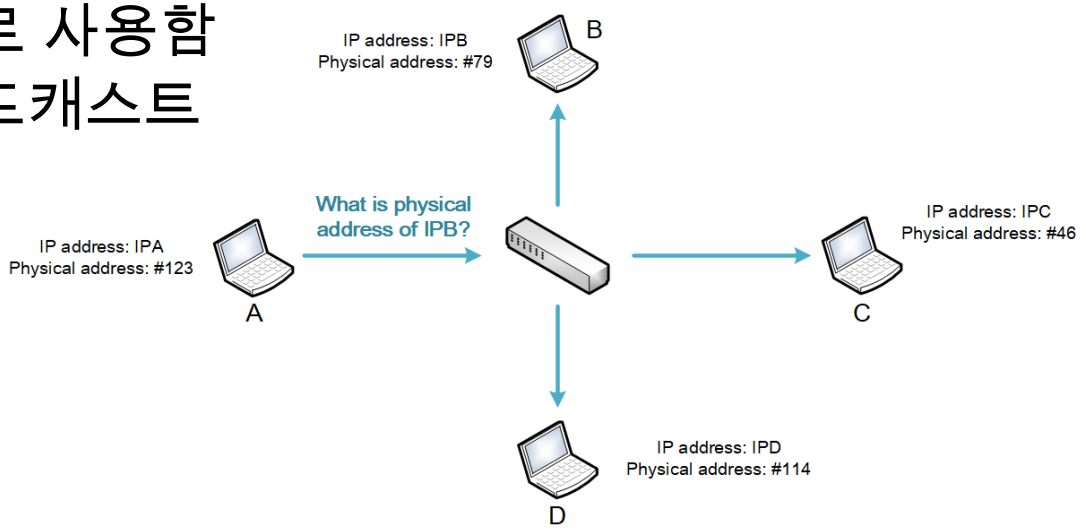
- 주소 변환 프로토콜(ARP) (6/11)
  - 패킷 구조 (3/4)
    - 발신지 하드웨어 주소(Sender Hardware Address) 필드
      - 송신자의 물리 주소를 나타내는 가변 길이의 필드
        - e.g., 이더넷의 경우 6바이트의 크기를 가짐
    - 발신지 프로토콜 주소(Sender Protocol Address) 필드
      - 송신자의 논리 주소를 나타내는 가변 길이의 필드
        - e.g., IP의 경우 4바이트의 크기를 가짐



- 주소 변환 프로토콜(ARP) (7/11)
  - 패킷 구조 (4/4)
    - 타깃 하드웨어 주소(Target Hardware Address) 필드
      - 수신자의 물리 주소를 나타내는 가변 길이의 필드
        - e.g., 이더넷의 경우 6바이트의 크기를 가짐
        - e.g., ARP 요청 메시지의 경우에 송신자는 타깃의 물리 주소를 알지 못하므로 0을 값으로 가짐
    - 타깃 프로토콜 주소(Target Protocol Address) 필드
      - 수신자의 논리 주소를 나타내는 가변 길이의 필드
        - e.g., IPv4의 경우 4바이트의 크기를 가짐

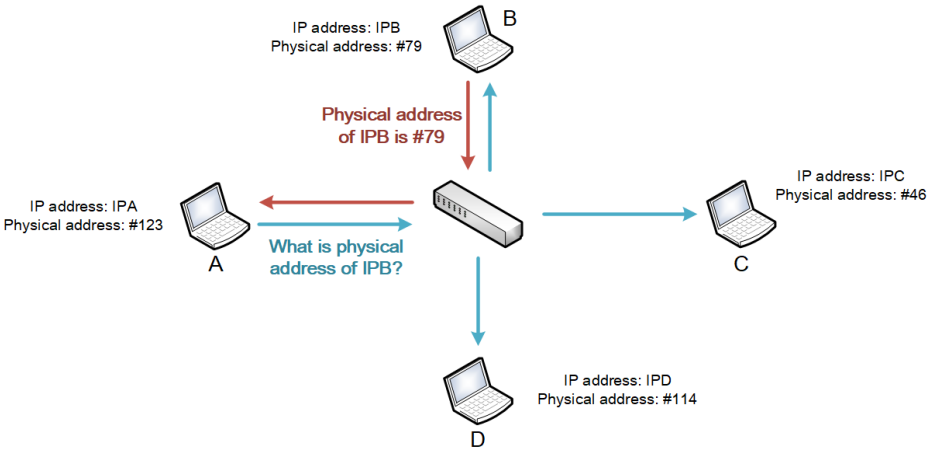


- 주소 변환 프로토콜(ARP) (8/11)
  - 동작 – ARP 요청
    1. 송신자는 수신자의 IP 주소를 알고 있음
    2. IP가 ARP에게 ARP 요청 메시지를 생성하도록 요청함
       - 송신자의 물리 주소와 IP 주소, 수신자의 IP 주소가 작성됨
       - 타깃의 물리 주소 필드는 0으로 채워짐
    3. 요청 메시지는 데이터 링크 계층에 전달되고, 논리 주소를 물리 주소로 매핑하는 캡슐화를 수행함
       - 발신지 주소를 물리 주소로 사용함
       - 목적지 주소를 물리 브로드캐스트 주소로 사용함



ARP

- 주소 변환 프로토콜(ARP) (9/11)
  - 동작 – ARP 응답
    1. 모든 호스트나 라우터는 구성된 프레임을 수신함
       - 모든 스테이션은 메시지를 자신의 ARP에게 전달함
       - 타깃을 제외하고 모든 기계는 해당 패킷을 폐기함
       - 타깃은 메시지의 IP 주소를 인식함
    2. 타깃 장비는 자신의 물리 주소를 포함하는 ARP 메시지를 유니캐스트 방식으로 응답함
    3. 송신자는 응답 메시지를 받고 타깃의 물리 주소를 알게 됨
    4. 타깃에게 보내질 데이터를 포함하는 IP 데이터그램을 3단계에서 구한 물리 주소를 가지는 프레임으로 캡슐화하여 목적지에 유니캐스트 함

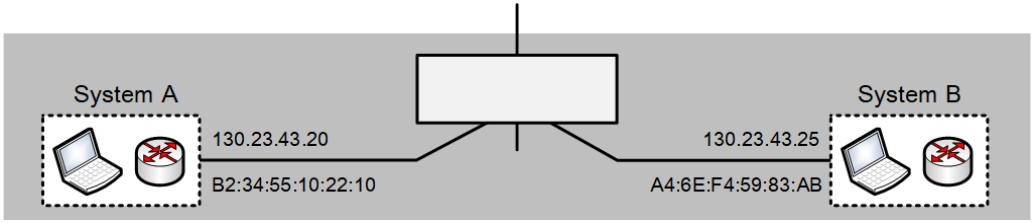


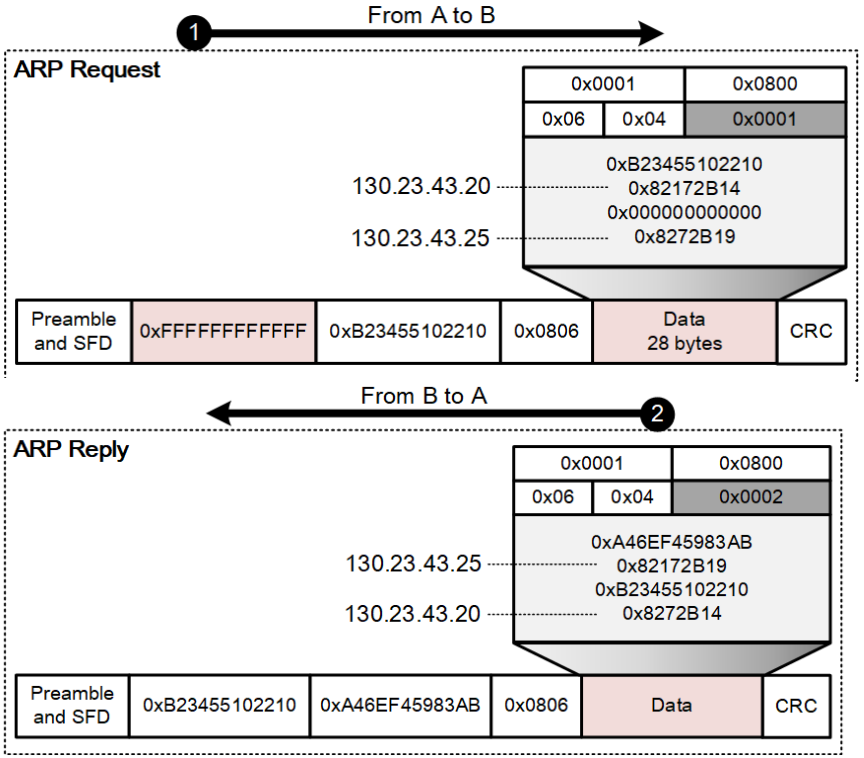
ARP

- 주소 변환 프로토콜(ARP) (10/11)
  - 예제 8.1

아래와 같은 주소들을 가진 호스트 A와 B가 있다.
같은 이더넷 네트워크 상에 있는 두 호스트 간에, 호스트 A가 호스트 B에게 패킷을 보낼 때 ARP 요청과 응답 패킷을 보여라

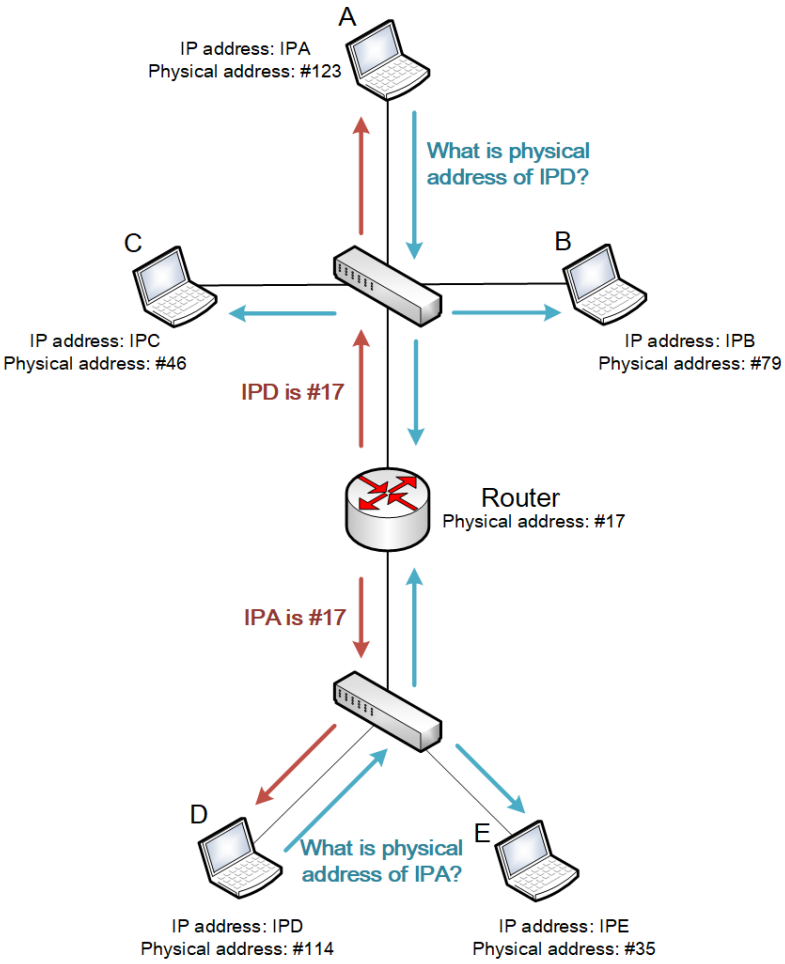
- 호스트 A
  - (IP 주소)130.23.43.20
  - (물리 주소)B2:34:55:10:22:10
- 호스트 B
  - (IP 주소)130.23.43.25
  - (물리 주소)A4:6E:F4:59:83:AB





ARP

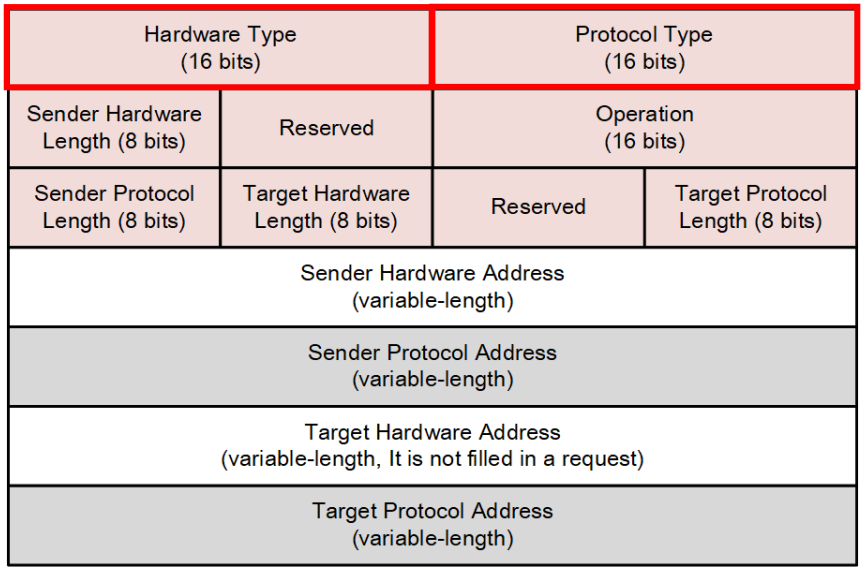
- 주소 변환 프로토콜(ARP) (11/11)
  - 프록시 ARP
    - 정의
      - 호스트 집합을 대행해 수행하는 ARP
    - 목적
      - 서브넷팅 효과를 만들기 위해 사용됨
    - 동작
      1. 프록시 ARP를 수행하는 라우터가 한 호스트의 물리 주소를 찾는 ARP 요청을 받음
      2. 라우터는 자신의 물리 주소를 ARP 응답 메시지로 알림
      3. 라우터가 실제 IP 패킷을 받으면 적절한 호스트나 라우터에게 전송함



ARP

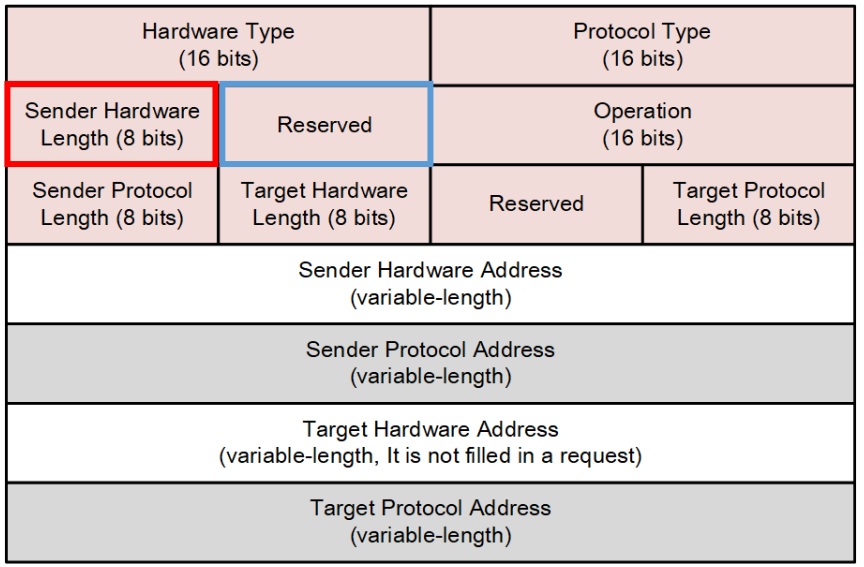
- ATMARP (1/7)
  - 정의
    - ATMWAN 내 진출점 라우터의 IP 주소가 주어졌을 때 진출점 라우터의 물리 주소를 찾기 위해 사용되는 프로토콜
  - LAN과 ATM 네트워크의 차이
    - LAN은 브로드캐스트 네트워크임
      - ARP는 ARP 요청을 전송하기 위해 LAN의 브로드캐스트를 사용함
    - ATM 네트워크는 브로드캐스트 네트워크가 아님
      - ARP 요청 전송을 처리하기 위한 다른 해결책이 필요함

- ATMARP (2/7)
  - 패킷 구조 (1/6)
    - 하드웨어 유형(Hardware Type) 필드
      - 물리적 네트워크를 정의하는 16비트 필드
      - ATM 네트워크에서 0013_{16}을 값으로 가짐
    - 프로토콜 유형(Protocol Type) 필드
      - 프로토콜의 유형을 정의하는 16비트 필드
      - IPv4의 경우 0800_{16}을 값으로 가짐



ARP

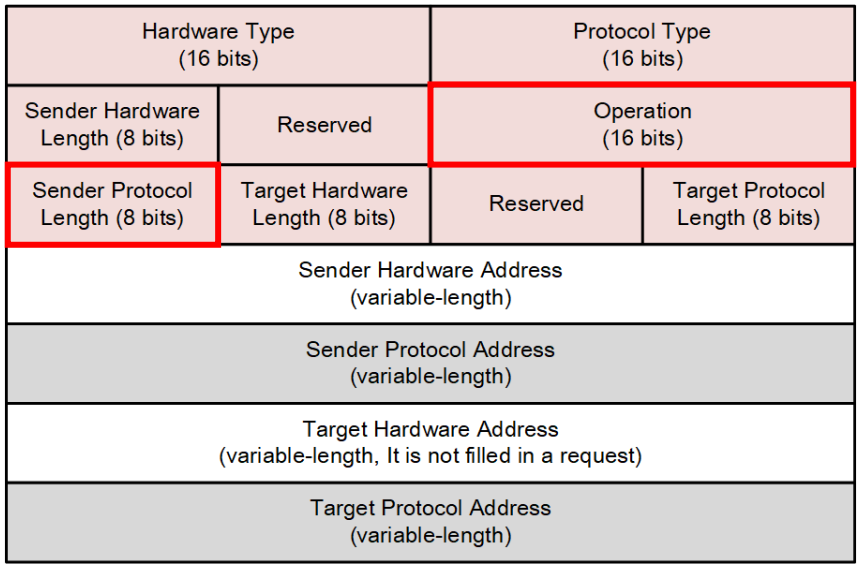
- ATMARP (3/7)
  - 패킷 구조 (2/6)
    - 송신자 하드웨어 길이(Sender Hardware Length) 필드
      - 송신자의 물리 주소의 길이를 바이트 단위로 정의하는 8비트 필드
      - ATM 네트워크의 경우 20을 값으로 가짐
      - 두 수준의 하드웨어 주소가 필요한 경우에는 근처의 reserved 필드가 두 번째 주소의 길이를 규정하는데 사용됨
        - ATM 네트워크에서는 하나의 노드가 기존 MAC 주소 외에 ATM 고유 주소 (End System Address)와 같은 ATM 자체 주소를 필요로 하는 경우가 존재함



ARP

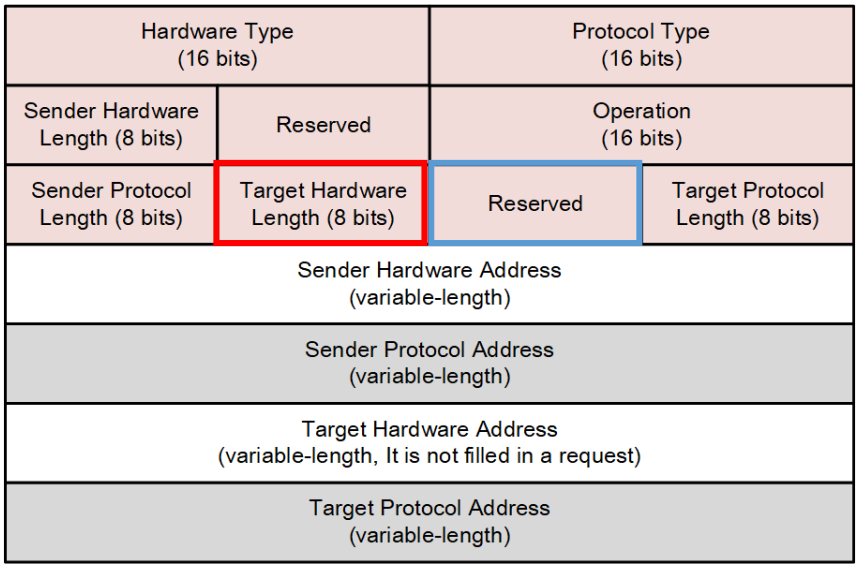
- ATMARP (4/7)
  - 패킷 구조 (3/6)
    - 동작(Operation) 필드
      - 패킷의 유형을 정의하는 16비트 필드
      - 다섯 개의 패킷 유형이 정의됨
    - 송신자 프로토콜 길이(Sender Protocol Length) 필드
      - 논리 주소의 길이를 바이트 단위로 정의한 8비트 필드
      - IPv4의 경우에 4를 값으로 가짐

Message	OPER value
Request	1
Reply	2
Inverse Request	8
Inverse Reply	9
NACK	10



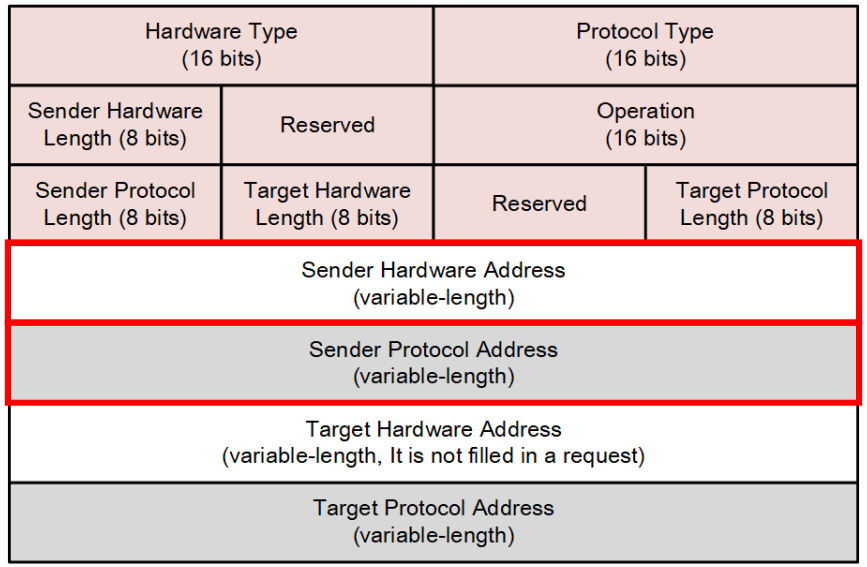
ARP

- ATMARP (5/7)
  - 패킷 구조 (4/6)
    - 타깃 하드웨어 길이(Target Hardware Length) 필드
      - 수신자의 물리 주소를 바이트 단위로 정의하는 8비트 필드
      - ATM 네트워크의 경우에 20을 값으로 가짐
      - 두 수준의 하드웨어 주소가 필요한 경우에는 근처의 reserved 필드가 두 번째 주소의 길이를 규정하는데 사용됨
        - ATM 네트워크에서는 하나의 노드가 기존 MAC 주소 외에 ATM 고유 주소 (End System Address)와 같은 ATM 자체 주소를 필요로 하는 경우가 존재함



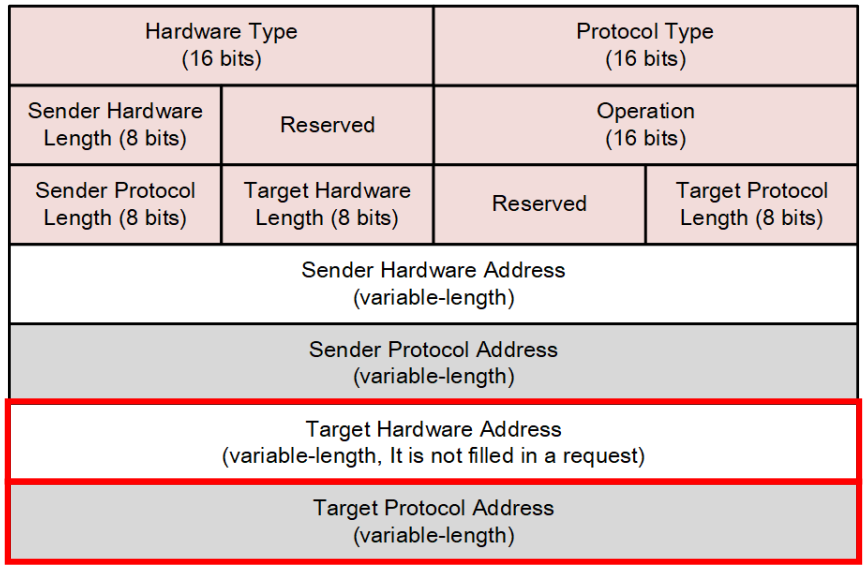
ARP

- ATMARP (6/7)
  - 패킷 구조 (5/6)
    - 송신자 하드웨어 주소(Sender Hardware Address) 필드
      - 송신자의 물리 주소를 정의하는 가변 길이 필드
      - ATM 네트워크의 경우에 20바이트의 길이를 가짐
    - 송신자 프로토콜 주소(Sender Protocol Address) 필드
      - 송신자의 논리 주소를 정의하는 가변 길이 필드
      - IPv4의 경우에 4바이트의 길이를 가짐



ARP

- ATMARP (7/7)
  - 패킷 구조 (6/6)
    - 타깃 하드웨어 주소(Target Hardware Address) 필드
      - 수신자의 물리 주소를 정의하는 가변 길이 필드
      - ATM 네트워크의 경우에 20바이트의 길이를 가짐
      - 요청 메시지에서는 비어있으며, 응답과 NACK 메시지에서 채워짐
    - 타깃 프로토콜 주소(Target Protocol Address) 필드
      - 수신자의 논리 주소를 정의하는 가변 길이 필드
      - IPv4의 경우에 4바이트의 길이를 가짐



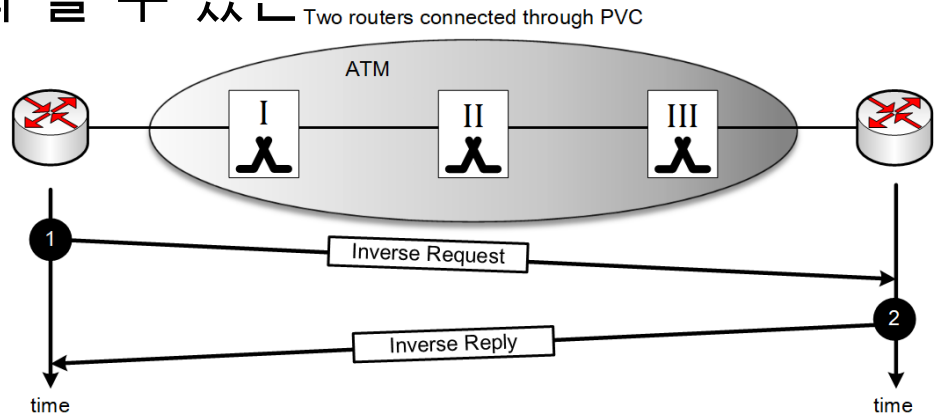
ARP

- ATMARP – 동작 (1/6)
  - 영구 가상 회선(PVC, Permanent Virtual Circuits) 연결 (1/2)
    - 정의
      - ATM 네트워크에서 두 종단점 간에 미리 설정된 경로를 이용하는 연결 방식
    - 특징
      - *VPI와 *VCI가 영구적인 연결을 위해 고정적으로 정의되어 입력됨
      - 라우터가 상대방의 물리 주소와 IP 주소를 알면 ATMARP 서버 없이 동작할 수 있음
      - 인버스 요청 메시지와 인버스 응답 메시지를 통해 물리 주소를 IP 주소에 결합할 수 있음

*VPI (Virtual Path Identifier)
ATM 네트워크에서 가상 경로를 식별하기 위해 셀 헤더에 포함되는 식별자

*VCI (Virtual Channel Identifier)
ATM 네트워크에서 가상 채널을 식별하기 위해 셀 헤더에 포함되는 식별자

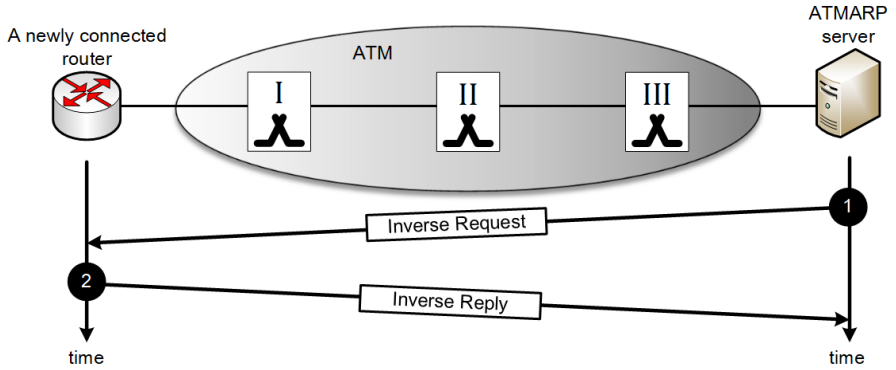
- ATMARP – 동작 (2/6)
  - 영구 가상 회선(PVC, Permanent Virtual Circuits) 연결 (2/2)
    - 동작 절차
      1. 라우터가 인버스 요청 메시지를 송신함
         - 인버스 요청 메시지는 송신자의 물리 주소와 IP 주소를 내장함
      2. 반대편의 라우터는 메시지를 수신하고 인버스 응답을 전송함
         - 인버스 응답은 응답 메시지 송신자의 물리 주소와 IP 주소를 내장함
      3. 두 라우터들은 물리 주소를 PVC에 매핑하는 테이블 항목에 추가함
      4. 라우터가 IP 데이터그램을 수신하면, 라우터가 PVC를 사용하여 데이터그램을 캡슐화 할 수 있는 정보를 제공함



ARP

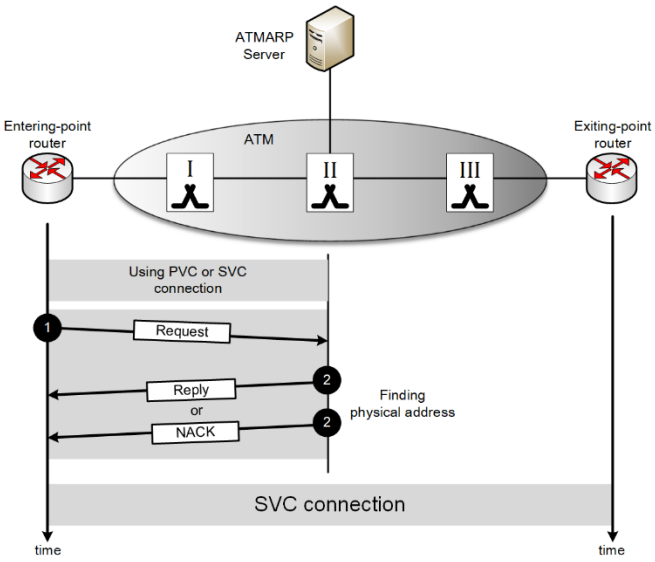
- ATMARP – 동작 (3/6)
  - 교환 가상 회선(SVC, Switched Virtual Connection) 연결 (1/4)
    - 정의
      - ATM 네트워크에서 통신이 필요할 때마다 동적으로 설정되고 종료되는 일시적인 연결 방식
    - 특징
      - 라우터 간에 통신을 할 때마다 필요에 따라 새로운 가상 회선이 설립되어야 함
      - 라우터가 상대방의 주소 정보를 모를 경우, ATMARP 서버를 통해 물리 주소를 알아내야함
      - ATMARP 요청 및 응답 메시지를 통해 물리 주소를 IP 주소에 매핑할 수 있음

- ATMARP – 동작 (4/6)
  - 교환 가상 회선(SVC, Switched Virtual Connection) 연결 (2/4)
    - ATM 서버 매핑 테이블
      - 생성 절차
        1. 라우터가 ATM 네트워크에 처음 연결된 후, PVC가 라우터와 서버 사이에 설립됨
        2. 서버는 라우터에게 인버스 요청 메시지를 보냄
        3. 라우터는 자신의 IP 주소와 물리 주소를 포함한 인버스 응답 메시지를 보냄
        4. 응답받은 라우터의 주소를 가지고, 서버는 라우팅 테이블에 해당 라우터가 추후에 진출점 라우터가 되는 경우에 사용할 항목을 추가함

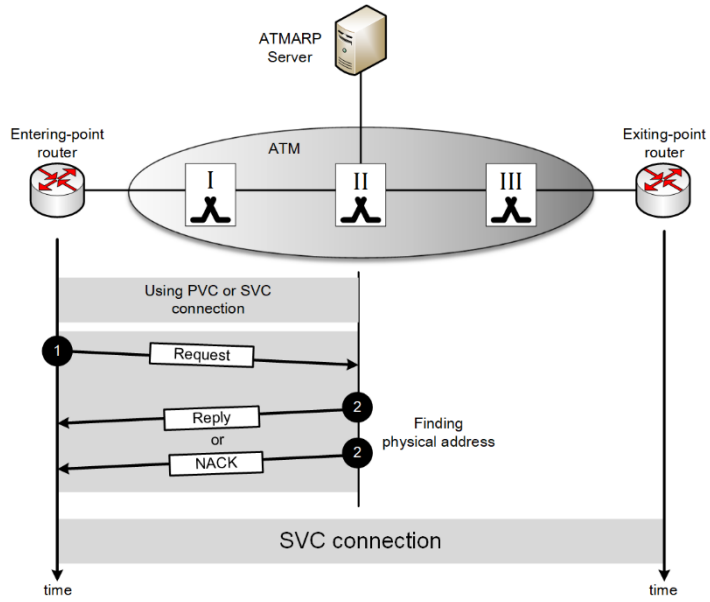


ARP

- ATMARP – 동작 (5/6)
  - 교환 가상 회선(SVC, Switched Virtual Connection) 연결 (3/4)
    - 동작 절차 (1/2)
      1. 라우터와 서버를 연결함
         - 라우터와 서버 사이에 연결된 PVC를 통해 서버에 연결함
         - 라우터와 서버 사이에 PVC가 없다면, 서버는 ATMARP 요청 및 응답 메시지 교환을 위한 SVC 연결을 생성하기 위해 최소한 라우터의 물리 주소를 알아야 함
      2. 물리 주소를 수신함
         - 진입점 라우터와 서버를 연결한 뒤, 라우터는 ATMARP 요청을 서버로 보냄
         - 서버가 물리 주소의 유무에 따라 응답을 보냄
           - 물리 주소를 찾을 수 있으면 ATMARP 응답을 보냄
           - 물리 주소를 찾을 수 없으면 ATMARP NACK을 보냄
         - 진입점 라우터가 NACK을 받으면 데이터그램을 폐기함

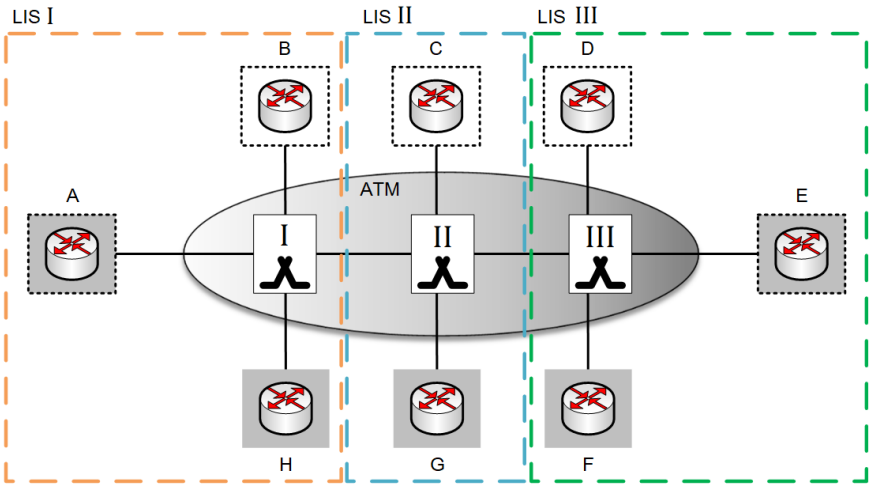


- ATMARP – 동작 (6/6)
  - 교환 가상 회선(SVC, Switched Virtual Connection) 연결 (4/4)
    - 동작 절차 (2/2)
      3. 가상 회선을 설립함
         - 진입점 라우터가 진출점 라우터의 물리 주소를 받으면, 자신과 진출점 라우터 간의 SVC를 요청함
         - ATM 네트워크는 SVC를 설립하기 위해 두 개의 물리 주소를 사용함
         - 네트워크 내의 교환기는 IP 데이터그램을 운반하는 셀들이 라우팅 되도록 테이블에 항목을 추가함
      4. 연결을 해제함ATM 네트워크는 진입점 요구하면 SVC 연결을 해제함



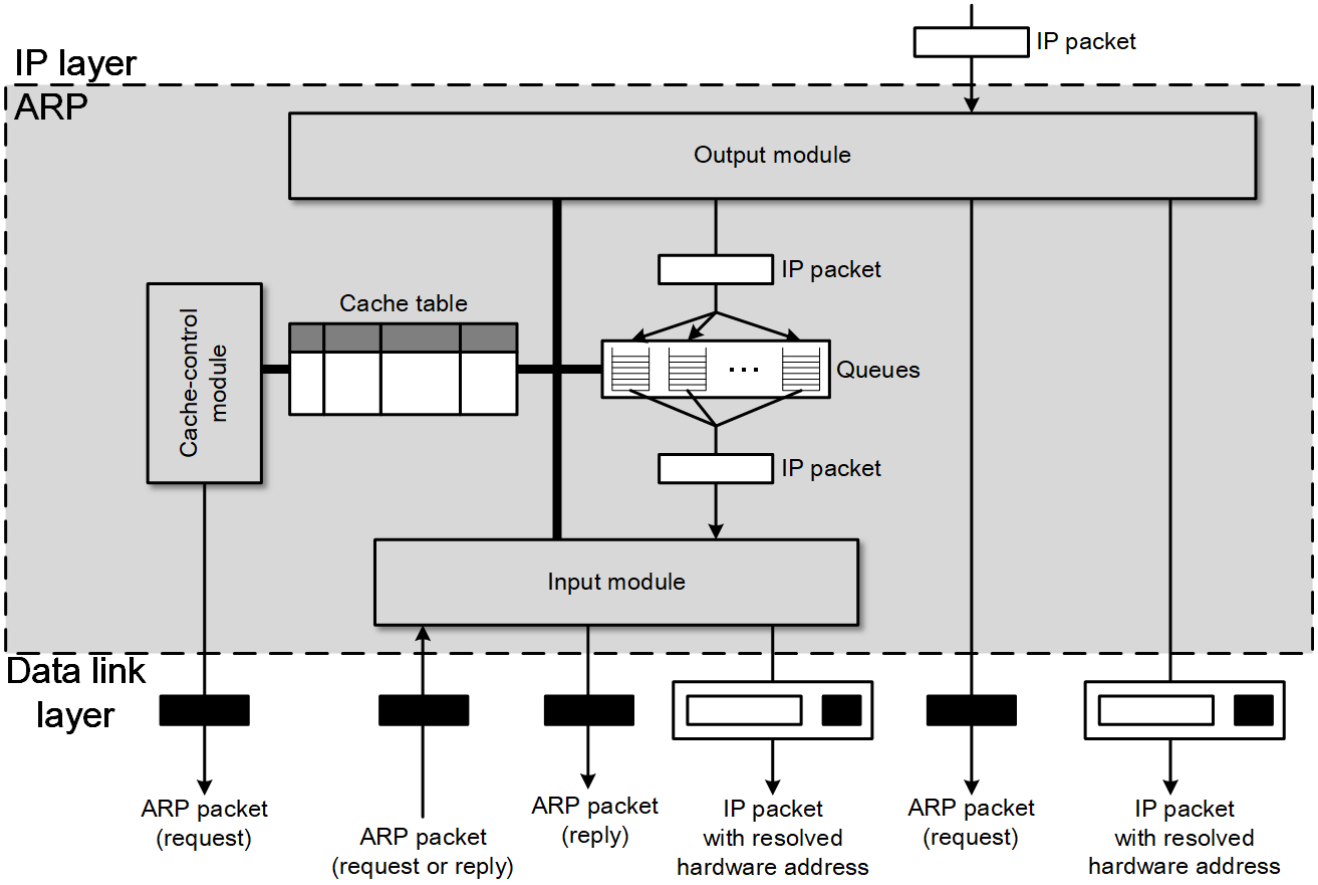
ARP

- ATMARP
  - 논리적 IP 서브넷(LIS, Logical IP Subnet)
    - 정의
      - IP 주소 공간을 논리적으로 나눈 작은 네트워크 단위
    - 특징
      - 브로드캐스트 기능이 없는 ATM에서 브로드캐스트 기능을 흉내 내는 방식으로 ATMARP와 같은 프로토콜이 사용될 수 있도록 함
      - ATM 네트워크에 연결된 라우터들은 하나 이상의 논리적 서브넷에 속할 수 있음
      - 여러 개의 서브넷에 속한 라우터를 통해 다른 서브넷에 속한 라우터와 통신할 수 있음
      - ATMARP를 사용하기 위해 각 서브넷에 각각의 ATMARP 서버가 있어야 함



ARP

- ARP 패키지
  - 정의
    - 가상적 ARP 패키지의 구성요소와 구성요소 간의 상호작용 전체를 표현한 것
  - 구성요소
    - 캐쉬 테이블
    - 큐
    - 출력 모듈
    - 입력 모듈
    - 캐쉬 제어 모듈



ARP

- ARP 패키지 – 구성요소 (1/9)
  - 캐쉬 테이블 (1/2)
    - 최근에 조회하거나 응답받은 IP 주소와 물리 주소의 매핑 정보를 임시로 저장해 두는 테이블
    - 같은 IP 주소에 대해 ARP 요청을 반복하지 않도록 함
    - 엔트리의 배열로 구현되어 있으며, 각 엔트리는 여러 필드를 포함하고 있음
      - e.g., 상태(state), 하드웨어 유형, 프로토콜 유형, 하드웨어 길이, 프로토콜 길이, 하드웨어 주소, 프로토콜 주소, 인터페이스 번호, 큐 번호, 시도, 타임아웃

ARP

- ARP 패키지 – 구성요소 (2/9)
  - 캐쉬 테이블 (2/2)

Field	Description
State	엔트리의 상태를 보여줌 • FREE : 해당 엔트리의 수명(TTL)이 만료 되었으며, 다른 엔트리에 사용될 수 있음 • PENDING : 해당 엔트리에 대한 요청을 보냈으나 응답을 아직 수신하지 못함 • RESOLVED : 엔트리가 목적지의 물리 주소를 가지고 있음
Hardware Type	ARP가 수행되고 있는 네트워크의 유형을 정의함 (ARP 패킷에서의 하드웨어 유형과 같음)
Protocol Type	프로토콜을 정의함 (ARP 패킷에서의 프로토콜 유형과 같음)
Hardware Length	물리 주소의 길이를 바이트 단위로 정의함 (ARP 패킷에서의 하드웨어 길이와 같음)
Protocol Length	논리 주소의 길이를 바이트 단위로 정의함 (ARP 패킷에서의 프로토콜 길이와 같음)
Interface Number	다수의 인터페이스를 가진 호스트에서, 각 네트워크 인터페이스를 식별할 번호를 보여줌
Queue Number	같은 목적지 IP 주소로 전송될 패킷들을 모아놓은 큐를 식별하기 위한 번호를 보여줌
Attempts	해당 엔트리에 대해 ARP 요청을 전송한 횟수를 보여줌
Time-Out	해당 엔트리의 TTL을 초단위로 환산한 값을 보여줌
Hardware Address	목적지 하드웨어의 주소를 보여줌 ARP 응답을 통해 주소변환이 이루어지기 전까지 비어있음
Protocol Address	목적지 IP 주소를 보여줌

ARP

- ARP 패키지 – 구성요소 (3/9)
  - 큐
    - ARP가 하드웨어 주소를 얻기 전 IP 데이터그램이 저장되는 자료구조
    - 한 목적지당 하나의 큐가 유지됨
    - 출력 모듈은 해결되지 않은 패킷을 해당 큐에 보냄
    - 입력 모듈은 큐에서 패킷을 해결된 물리 주소와 같이 가져와 데이터 링크 계층으로 전달함

- ARP 패키지 – 구성요소 (4/9)
  - 출력 모듈 (1/2)
    - 동작
      - IP 소프트웨어로부터 IP 패킷을 기다림
      - 캐쉬 테이블에서 IP 패킷의 목적지와 일치하는 항목을 찾음
        - 일치하는 항목을 찾지 못한 경우
          - 상태를 PENDING으로 설정한 캐쉬 항목을 생성함
          - 시도 횟수를 1로 설정함
          - 큐를 생성하고 패킷을 삽입함
          - ARP 요청을 보냄
        - 일치하는 항목을 찾은 경우
          - 상태가 RESOLVED라면 하드웨어 주소 값을 추출하고 패킷과 함께 데이터 링크 계층으로 전송함
          - 상태가 PENDING이라면 패킷을 일치하는 큐에 삽입하고 종료함

- ARP 패키지 – 구성요소 (5/9)
  - 출력 모듈 (2/2)
    - 의사코드

```
ARP_Output_Module ( )
{
    Sleep until an IP packet is received from IP software.
    Check cache table for an entry corresponding to the destination of IP packet.
    if (entry is found) {
        if (the state is RESOLVED) {
            Extract the value of the hardware address from the entry.
            Send the packet and the hardware address to data link layer.
            Return
        }   // end if
        if (the state is PENDING) {
            Enqueue the packet to the corresponding queue.
            Return
        }   //end if
    }       //end if
    if (entry is not found) {
        Create a cache entry with state set to PENDING and ATTEMPTS set to 1.
        Create a queue.
        Enqueue the packet.
        Send an ARP request.
        Return
    }       //end if
}           //end module
```

- ARP 패키지 – 구성요소 (6/9)
  - 입력 모듈 (1/2)
    - 동작
      - ARP 패킷을 기다림
      - 캐쉬 테이블에서 IP 패킷의 목적지와 일치하는 항목을 찾음
        - 일치하는 항목을 찾지 못한 경우
          - 새 항목을 생성하고 테이블에 항목을 추가하고 종료함
        - 일치하는 항목을 찾은 경우
          - 항목을 업데이트 하고 계속 진행함
      - 패킷이 요청인지 확인함
        - ARP 응답를 전송하고 종료함
      - 상태가 PENDING이라면 큐가 비어있지 않은 동안 패킷을 하나씩 꺼내서 패킷과 하드웨어 주소를 전송함

- ARP 패키지 – 구성요소 (7/9)
  - 입력 모듈 (2/2)
    - 의사코드

```
ARP_Input_Module ( )
{
    Sleep until an ARP packet (request or reply) arrives.
    Check the cache table to find the corresponding entry.
    if (found) {
        Update the entry.
        if (the state is PENDING) {
            While (the queue is not empty) {
                Dequeue one packet.
                Send the packet and the hardware address.
            } //end if
        }   //end if
    } //end if
    if (not found) {
        Create an entry.
        Add the entry to the table.
    }
    if (the packet is a request)
    {
        Send an ARP reply.
    }       //end if
    Return
} //end module
```

- ARP 패키지 – 구성요소 (8/9)
  - 캐쉬 제어 모듈 (1/2)
    - 동작
      - 주기적으로 타이머가 만료될 때 까지 기다림
      - 캐쉬 테이블의 모든 항목에 대해 반복함
        - 상태가 FREE인 경우
          - 다음 항목을 진행함
        - 상태가 PENDING인 경우
          - 시도 횟수를 1만큼 증가시키고 시도 횟수가 최대값을 초과하는지 확인함
            - 초과하지 않았다면 ARP 요청을 전송함
            - 초과했다면 상태를 FREE로 변경하고 해당 큐를 삭제함
          - 다음 항목을 진행함
        - 상태가 RESOLVED인 경우
          - Time-Out 값을 감소시키고 해당 값이 0이하인지 확인함
            - 0이하라면 상태를 FREE로 변경하고 해당 큐를 삭제함

- ARP 패키지 – 구성요소 (9/9)
  - 캐쉬 제어 모듈 (2/2)
    - 의사코드

```
ARP_Cache_Control_Module ( )
{
    Sleep until the periodic timer matures.
    Repeat for every entry in the cache table
    {
        if (the state is FREE) {
            Continue.
        } // end if
        if (the state is PENDING) {
            Increment the value of attempts by 1.
            if (attempts greater than maximum) {
                Change the state to FREE.
                Destroy the corresponding queue.
            } //end if
            else {
                Send and ARP request
            } //end else
            continue.
        } //end if
        if (the state is RESOLVED) {
            Decrement the value of time-out.
            if (time-out less than or equal 0) {
                Change the state to FREE.
                Destroy the corresponding queue.
            } // end if
        } //end if
    } //end repeat
    Return.
} // end module
```

ARP

- ARP 패키지 (1/4)
  - 예제 8.2

ARP 출력 모듈은 IP 계층으로부터 목적지 주소가 114.5.7.89인 IP 데이터그램을 수신한다. 해당 표는 캐쉬 테이블의 초기 값이다. 캐쉬 테이블의 변화를 관찰하라.

- 캐쉬 테이블을 검사한 결과, 해당 주소를 위한 엔트리가 RESOLVED 상태를 갖고 존재함
- 하드웨어 주소 457342ACAE32를 꺼내어 패킷과 이 주소를 데이터 링크 계층에 전달함
- 캐쉬 테이블에는 변화 없음

State	Queue	Attempt	Time-Out	Protocol Addr.	Hardware Addr.
R	5	-	900	180.3.6.1	ACAE32457342
P	2	2	-	129.34.4.8	-
P	14	5	-	201.11.56.7	-
R	8	-	450	114.5.7.89	457342ACAE32
P	12	1	-	220.55.5.7	-
F	-	-	-	-	-
R	9	-	60	19.1.7.82	4573E3242ACA
P	18	3	-	188.11.8.71	-

ARP

- ARP 패키지 (2/4)
  - 예제 8.3

20초 후 ARP 출력 모듈이 IP 계층으로부터 목적지 주소가 116.1.7.22인 IP 데이터그램을 수신한다. 해당 상황에 대하여 동작을 분석하라.

- 캐쉬 테이블을 검사한 결과, 목적지 주소가 동일한 항목이 존재하지 않음
  - 모듈은 새 항목을 테이블에 추가하고, 해당 엔트리의 상태를 PENDING으로 설정한 후 시도 값을 1로 설정함
  - 모듈은 큐를 새로 만들어 패킷을 해당 큐에 넣고 ARP 요청을 데이터 링크 계층을 통하여 목적지로 전송함

State	Queue	Attempt	Time-Out	Protocol Addr.	Hardware Addr.
R	5	-	900	180.3.6.1	ACAE32457342
P	2	2	-	129.34.4.8	-
P	14	5	-	201.11.56.7	-
R	8	-	450	114.5.7.89	457342ACAE32
P	12	1	-	220.55.5.7	-
P	23	1	-	116.1.7.22	-
R	9	-	60	19.1.7.82	4573E3242ACA
P	18	3	-	188.11.8.71	-

ARP

- ARP 패키지 (3/4)
  - 예제 8.4

15초 후 ARP 입력 모듈은 타겟 프로토콜(IP) 주소 188.11.8.71을 가진 ARP 패킷을 수신한다. 해당 상황에 대하여 동작을 분석하라.

- 캐쉬 테이블을 검사한 결과, 해당 주소를 가지고 있음을 확인할 수 있음
  - 모듈은 타깃 하드웨어 주소 E34573242ACA를 삽입함
  - 모듈은 큐 18에 접근하여 해당 큐 속에서 대기 중인 모든 패킷을 데이터 링크 계층으로 하나씩 전송함

State	Queue	Attempt	Time-Out	Protocol Addr.	Hardware Addr.
R	5	-	900	180.3.6.1	ACAE32457342
P	2	2	-	129.34.4.8	-
P	14	5	-	201.11.56.7	-
R	8	-	450	114.5.7.89	457342ACAE32
P	12	1	-	220.55.5.7	-
P	23	1	-	116.1.7.22	-
R	9	-	60	19.1.7.82	4573E3242ACA
P	18	3	-	188.11.8.71	E34573242ACA

ARP

- ARP 패키지 (4/4)
  - 예제 8.5

25초 뒤 캐쉬 제어 모듈이 모든 항목을 갱신한다. 해당 상황에 대해 테이블의 변화를 분석하라.

- 네 개의 해결된 항목의 타임아웃 값은 60만큼 감소함
- 맨 마지막 바로 전 항목은 타임아웃 값이 0이 되므로 상태가 FREE로 변경됨
- 모듈은 시도 필드의 값을 증가시킴
  - IP 주소 값이 201.11.56.7인 항목은 시도 필드의 값이 최대값보다 크게 되므로 상태를 FREE로 바꾸고 해당하는 큐를 제거함

State	Queue	Attempt	Time-Out	Protocol Addr.	Hardware Addr.
R	5	-	840	180.3.6.1	ACAE32457342
P	2	3	-	129.34.4.8	-
F	-	-	-	-	-
R	8	-	390	114.5.7.89	457342ACAE32
P	12	2	-	220.55.5.7	-
P	23	2	-	116.1.7.22	-
F	-	-	-	-	-
P	18	-	-	188.11.8.71	E34573242ACA

Thanks!

송 민 희(minhee@pel.sejong.ac.kr)

#부록 1

- 실습 - 옵션 유형(Option Type) (1/5)
  - 예제 7.12

ping 프로그램을 사용하여 호스트에 연결이 가능한지 점검해보자.

```
$ ping fhda.edu
PING fhda.edu (153.18.8.1) 56(84) bytes of data.
64 bytes from tiptoe.fhda.edu (153.18.8.1) : icmp_seq =
0 ttl=62 time=1.87 ms
…
```

- 분석
  - De Anza College의 fhda.edu 서버를 ping 함
  - 호스트의 주소는 153.18.8.1임
  - 64bytes의 응답을 받음
  - 왕복시간(time)은 1.87ms로 네트워크 지연이 매우 짧음
  - 현재 네트워크에서 정상적으로 접근 가능함.

- 실습 - 옵션 유형(Option Type) (2/5)
  - 예제 7.13

ping –R 옵션을 사용하여 경로 기록 옵션이 어떻게 작동하는지 확인해보자.

```
$ ping -R fhda.edu
PING fhda.edu (153.18.8.1) 56(124) bytes of data.
64 bytes from tiptoe.fhda.edu (153.18.8.1): icmp_seq=0 ttl=62 time=2.70 ms
RR:  voyager.deanza.fhda.edu (153.18.17.11)
       Dcore_G0_3-69.fhda.edu (153.18.251.31)
       Dbackup.V13.fhda.edu (153.18.191.249)
       tiptoe.fhda.edu (153.18.8.1)
       Dbackup_V62.fhda.edu (153.18.251.34)
       Dcore_G0_1-6.fhda.edu (153.18.31.254)
       voyager.deanza.fhda.edu (153.18.17.11)
```

- 분석
  - RR 아래 나열된 주소는 패킷이 거쳐간 라우터의 IP 주소로, 총 7개의 라우터를 거쳐 tiptoe.fhda.edu에 도달함

- 실습 - 옵션 유형(Option Type) (3/5)
  - 예제 7.14

traceroute 유틸리티를 사용해서 데이터그램의 경로를 추적해보자.

```
$ traceroute fhda.edu
traceroute to fhda.edu (153.18.8.1), 30 hops max, 38 byte packets
 1 Dcore_G0_1-6.fhda.edu (153.18.31.254) 0.972 ms 0.902 ms 0.881 ms
 2 Dbackup_V69.fhda.edu (153.18.251.4) 2.113 ms 1.996 ms 2.059 ms
 3 tiptoe.fhda.edu (153.18.8.1) 1.791 ms 1.741 ms 1.751 ms
```

- 분석
  - 3개의 홉을 거쳐 목적지에 도달함
  - 각 라우터에 대한 왕복시간이 기록되어있음
    - 신뢰도를 높이기 위해 패킷을 3회 전송함

#부록 1

- 실습 - 옵션 유형(Option Type) (4/5)
  - 예제 7.15

traceroute –g 명령어를 사용해서 느슨한 발신지 경로 옵션을 위한 중간 경로를 설정해보자

```
$ traceroute -g 153.18.251.4 fhda.edu
traceroute to fhda.edu (153.18.8.1), 30 hops max, 46 byte packets
 1 Dcore_G0_1-6.fhda.edu (153.18.31.254) 0.976 ms 0.906 ms 0.889 ms
 2 Dbackup_V69.fhda.edu (153.18.251.4) 2.168 ms 2.148 ms 2.037 ms
```

- 분석
  - -g 옵션으로 153.18.251.4 의 라우터를 반드시 거쳐가도록 설정함
  - 두 번째 홉(Dbackup)이 설정한 라우터 값과 동일함
  - 네트워크가 해당 옵션을 반영하여 동작중임

#부록 1

- 실습 - 옵션 유형(Option Type) (5/5)
  - 예제 7.16

traceroute –G 명령어를 사용해서 엄격한 발신지 경로 옵션을 위한 경로를 설정해보자

```
$ traceroute -G 153.18.251.4 fhda.edu
traceroute to fhda.edu (153.18.8.1), 30 hops max, 46 byte packets
 1 Dbackup_V69.fhda.edu (153.18.251.4) 2.168 ms 2.148 ms 2.037 ms
```

- 분석
  - -G 옵션으로 153.18.251.4 의 라우터만 지정함
  - 첫 번째 홉(Dbackup)이 설정한 라우터 값과 동일함
    - 다른 중간 경로 없이, 지정한 라우터로 바로 전달됨
  - 지정한 라우터를 거쳐 원래의 목적지로 전달됨 (출력 생략됨)