2024/08/20, 2024 네트워크 세미나

TCP/IP 프로토콜

-7장 IPv4, 8장 ARP-

이 정 민(jeongmin@pel.sejong.ac.kr)

세종대학교 프로토콜공학연구실

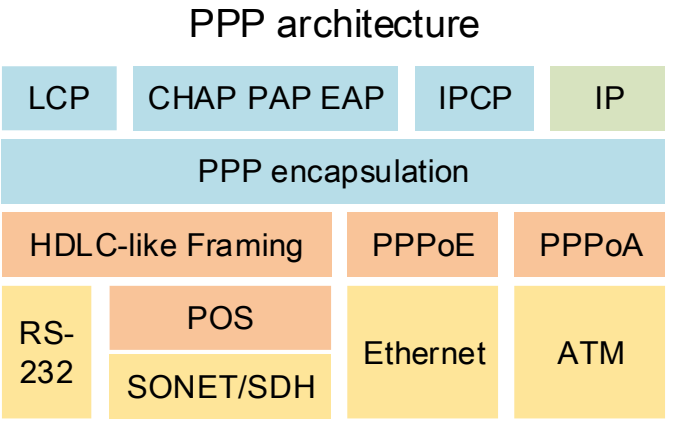
목 차

- 보충
- IPv4
- ARP

목 차

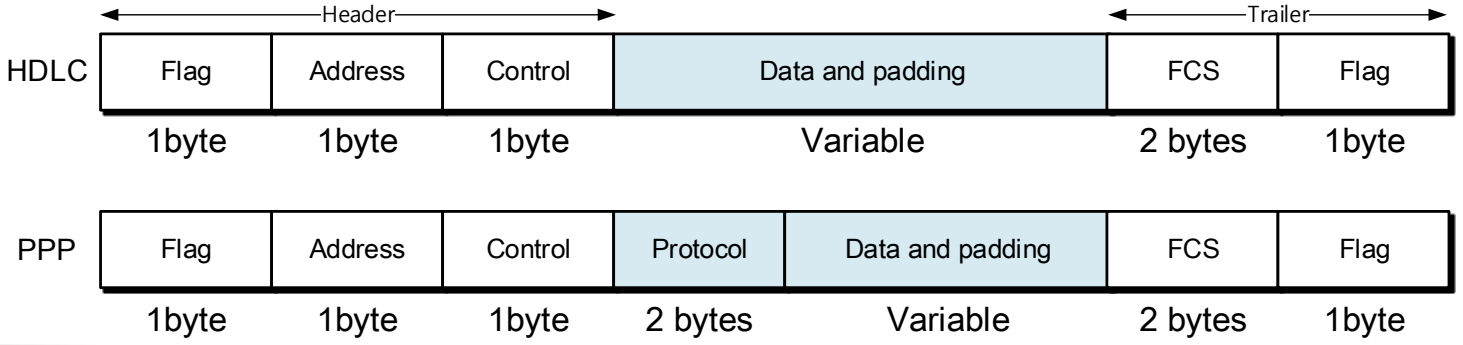
보충 (1/8)

- PPP (Point-to-Point Protocol) (1/8)
  - 정의
    - 두 네트워크 장치 간 데이터 전송을 제어하고 관리하는 프로토콜
  - 특징
    - 점대점 데이터링크 프로토콜임
    - 단일 링크 상에서 복수의 네트워크 계층용 프로토콜을 다중화시켜 사용할 수 있음
    - HDLC (High-level Data Link Control)에서 유래됨



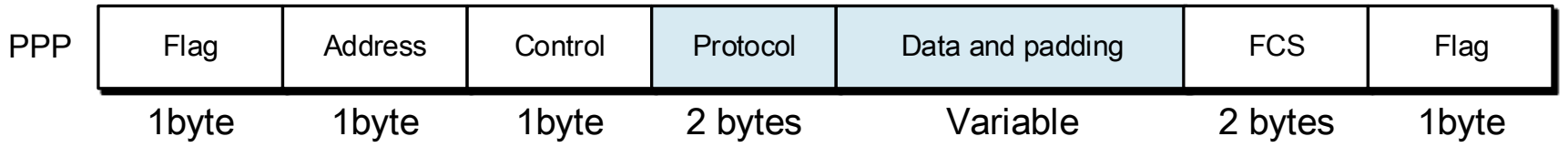
보충 (2/8)

- PPP (2/8)
  - 프레임 구조 (1/2)
    - 플래그 (Flag)
      - PPP 프레임의 경계를 나타내며 , 01111110 값을 가짐
    - 주소 (Address)
      - 항상 11111111의 값을 가짐
        - 두 점 간의 링크이므로 주소가 필요 없음
    - 제어
      - 항상 00000011의 값을 가짐
        - HDLC에서는 다양한 용도로 사용하나, PPP에서는 번호 없는 정보 (Unnumbered Information)를 가리키는 값으로 세팅함



보충 (3/8)

- PPP (3/8)
  - 프레임 구조 (2/2)
    - 프로토콜
      - 첫 비트가 1로 시작하는 경우
        - 다른 프로토콜과 교섭하기 위한 프로토콜임
      - 첫 비트가 0으로 시작하는 경우
        - 캡슐화되는 네트워크 프로토콜이 어떤 것인지를 지칭함
    - 데이터
      - 상위 계층 데이터 정보를 담고 있음
    - FCS
      - 오류 검출을 위해 사용되는 값임
        - (주소 +제어+프로토콜+데이터+패딩)을 이용해서 계산됨

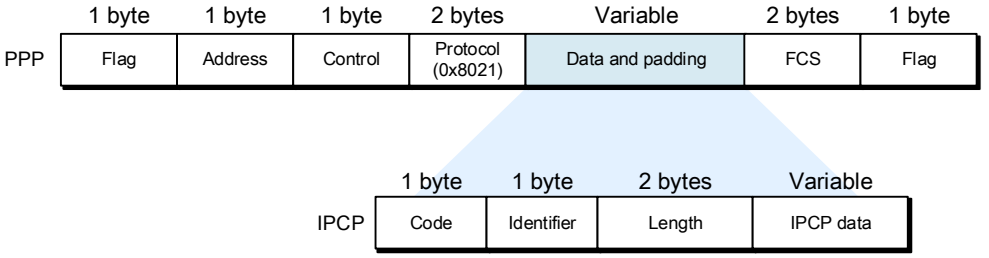


보충 (4/8)

- PPP (4/8)
  - 주요 구성 요소
    - 캡슐화 (Encapsulation)
      - 상위 계층 메시지를 받아서 하위 물리 계층 링크로 전송하는데 사용되는 방법
    - 링크 제어 프로토콜 (LCP, Link Control Protocol)
      - PPP 링크의 회선 관리를 담당하는 프로토콜
    - 네트워크 제어 프로토콜 (NCP, Network Control Protocol)
      - PPP 링크에서 다양한 네트워크 계층 프로토콜의 설정과 관리를 담당하는 프로토콜

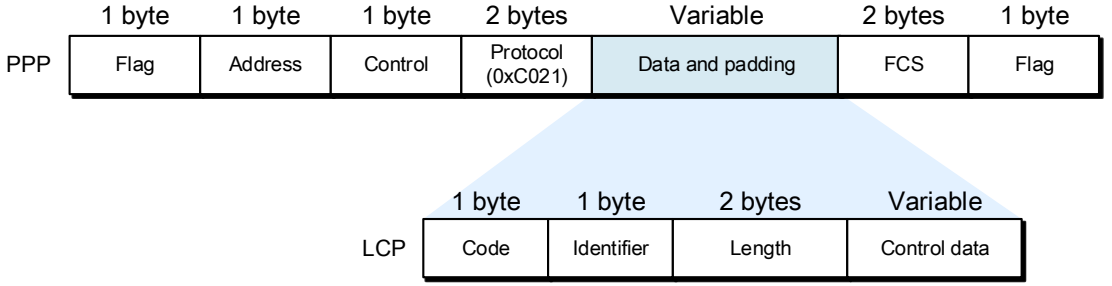
보충 (5/8)

- PPP (5/8)
  - NCP
    - 정의
      - PPP 링크에서 다양한 네트워크 계층 프로토콜의 설정과 관리를 담당하는 프로토콜
    - 주요 기능
      - 네트워크 계층에서 오는 데이터를 PPP 프레임에 캡슐화 함
      - 단일 PPP 링크에 복수의 망 계층 프로토콜 사용이 가능하게 함
    - 종류
      - IP 제어 프로토콜 (IPCP)
      - OSI 네트워크 제어 프로토콜 (OSINLCP)
      - AppleTalk 제어 프로토콜 (ATCP)



보충 (6/8)

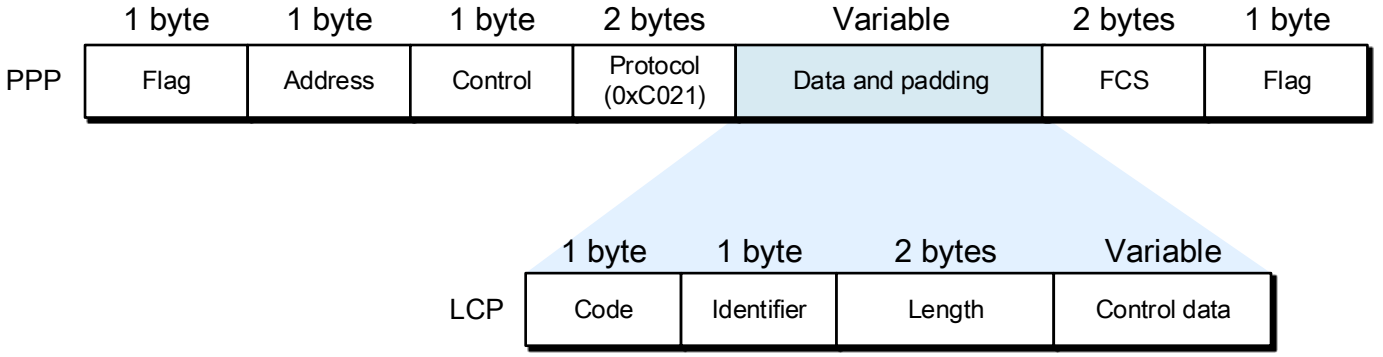
- PPP (6/8)
  - LCP (1/3)
    - 정의
      - PPP 링크의 회선 관리를 담당하는 프로토콜
    - 주요 기능
      - PPP 데이터 링크를 개설, 유지, 종료함
      - 인증 프로토콜의 종류를 결정함
    - 구조 (1/2)
      - 코드
        - LCP 패킷 타입을 결정함
        - 링크 상태에 따라 다른 코드를 사용함



링크 상태	코드	프레임 유형
링크 설정	0x01	설정 요청 (Configure Request)
	0x02	설정 승인 (Configure Ack)
	0x03	설정 비승인 (Configure Nak)
	0x04	설정 거부 (Configure Reject)
링크 종료	0x05	종료 요청 (Terminate Request)
	0x06	종료 승인 (Terminate Ack)
링크 유지	0x07	코드 거부 (Code Reject)
	0x08	프로토콜 거부 (Protocol Reject)
	0x09	에코 요청 (Echo Request)
	0x0A	에코 응답 (Echo Reply)
	0x0B	버림 요청 (Discard Request)

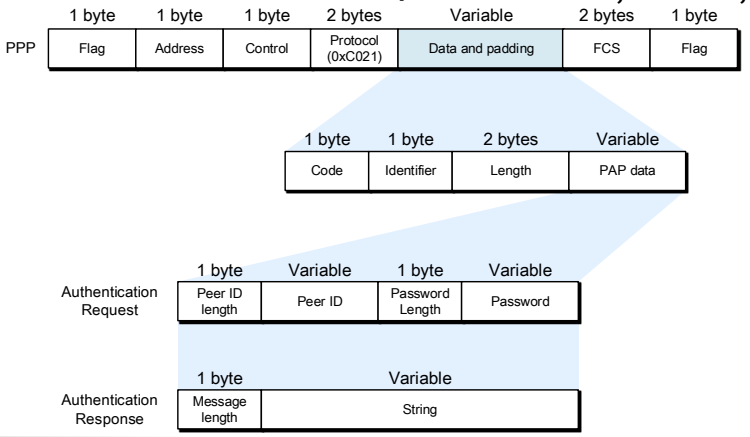
보충 (7/8)

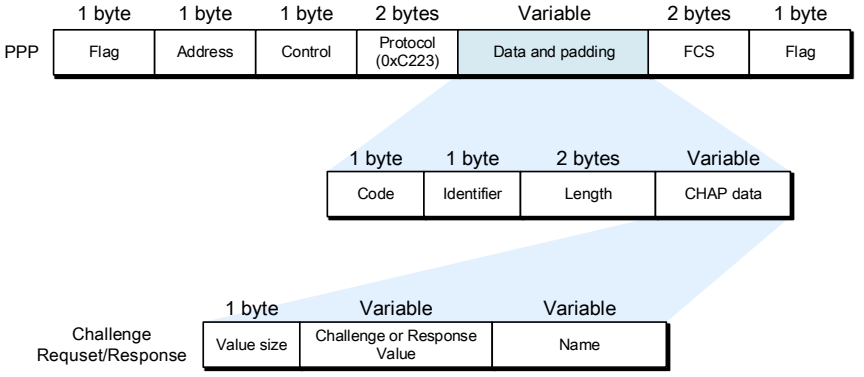
- PPP (7/8)
  - LCP (2/3)
    - 구조 (2/2)
      - 식별자
        - LCP 패킷 요청, 응답 간 번호 일치를 위한 일련번호임
      - 길이
        - LCP 패킷 전체의 길이를 가짐
          - 코드 + 식별자 + 길이 + 제어데이터
      - 제어데이터
        - LCP 패킷 협상용 옵션 값들이 들어있음



보충 (8/8)

- PPP (8/8)
  - LCP (3/3)
    - 인증 프로토콜
      - PAP (Password Authentication Protocol)
        - 인증 요청, 인증 응답 2개의 절차를 수행함
        - 요청 시, ID 및 패스워드를 제어데이터에 보냄
        - 응답 시, 인증 승인 또는 거부를 나타내는 문자열을 제어데이터에 보냄
      - CHAP (Challenge Handshake Authentication Protocol)
        - PAP 방식에서 평문으로 ID, 패스워드를 사용하는 점을 해시 함수를 통해 개선함
        - 챌린지 메시지 전송, 응답, 인증 승인 또는 거부 3단계 절차를 수행함



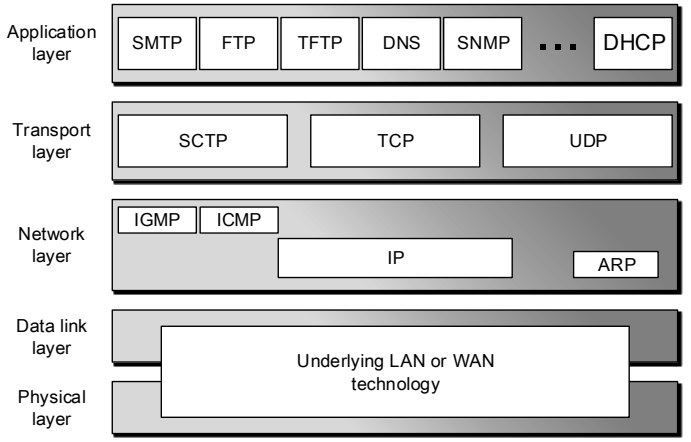


목 차

- 보충
- IPv4
- ARP

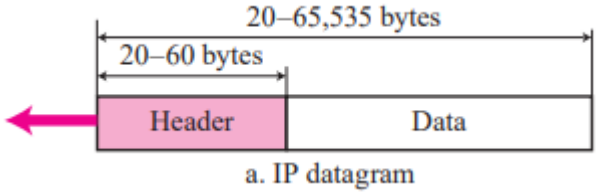
IPv4 (1/62)

- 인터넷 프로토콜 (IP, Internet Protocol)
  - 정의
    - 네트워크 계층에서 TCP/IP 프로토콜이 사용하는 전송 메커니즘
  - 특징
    - 신뢰성 (에러제어) 및 흐름제어 기능이 없음
      - 데이터 전송을 위해 최대의 노력을 하지만, 전송을 확실히 보장하지는 않아 최선 노력 서비스 (Best-Effort Service)라고도 함
    - 비연결성 (Connectionless) 데이터그램 방식임



IPv4 (2/62)

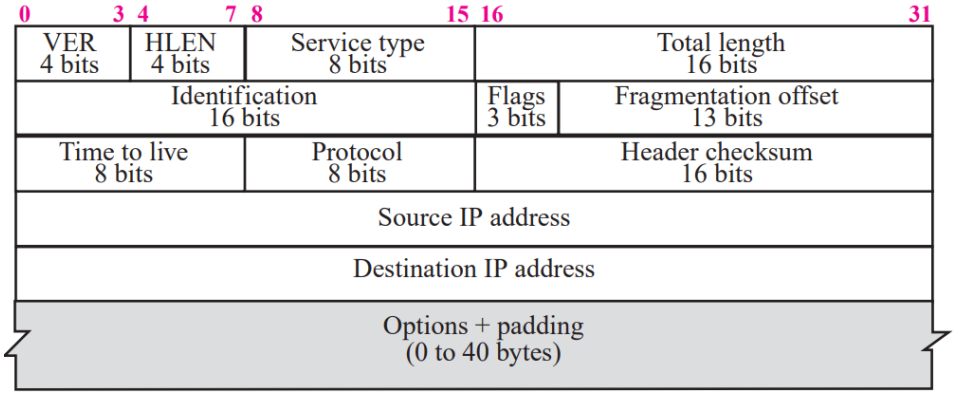
- 데이터그램 (Datagram) (1/12)
  - 정의
    - 패킷 교환에서 독립적으로 취급되는 각각의 패킷
  - 특징
    - 가변 길이를 가짐
      - 헤더는 20~60바이트의 크기를 가짐
    - 헤더와 데이터로 구성됨
    - 모든 상위 계층 프로토콜들 (e.g., TCP, UDP, ICMP, IGMP 등)이 IP 데이터그램에 실려서 전송됨



a. IP datagram

IPv4 (3/62)

- 데이터그램 (2/12)
  - 구조 (1/7)
    - 버전 (VER)
      - 의미
        - IP 프로토콜의 버전을 나타냄
      - 특징
        - 4비트의 크기를 가짐
        - IPv4에서는 4, IPv6에서는 6의 값을 가짐
    - 헤더 길이 (HLEN)
      - 의미
        - 데이터그램 헤더 길이를 나타냄
      - 특징
        - 4비트의 크기를 가짐
        - 4바이트 단위임
          - e.g., 5의 값을 가지면 헤더 길이는 20바이트임 ($5 \times 4 = 20$)



IPv4 (4/62)

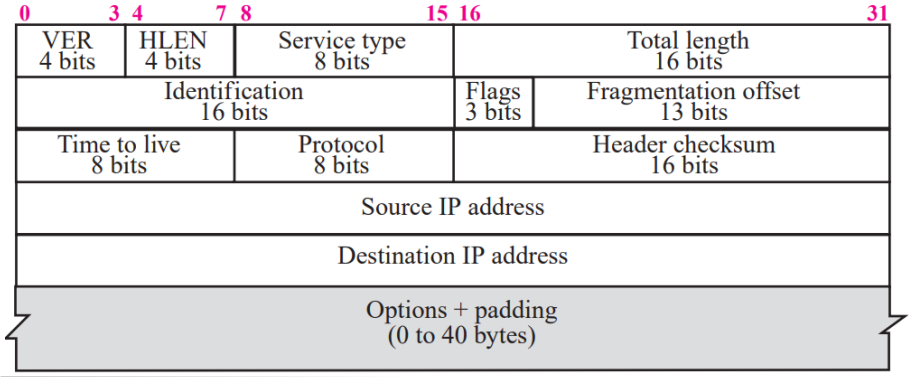
- 데이터그램 (3/12)
  - 구조 (2/7)
    - 서비스 유형 (Service type) (1/2)
      - 의미
        - 데이터그램이 어떻게 처리되어야 하는지를 나타냄
      - 특징
        - 8비트의 크기를 가짐
        - 초기 설계에서는 TOS (Type of Service)라 불림
        - 차별화 서비스 (DiffServ, Differentiated Services) 집합으로 불리며 DSCP (Differentiated Service Code Point)를 포함함
          - IETF가 필드의 의미를 수정하기 전에는 우선순위 및 지연, 처리율, 신뢰성 등을 정의하기 위해 사용됨

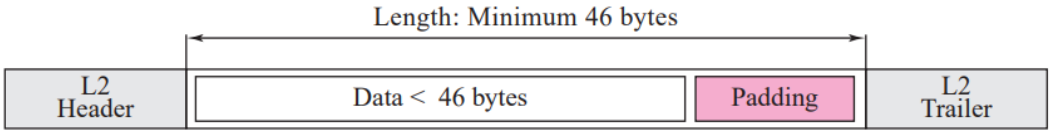
IPv4 (5/62)

- 데이터그램 (4/12)
  - 구조 (3/7)
    - 서비스 유형 (2/2)
      - DSCP
        - 오른쪽 세 비트가 0인 경우
          - 0 ~ 7 사이의 데이터그램 우선순위를 표현함
        - 오른쪽 세 비트가 0이 아닌 경우
          - 부여된 우선순위에 따라 56가지 서비스를 정의함
          - 인터넷 당국 (IETF), 지역 당국, 실험 목적에 따라 범주가 나뉨
      - ECN (Explicit Congestion Notification)
        - 00: 패킷이 ECN 기능을 사용하지 않음
        - 01 or 10: 종단점이 ECN 기능을 수용함을 나타냄
        - 11: 라우터에 혼잡이 발생함

IPv4 (6/62)

- 데이터그램 (5/12)
  - 구조 (4/7)
    - 전체 길이 (Total length)
      - 의미
        - 헤더와 데이터를 포함하는 IP 데이터그램의 전체 길이를 나타냄
      - 특징
        - 16비트의 크기를 가짐
        - 1바이트 단위를 사용함
        - 데이터그램에서 패딩과 데이터를 구분하기 위해서 사용함



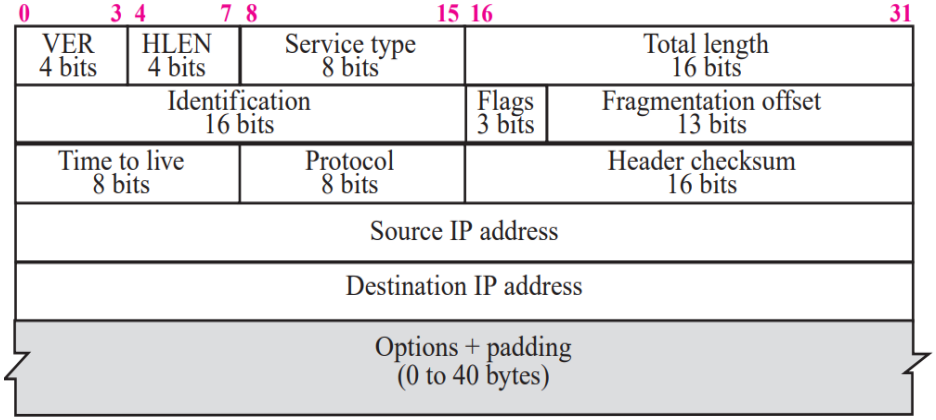


IPv4 (7/62)

- 데이터그램 (6/12)

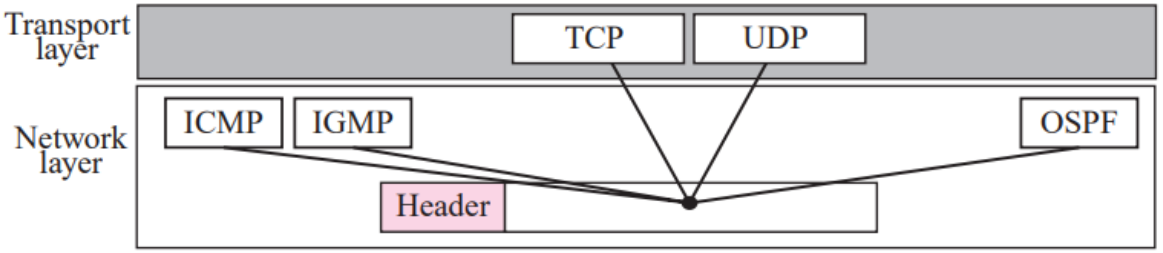
*홉 (Hop): 출발지에서 목적지로 가는 중에 경유하는 라우터 혹은 다른 네트워크 장치

  - 구조 (5/7)
    - 수명 (TTL, Time to live)
      - 의미
        - 데이터그램이 방문할 수 있는 최대 홉* 수를 나타냄
      - 특징
        - 8비트의 크기를 가짐
        - 홉을 경유할 때 마다 필드의 값이 1씩 감소하며, 0이 되면 데이터그램이 폐기됨
    - 프로토콜 (Protocol) (1/2)
      - 의미
        - IP 계층 서비스를 사용하는 상위 프로토콜을 정의함
      - 특징
        - 8비트의 크기를 가짐
        - 데이터그램이 최종 목적지에 도달한 경우, 역다중화를 도움



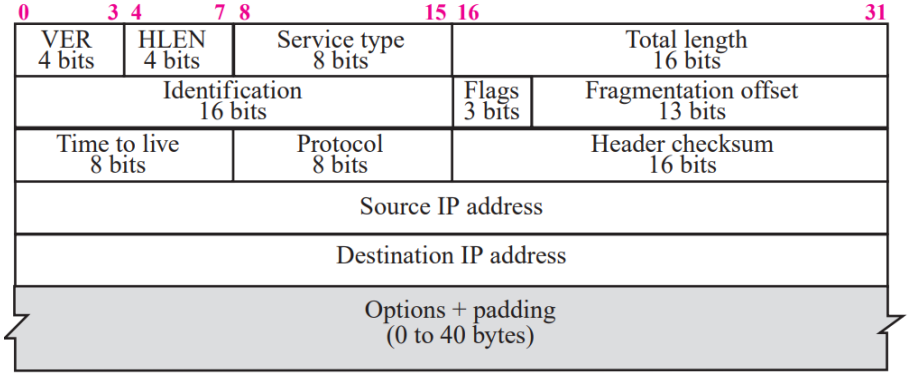
IPv4 (8/62)

- 데이터그램 (7/12)
  - 구조 (6/7)
    - 프로토콜 (2/2)
      - 전송 계층에서의 다중화
        - 캡슐화된 세그먼트들을 네트워크 계층으로 전달하는 작업
      - 전송 계층에서의 역다중화
        - 네트워크 계층을 통해 도착한 데이터를 트랜스포트 계층의 올바른 소켓으로 전달하는 작업



IPv4 (9/62)

- 데이터그램 (8/12)
  - 구조 (7/7)
    - 발신지 주소 (Source Address)
      - 의미
        - 발신지의 IP 주소를 의미함
      - 특징
        - 32비트의 크기를 가짐
        - 데이터그램이 발신지에서 목적지로 전달되는 동안 변해서는 안됨
    - 목적지 주소 (Destination Address)
      - 의미
        - 목적지의 IP 주소를 의미함
      - 특징
        - 32비트의 크기를 가짐
        - 데이터그램이 발신지에서 목적지로 전달되는 동안 변해서는 안됨



IPv4 (10/62)

- 데이터그램 (9/12)
  - 예제 7.1

처음 8비트가 다음과 같은 IP 패킷이 도착하였다.

$$01000010$$

수신자가 이 패킷을 폐기하는 이유를 설명하여라.

- 풀이
  - 왼쪽 4비트(0100)는 버전으로 이 값을 올바름
  - 다음 4비트(0010)는 헤더의 길이, $2 \times 4 = 8$이므로, 헤더 길이가 8바이트임을 의미함
  - 헤더의 길이는 최소 20 바이트임

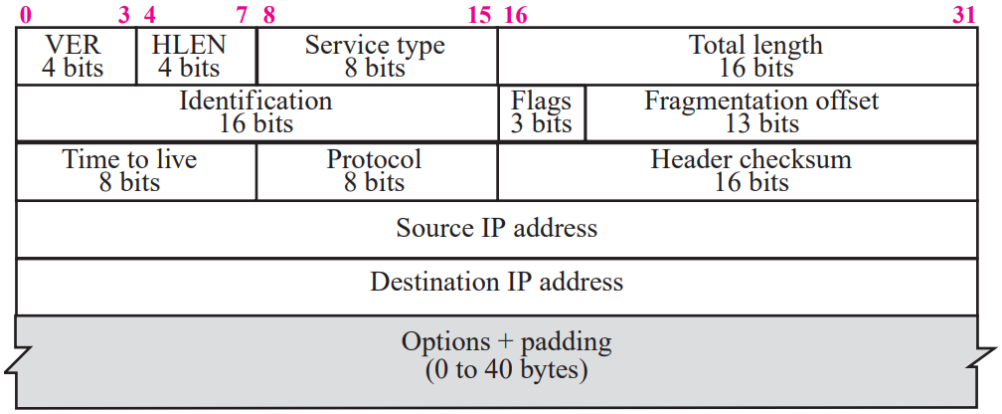
  ∴ 패킷은 전송 도중 훼손되었으므로 폐기됨

IPv4 (11/62)

- 데이터그램 (10/12)
  - 예제 7.2

IP 패킷에서 HLEN 값이 $1000_{(2)}$이다.
이 패킷에는 옵션이 몇 바이트 있는가?

- 풀이
  - HLEN 값은 8이고, 이는 헤더의 총 길이가 $8 \times 4 = 32$바이트임을 의미함
  - 처음 20바이트는 주 헤더이므로, 나머지 12바이트가 옵션이 됨



IPv4 (12/62)

- 데이터그램 (11/12)
  - 예제 7.3

IP 패킷에서 HLEN 값이 0x5이고, 전체 길이 필드 값이 0x28이다.
이 패킷은 몇 바이트의 데이터를 전송하는가?

- 풀이
  - HLEN 값이 5이므로 헤더는 $5 \times 4 = 20$바이트이고, 옵션은 없음
  - 전체 길이는 0x28, 즉 40바이트임
  - $40 - 20 = 20$이므로, 20바이트의 데이터가 전송되고 있음

IPv4 (13/62)

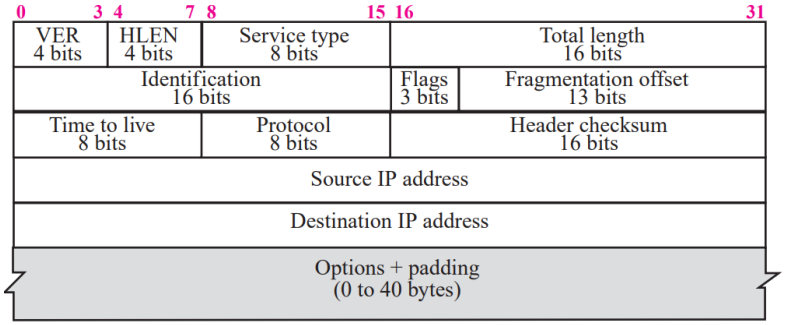
- 데이터그램 (12/12)
  - 예제 7.4

16진수 형식의 첫 패킷이 다음과 같이 도착하였다.

45 00 00 28 00 01 00 00 01 02

이 패킷은 폐기되기 전에 몇 홉을 지나갈 수 있으며, 어떤 상위계층 프로토콜 데이터를 전송하는가?

- 풀이
  - 수명 필드는 9번째 바이트에 위치하며 01의 값을 가지므로, 패킷이 한 홉만 갈 수 있음
  - 프로토콜 필드는 다음 필드로 02의 값을 가지고, 이는 상위 계층 프로토콜이 IGMP임을 의미함

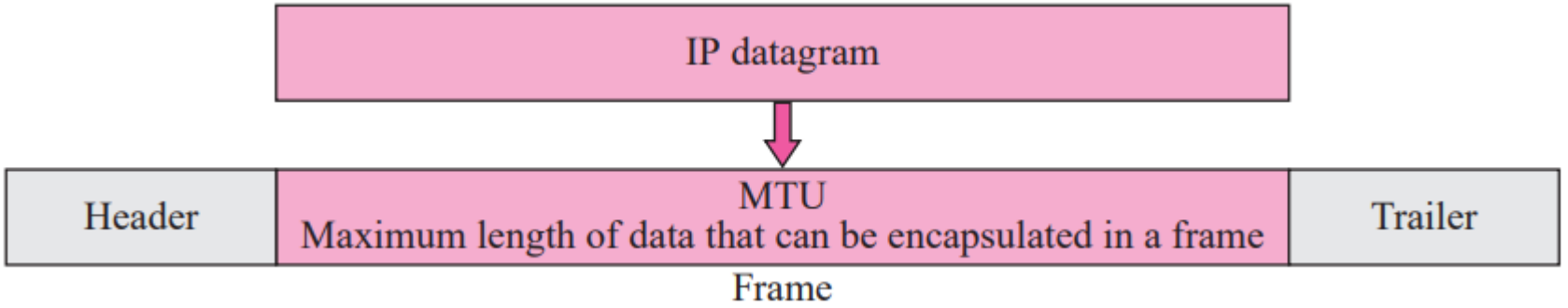


IPv4 (14/62)

- IP 단편화 (IP Fragmentation) (1/11)
  - 정의
    - 큰 IP 패킷을 전송 가능한 작은 조각으로 나누어 전달하는 과정
  - 특징
    - 네트워크의 최대 전달 단위 (MTU, Maximum Transfer Unit) 보다 큰 패킷이 있을 때 발생함
    - 데이터그램 내의 데이터만 단편화됨
    - 단편화가 수행되면 일반적으로 최종 목적지에서 재조립 (Reassembly)됨
    - 최종 목적지에 도착하기까지 여러 번 수행될 수 있음
    - 각 조각의 헤더에서 플래그, 단편화 옵셋, 전체 길이를 제외한 나머지 필드들은 공통으로 사용됨

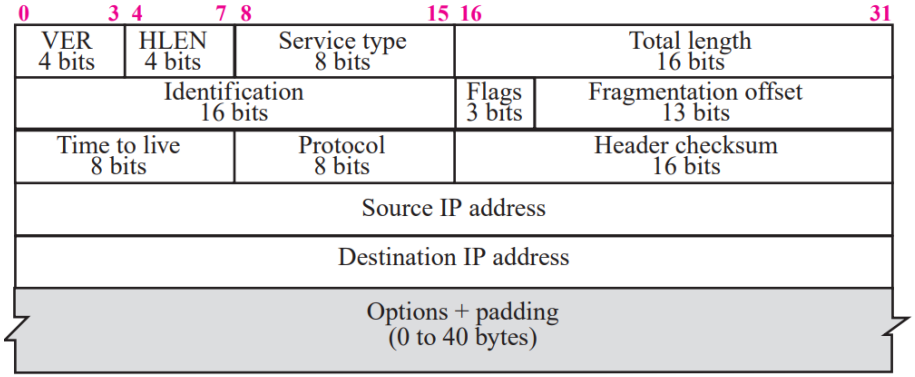
IPv4 (15/62)

- IP 단편화 (2/11)
  - MTU
    - 정의
      - 하나의 프레임에 담아 운반 가능한 최대 크기
    - 특징
      - 데이터그램이 프레임 속에 캡슐화될 때 데이터그램의 크기는 MTU 보다 작아야 함
      - MTU는 네트워크 내에서 사용되는 하드웨어와 소프트웨어에 의해 결정됨
        - e.g., Ethernet LAN은 1500바이트, FDDI LAN은 4352바이트, PPP는 296바이트의 MTU를 가짐



IPv4 (16/62)

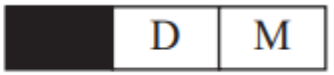
- IP 단편화 (3/11)
  - 관련 필드 (1/4)
    - 식별자 (Identification)
      - 의미
        - 발신지 호스트로부터 나온 데이터그램을 유일하게 식별하는 값임
      - 특징
        - 16비트의 크기를 가짐
        - 데이터그램을 보낼 때마다 카운터를 통해 1씩 증가됨
        - 단편화된 조각들은 같은 식별자 값을 가짐



IPv4 (17/62)

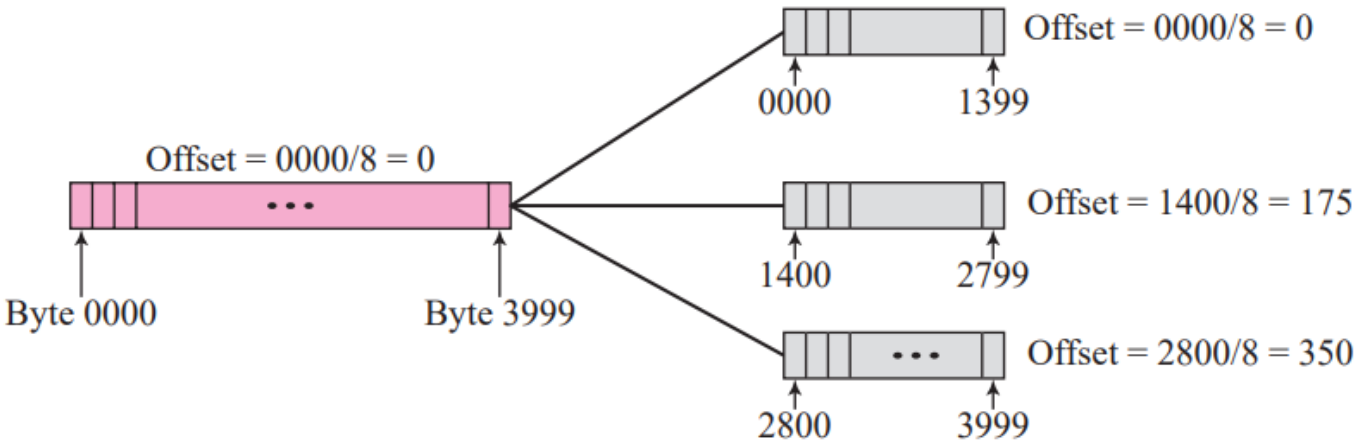
- IP 단편화 (4/11)
  - 관련 필드 (2/4)
    - 플래그 (Flag)
      - 의미
        - 단편화와 관련한 옵션을 나타냄
      - 특징
        - 두 번째 비트의 값이 1이면, 데이터그램을 단편화해서는 안되는 것을 의미함
          - 단편화를 수행해야 하는데 이 플래그가 설정되어 있으면, 데이터그램을 폐기하고 ICMP 오류 메시지를 발신지에 보냄
        - 두 번째 비트의 값이 0이면, 필요한 경우 데이터그램을 단편화 할 수 있음
        - 세 번째 비트의 값이 1이면, 마지막 단편이 아님을 의미함
        - 세 번째 비트의 값이 0이면, 마지막 단편이거나 유일한 단편임을 의미함

D: Do not fragment
M: More fragments



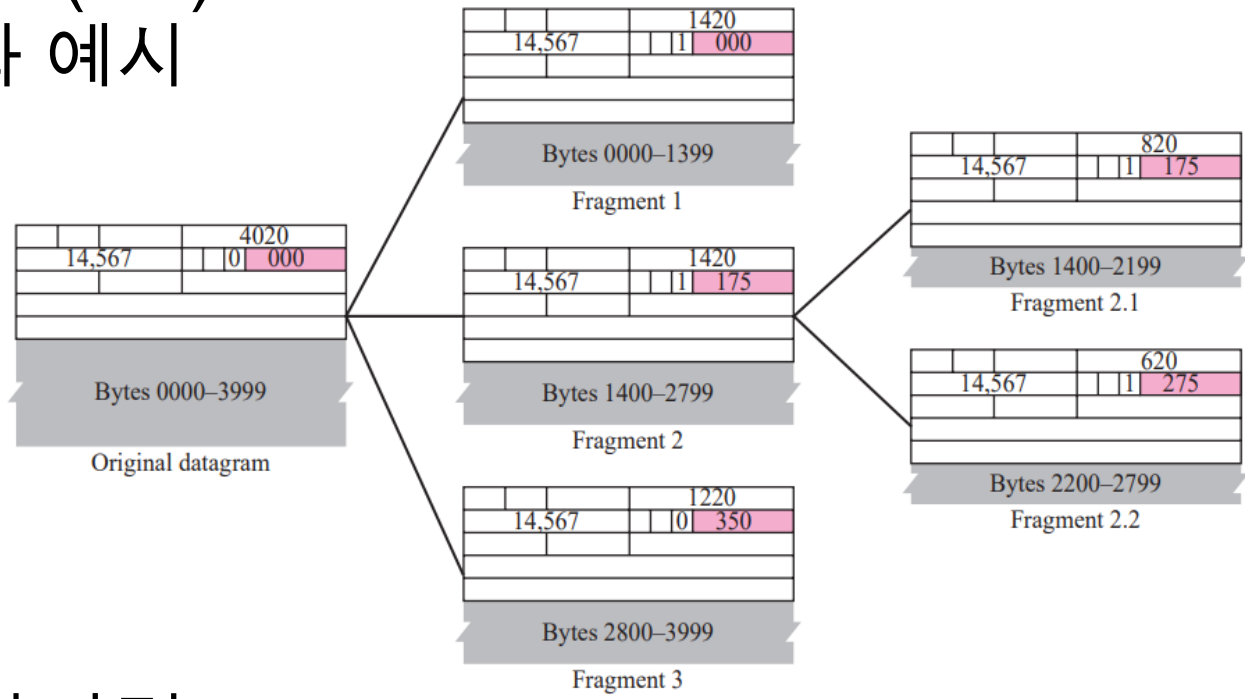
IPv4 (18/62)

- IP 단편화 (5/11)
  - 관련 필드 (3/4)
    - 단편화 옵셋 (Fragmentation Offset)
      - 의미
        - 전체 데이터그램 내에서 단편의 상대적 위치를 나타냄
      - 특징
        - 13비트의 크기를 가짐
        - 원 데이터그램에서 옵셋을 8바이트 단위로 표현함
          - 데이터그램을 단편화 할 때, 각 단편의 첫 번째 바이트 번호가 8로 나누어 떨어져야 함



IPv4 (19/62)

- IP 단편화 (6/11)
  - 관련 필드 (4/4)
    - 단편화 예시



    - 재조립 과정
      1. 첫 번째 단편의 옵셋 값을 확인함
      2. 첫 번째 단편의 길이를 8로 나누고, 두 번째 옵셋 값이 이와 같은지 확인함
      3. 이 과정을 반복, 마지막 옵셋은 More fragments 값이 0임

IPv4 (20/62)

- IP 단편화 (7/11)
  - 예제 7.5

M(More fragments) 비트 값이 0인 패킷이 도착하였다. 이것은 첫 번째 단편인가, 마지막 단편인가 또는 중간 단편인가? 패킷이 단편화되었는지 알 수 있는가?

- 풀이
  - M 비트가 0이면 단편이 더 이상 없다는 의미로, 해당 패킷은 마지막 단편임
  - 원래의 패킷이 단편화되었는지는 알 수 없음

IPv4 (21/62)

- IP 단편화 (8/11)
  - 예제 7.6

M 비트 값이 1인 패킷이 도착하였다. 이것은 첫 번째 단편인가, 마지막 단편인가 또는 중간 단편인가? 패킷이 단편화되었는지 알 수 있는가?

- 풀이
  - M 비트가 1이면 단편이 더 있다는 의미이므로, 이 패킷은 첫 번째 또는 중간 단편임
  - 첫 번째인지, 중간 단편인지를 알기 위해서는 단편화 옵셋과 같은 추가 정보가 필요함
  - 패킷이 단편화되었다는 것을 알 수 있음

IPv4 (22/62)

- IP 단편화 (9/11)
  - 예제 7.7

M 비트 값이 1이고 단편화 옵셋이 0인 패킷이 도착하였다. 이것은 첫 번째 단편인가, 마지막 단편인가 또는 중간 단편인가?

- 풀이
  - M 비트가 1이므로, 첫 번째 또는 중간 단편임
  - 단편화 옵셋이 0이므로, 첫 번째 단편임

IPv4 (23/62)

- IP 단편화 (10/11)
  - 예제 7.8

옵셋 값이 100인 패킷이 도착하였다. 해당 패킷의 첫 번째 바이트는 몇 번째인가? 마지막 바이트가 몇 번째인지 알 수 있는가?

- 풀이
  - 첫 번째 바이트의 번호를 구하기 위해, 옵셋 값을 8로 곱하면 800번째라는 것을 알 수 있음
  - 데이터의 길이가 주어지지 않았기 때문에, 마지막 바이트가 몇 번째인지 알 수 없음

IPv4 (24/62)

- IP 단편화 (11/11)
  - 예제 7.9

옵셋 값이 100이고 HLEN이 5이며 전체 길이가 100인 패킷이 도착하였다. 해당 패킷의 첫 번째 바이트와 마지막 바이트가 몇 번째인지 구하여라.

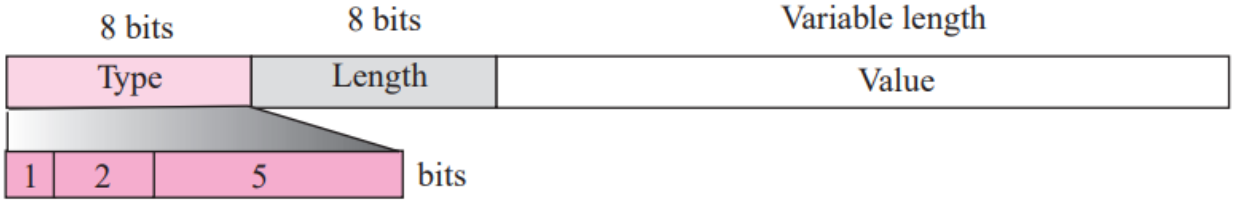
- 풀이
  - 첫 번째 바이트는 $100 \times 8 = 800$번째임
  - 전체 길이는 100 바이트이고 헤더 길이는 20바이트 ($\because 5 \times 4$)이므로, 데이터그램 내에는 80바이트의 데이터가 있음
  - 첫 번째 바이트가 800번째 이므로, 마지막 바이트는 879번째임

IPv4 (25/62)

- 옵션 필드 (1/3)
  - 구조 (1/3)
    - 유형 (Type) (1/2)
      - 복사 (Copy)
        - 단편화에 옵션을 포함시킬 것인지 아닌지를 의미함
      - 클래스 (Class)
        - 옵션의 일반적인 목적을 나타냄

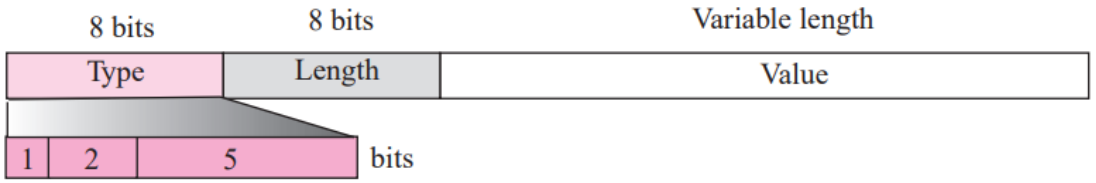
복사 필드	의미
0	옵션이 첫 번째 단편에만 들어있음
1	옵션이 모든 단편에 들어있음

클래스 필드	의미
00	옵션이 데이터그램 제어에 사용됨
01	옵션이 디버그나 관리 목적임
10	N/A
11	N/A



IPv4 (26/62)

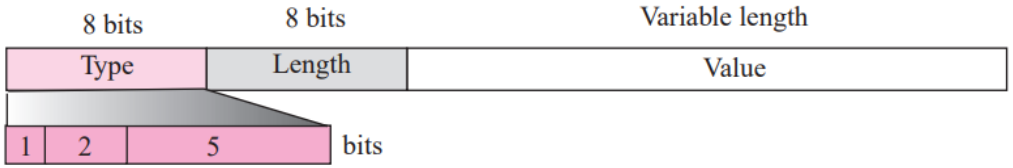
- 옵션 필드 (2/3)
  - 구조 (2/3)
    - 유형 (2/2)
      - 번호 (Number)
        - 옵션의 유형을 정의함
        - 아직 모든 옵션 유형이 정의되지는 않음



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

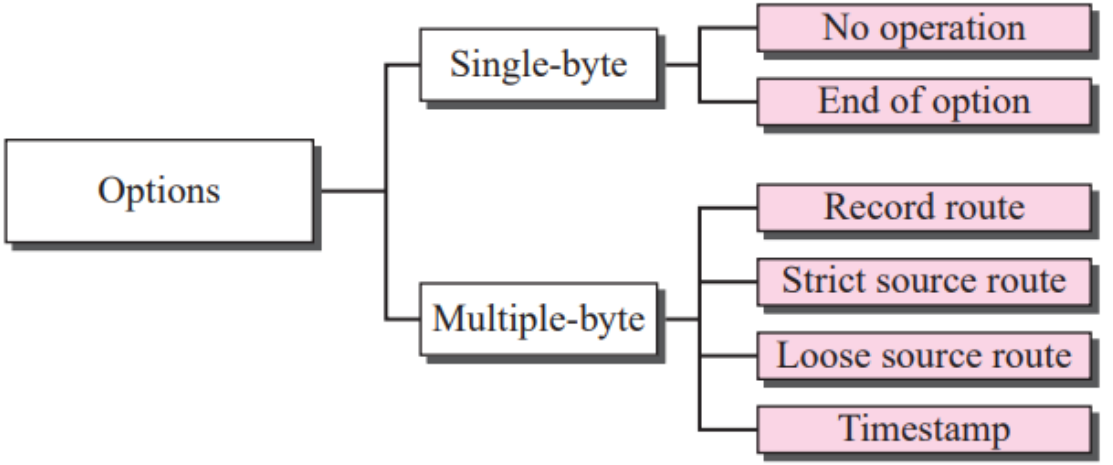
IPv4 (27/62)

- 옵션 필드 (3/3)
  - 구조 (3/3)
    - 길이 (Length)
      - 의미
        - 옵션 필드의 전체 길이를 정의함
      - 특징
        - 8비트의 크기를 가짐
        - 모든 옵션 유형에 있는 것은 아님
    - 값 (Value)
      - 의미
        - 옵션이 필요로 하는 데이터를 포함함
      - 특징
        - 가변적인 크기를 가짐
        - 모든 옵션 유형에 있는 것은 아님



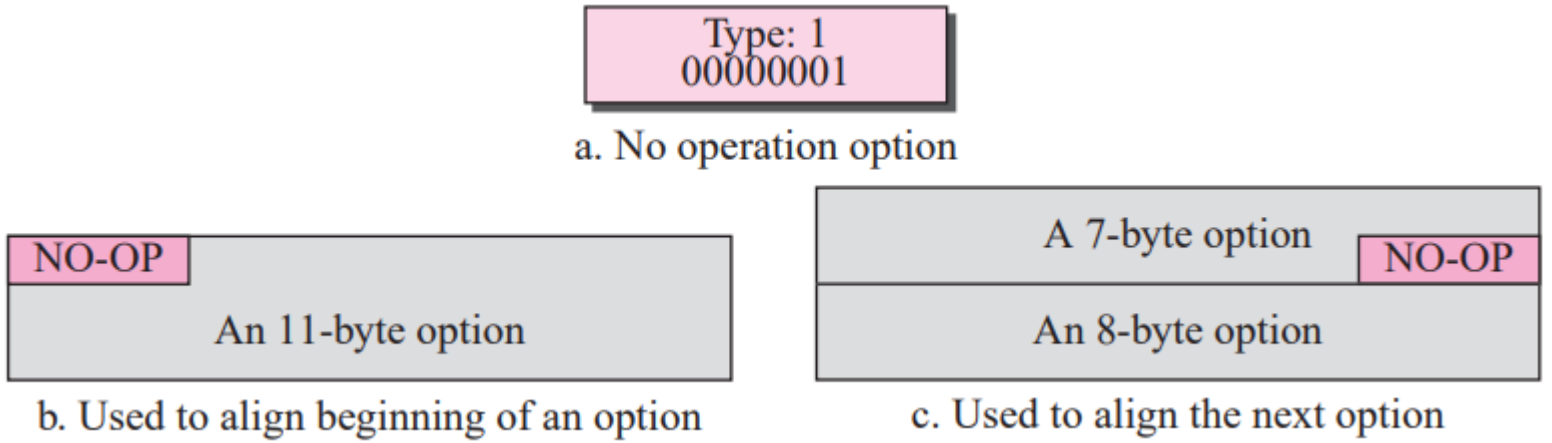
IPv4 (28/62)

- 옵션 유형 (1/13)
  - Single-byte 옵션
    - 정의
      - 단일 바이트로 구성되는 옵션
    - 특징
      - Type 값만 사용함
      - Padding용으로, 특별한 역할을 수행하지 않음
  - Multiple-byte 옵션
    - 정의
      - 2바이트 이상의 길이로 구성되는 옵션
    - 특징
      - Type, Length, Value 값을 사용함



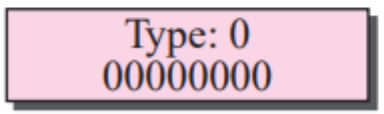
IPv4 (29/62)

- 옵션 유형 (2/13)
  - Single-byte 옵션 (1/2)
    - 무연산 (No operation)
      - 목적
        - 옵션들 사이의 여백을 채워주기 위해 사용됨
      - 특징
        - 여러 번 사용 가능함

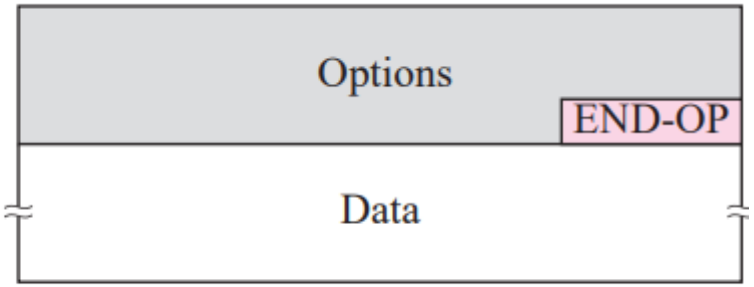


IPv4 (30/62)

- 옵션 유형 (3/13)
  - Single-byte 옵션 (2/2)
    - 옵션 종료 (End of option)
      - 목적
        - 옵션의 필드 끝에 패딩 목적으로 사용됨
      - 특징
        - 마지막 옵션으로만 사용 가능함
        - 한 번만 사용 가능함



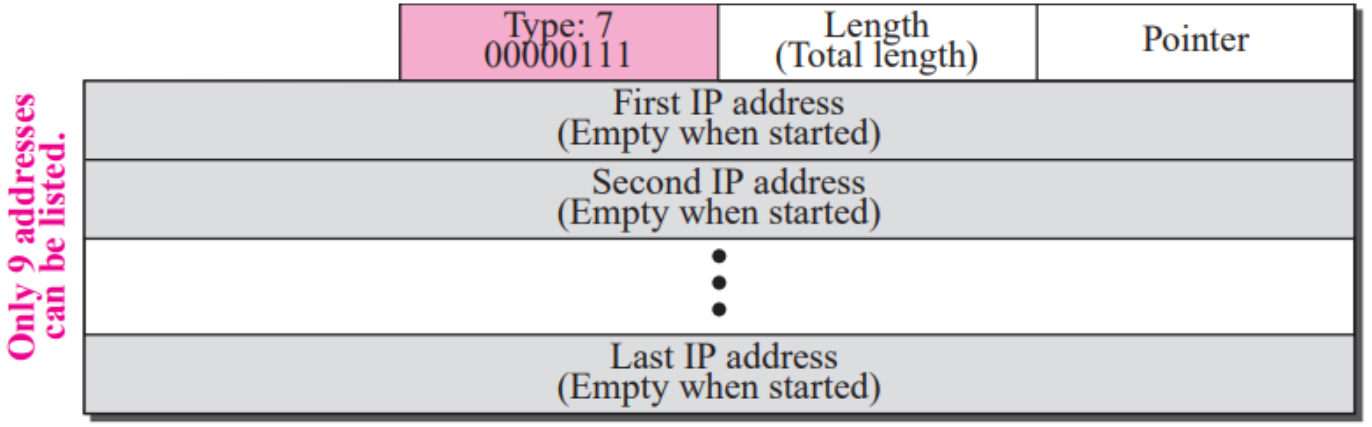
a. End of option



b. Used for padding

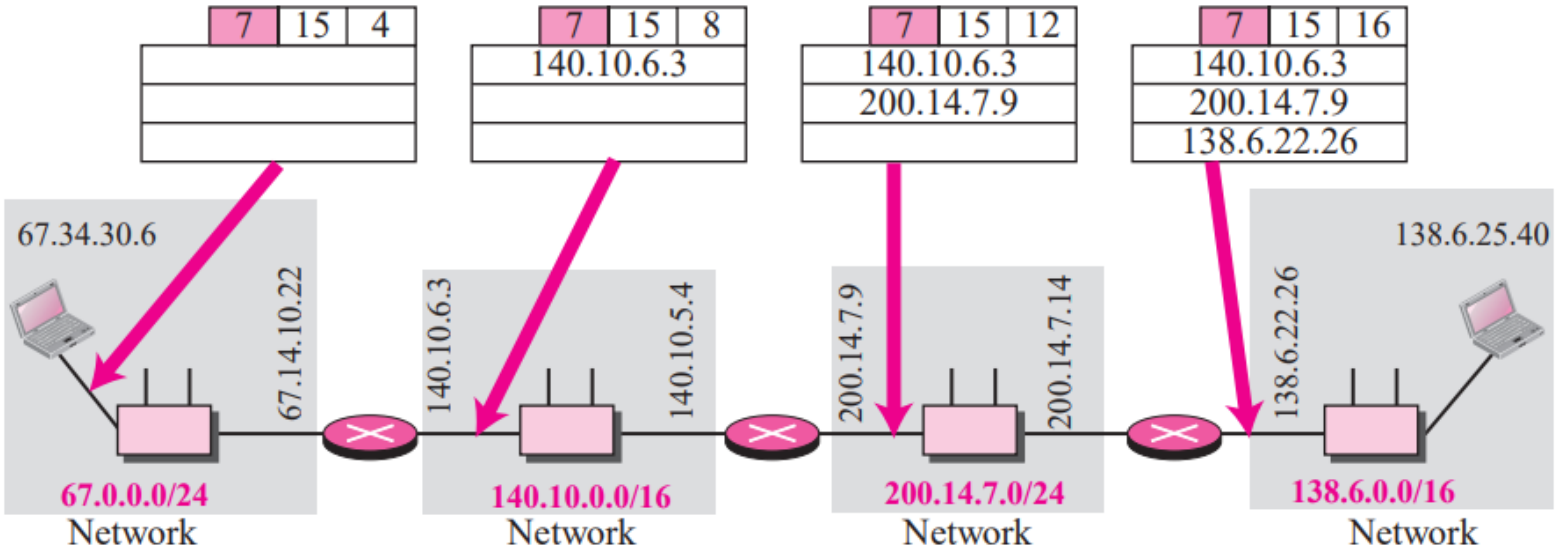
IPv4 (31/62)

- 옵션 유형 (4/13)
  - Multiple-byte 옵션 (1/8)
    - 경로 기록 (Record route) (1/2)
      - 목적
        - 데이터그램을 처리한 인터넷 라우터들을 기록하기 위함임
      - 특징
        - 최대 9개의 IP 주소를 기록 가능함
        - 포인터 필드는 기본 4의 값을 가지며, 빈 필드를 가리키고 있음



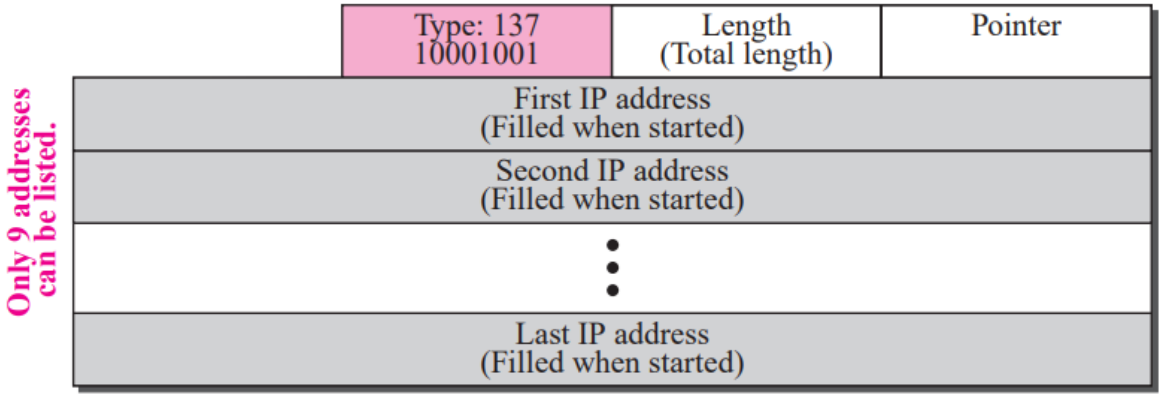
IPv4 (32/62)

- 옵션 유형 (5/13)
  - Multiple-byte 옵션 (2/8)
    - 경로 기록 (2/2)
      - 라우터에서의 처리 예시
        1. 데이터그램을 처리하는 라우터는 포인터 필드 값과 길이 값을 비교함
        2. 빈 필드에 자신의 IP 주소를 기록함
           - 출력 인터페이스의 IP 주소가 기록됨
        3. 포인터 값을 4 증가시키고 다음 경로로 보냄



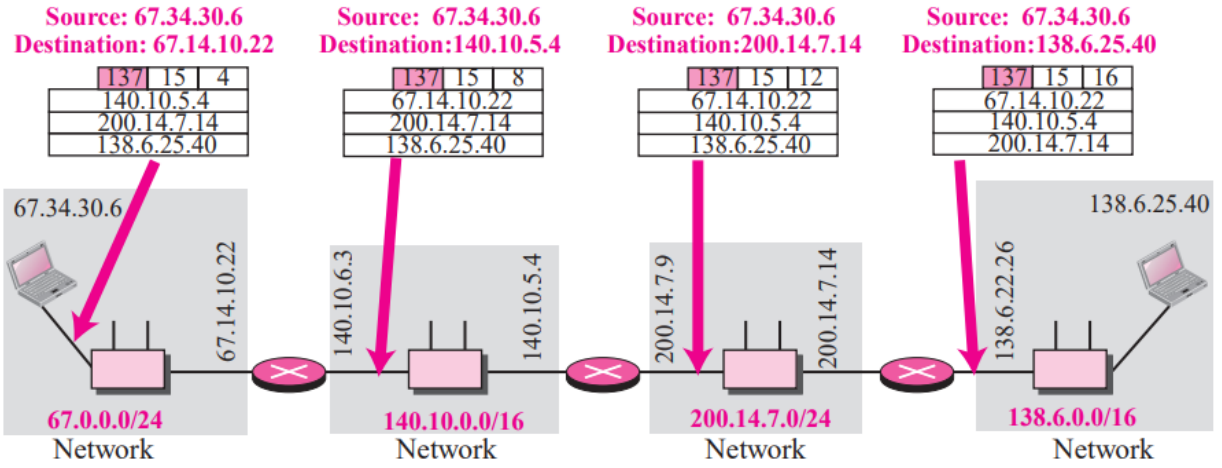
IPv4 (33/62)

- 옵션 유형 (6/13)
  - Multiple-byte 옵션 (3/8)
    - 엄격한 발신지 경로 (Strict source route) (1/2)
      - 목적
        - 데이터그램이 인터넷에서 거쳐야 할 경로를 미리 지정하기 위함
      - 특징
        - 해당 옵션을 통해, 서비스 유형에 맞춰 최소지연, 최대 처리율, 신뢰성 등을 가지는 경로를 선택할 수 있음
        - 해당 옵션이 지정되면, 데이터그램은 옵션에만 정의되어 있는 모든 라우터를 방문해야 함
          - 리스트에 없는 라우터를 방문하거나 방문하지 않은 라우터가 있으면 해당 데이터그램은 폐기되고 발신지에 오류 메시지가 보내짐



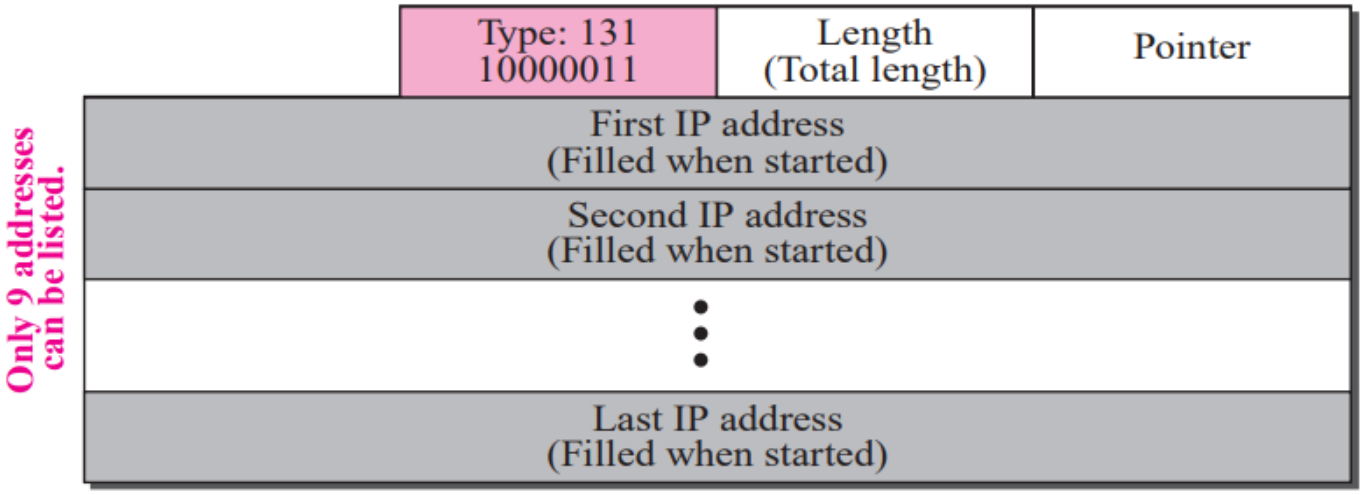
IPv4 (34/62)

- 옵션 유형 (7/13)
  - Multiple-byte 옵션 (4/8)
    - 엄격한 발신지 경로 (2/2)
      - 라우터에서의 처리 예시
        1. 포인터 값과 길이 값을 비교함
        2. 포인터 값이 길이 값보다 크지 않으면, 라우터는 포인터가 가리키는 IP 주소와 라우터에서 데이터그램이 들어온 IP 주소를 비교함
        3. 주소가 같으면 데이터그램을 정상적으로 처리하고 현재 포인터가 가리키는 IP주소를 목적지 IP주소와 교체함
        4. 포인터 값을 4 증가시킨 후, 데이터그램을 전달함



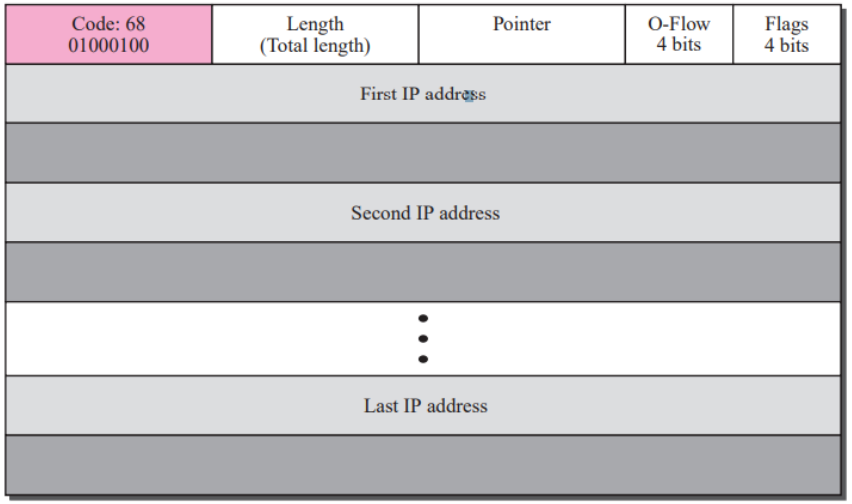
IPv4 (35/62)

- 옵션 유형 (8/13)
  - Multiple-byte 옵션 (5/8)
    - 느슨한 발신지 경로 (Loose source route)
      - 목적
        - 데이터그램이 인터넷에서 거쳐야 할 경로를 미리 지정하기 위함
      - 특징
        - 엄격한 발신지 경로와 비슷하나 제약이 완화되어, 리스트 속의 라우터는 반드시 방문되어야하지만 리스트에 없는 라우터도 방문할 수 있음



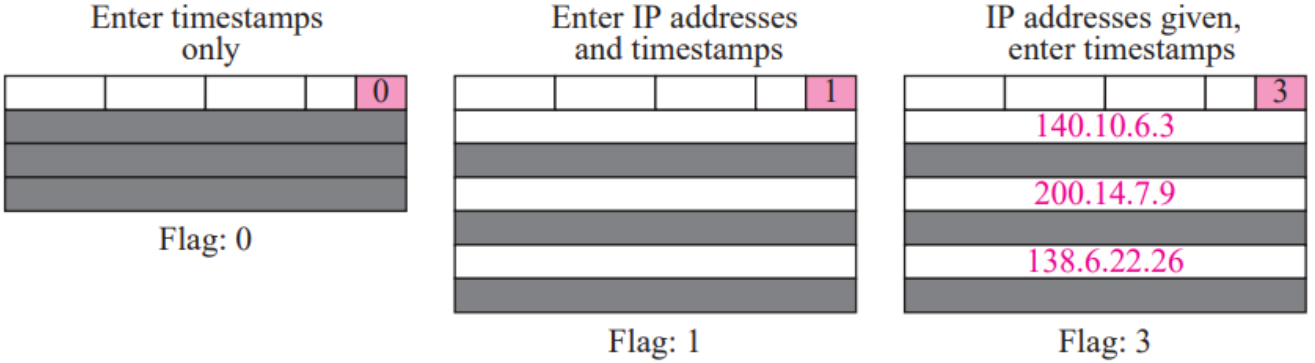
IPv4 (36/62)

- 옵션 유형 (9/13)
  - Multiple-byte 옵션 (6/8)
    - 타임스탬프 (Timestamp) (1/3)
      - 목적
        - 라우터가 데이터그램을 처리하는 시간을 기록하기 위함
      - 특징
        - 시간은 세계 표준시 (Universal Time) 자정으로부터의 millisecond 단위로 표시됨
        - 특별한 권한이 없는 사용자는 인터넷의 물리적 구조를 알 수 없기 때문에, 해당 옵션을 사용할 수 없음



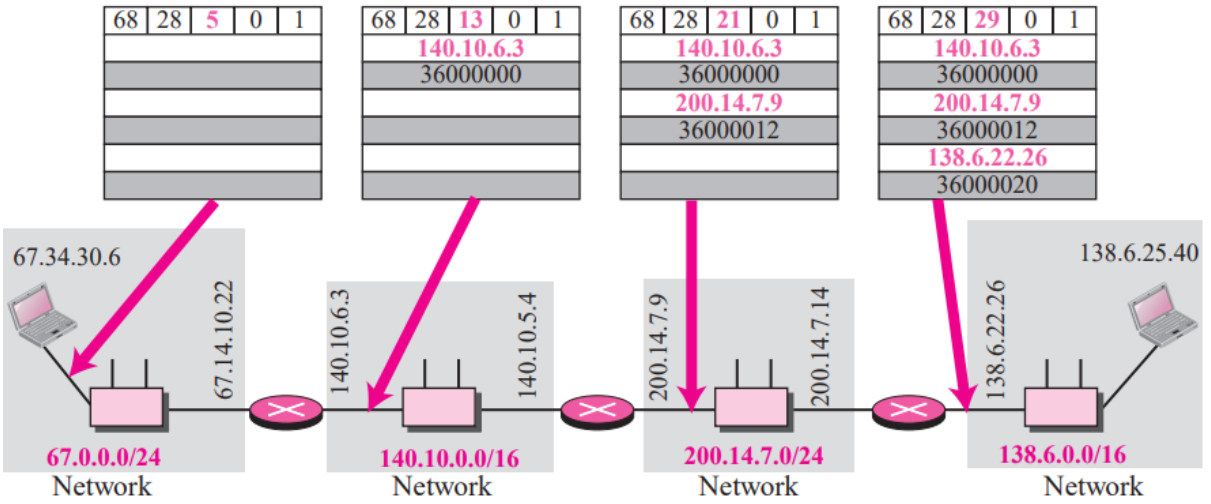
IPv4 (37/62)

- 옵션 유형 (10/13)
  - Multiple-byte 옵션 (7/8)
    - 타임스탬프 (2/3)
      - 구조
        - 오버플로우 (O-flow)
          - 필드가 더 이상 없어서 타임 스탬프를 기록하지 못한 라우터의 수가 기록됨
        - 플래그 (Flags)
          - 0이면 각 라우터는 주어진 필드에 타임스탬프만 추가함
          - 1이면 라우터는 출력 인터페이스 IP주소와 타임스탬프를 기록함
          - 3이면 라우터는 주어진 IP 주소와 입력 인터페이스 IP 주소를 비교하고, 같은 경우에 IP 주소에 출력 인터페이스 주소를 덮어쓰고 타임스탬프를 추가함



IPv4 (38/62)

- 옵션 유형 (11/13)
  - Multiple-byte 옵션 (8/8)
    - 타임스탬프 (3/3)
      - 라우터에서의 처리 예시
        1. 플래그 값이 1임을 확인함
        2. 포인터가 가리키는 필드에 IP주소와 타임스탬프 값을 기록함
        3. 포인터 값을 8 증가시킨 후, 데이터그램을 전달함



IPv4 (39/62)

- 옵션 유형 (12/13)
  - 예제 7.10

다음 3개 옵션 중 어느 것이 각 단편들에 포함되어야 하는가?
a. 00000001
b. 10000011
c. 01000100

- 풀이
  - 각 옵션 필드의 첫 번째(가장 왼쪽) 비트를 통해 알 수 있음
  - a는 무연산 옵션으로, 첫 비트가 0이므로 첫 번째 단편에만 옵션이 포함됨
  - b는 느슨한 발신지 경로 옵션으로, 첫 비트가 1이므로 각 단편들에 옵션이 포함됨
  - c는 타임스탬프 옵션으로, 첫 비트가 0이므로 첫 번째 단편에만 옵션이 포함됨

IPv4 (40/62)

- 옵션 유형 (13/13)
  - 예제 7.11

다음 2개 옵션 중 어떤 것이 데이터그램 제어에 사용되고, 어떤 것이 디버깅과 관리에 사용되는가?

a. 10000011
b. 01000100

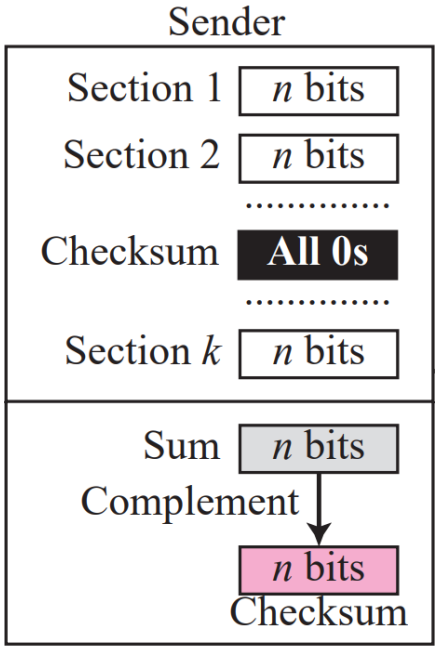
- 풀이
  - 각 옵션 필드의 두 번째와 세 번째 비트를 통해 알 수 있음
  - a는 느슨한 발신지 경로 옵션으로, 00이므로 데이터그램 제어 목적으로 사용됨
  - b는 타임스탬프 옵션으로, 10이므로 디버깅과 관리 목적으로 사용됨

IPv4 (41/62)

- 검사합 (Checksum) (1/4)
  - 정의
    - 데이터 전송 시 오류를 검출하기 위해 사용하는 값
  - 목적
    - 패킷이 전달 중에 발생할 수 있는 오류로부터 패킷을 보호하기 위함임
  - 검증 과정
    1. 송신자에 의해 계산되고 패킷과 함께 전송됨
    2. 수신자는 감사합을 포함하는 전체 패킷에 대해 같은 계산을 수행하여 결과가 일치하는 지 확인함
       - 결과가 일치하면 패킷을 받아들임
       - 결과가 일치하지 않으면 패킷을 폐기함

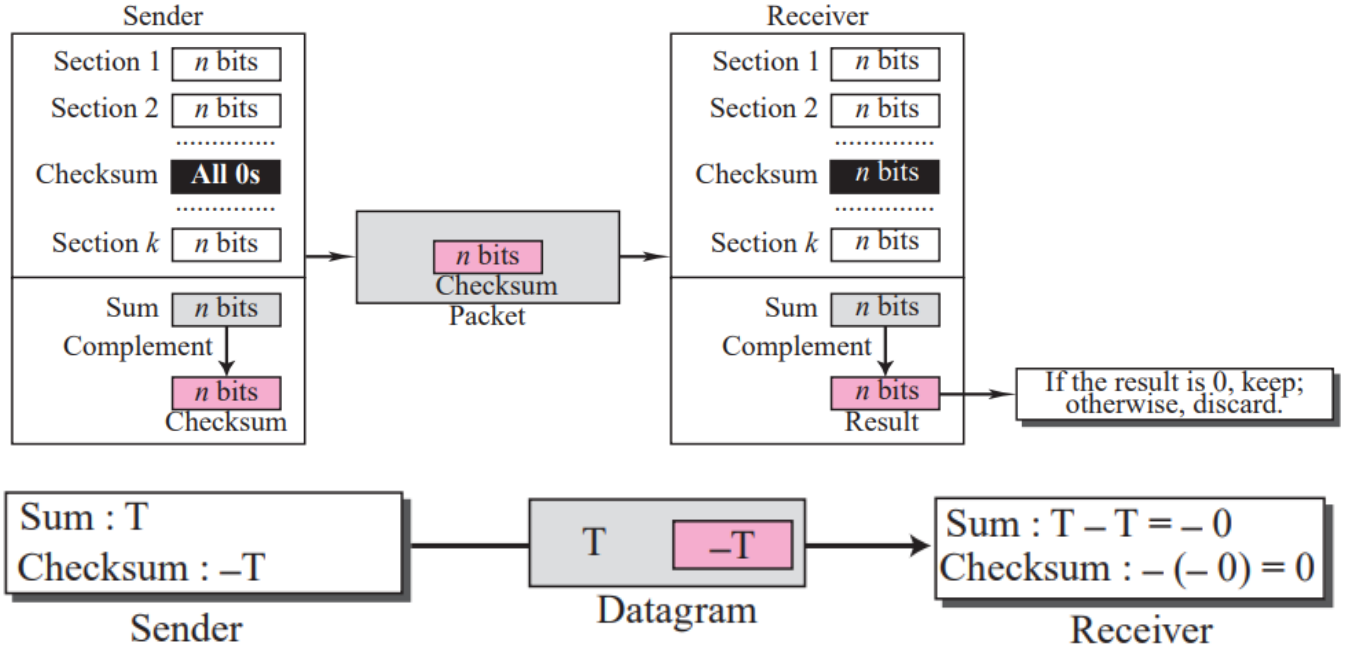
IPv4 (42/62)

- 검사합 (2/4)
  - 송신자의 검사합 계산
    - 패킷을 n비트 조각으로 나눔
      - 일반적으로 n은 16으로 사용함
    - 모든 조각을 1의 보수 연산을 사용하여 합함
    - 합에 대한 1의 보수가 검사합임



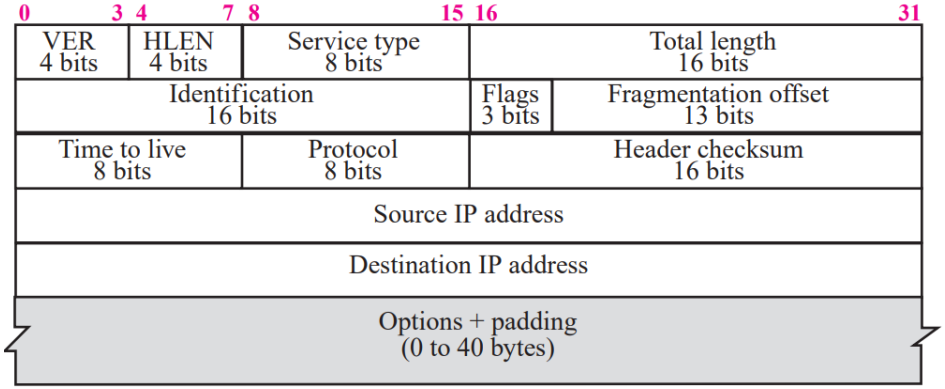
IPv4 (43/62)

- 검사합 (3/4)
  - 수신자의 검사합 계산
    - 수신된 패킷을 크기가 n비트인 조각으로 나눔
    - 모든 조각을 합함
    - 합에 대한 1의 보수를 구함



IPv4 (44/62)

- 검사합 (4/4)
  - IP 패킷의 검사합
    - 과정
      - 검사합 필드를 0으로 세팅함
      - 모든 헤더를 16비트 조각으로 나누고 이들의 합을 구함
      - 합 결과에 대한 보수를 구하여 검사합 필드에 삽입함
    - 특징
      - 헤더만 포함하고 데이터는 포함하지 않음
        - 데이터는 상위 계층에서 검사합이 수행됨
        - 헤더는 라우터를 방문할 때마다 변화하지만, 데이터는 그렇지 않기 때문임

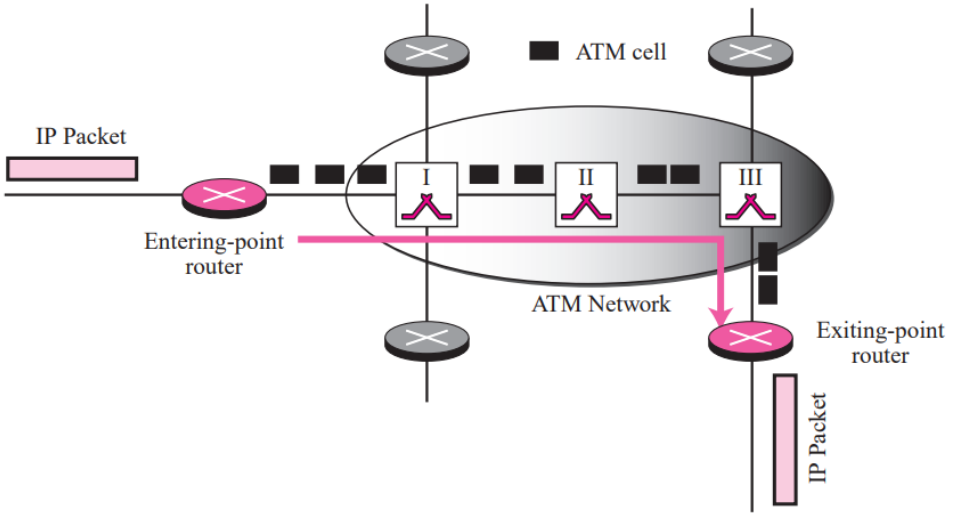


IPv4 (45/62)

- ATM 상 IP (1/3)
  - AAL5
    - 특징
      - 인터넷에서 유일하게 사용되는 AAL 계층임
      - AAL5는 한 IP 데이터그램으로부터 생성된 모든 셀들이 하나의 메시지에 속한다고 가정하기 때문에, 주소 정보나 순서 맞추기 등 다른 헤더를 정보를 제공하지 않음
      - 패딩과 4개의 필드로 된 트레일러만 IP 패킷에 추가됨
    - 사용 이유
      - IP 패킷을 직접 셀로 캡슐화하면 헤더 정보들로 인해 데이터 전송 효율성이 낮아지기 때문에, AAL5를 통해 캡슐화를 수행함
    - 동작 과정
      - 65536바이트보다 크지 않은 IP 패킷을 받아들여, 패딩과 8바이트 트레일러를 추가함
      - 메시지를 48 바이트 조각으로 나누어 ATM 계층으로 전달함

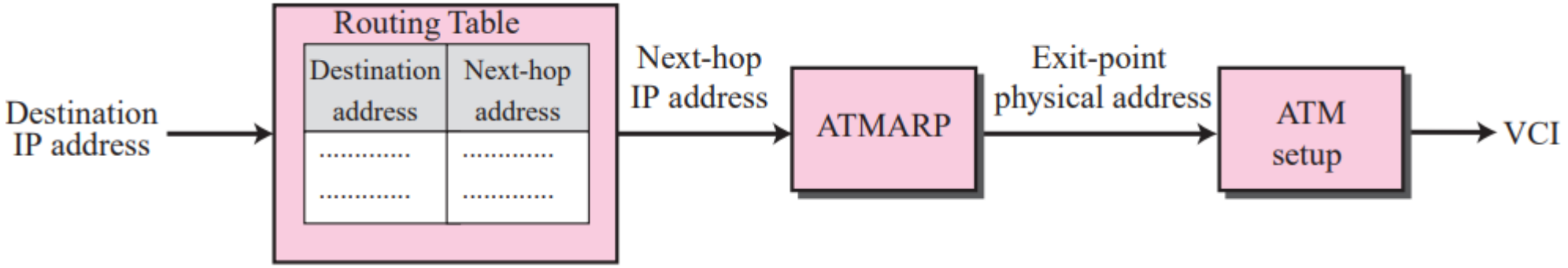
IPv4 (46/62)

- ATM 상 IP (2/3)
  - 셀 라우팅 (1/2)
    - 수행
      - 하나의 진입점 라우터와 진출점 라우터로 라우팅을 수행함
    - 주소
      - IP 주소
        - ATM 네트워크에 연결된 라우터의 IP 주소를 정의함
        - IP 계층에서의 라우터를 정의하며, ATM 네트워크와는 무관함
      - 물리 주소
        - ATM 네트워크를 위해 정의된 20바이트의 주소
        - 연결 설정 동안 사용됨
        - ATM 네트워크와 연관되어 있고 인터넷과는 무관함
      - 가상회선 식별자
        - ATM 네트워크의 교환기가 라우팅에 사용하는 주소
        - 데이터 전송에 사용됨



IPv4 (47/62)

- ATM 상 IP (3/3)
  - 셀 라우팅 (2/2)
    - 주소 결합 (Address Binding)
      - 사용 이유
        - ATM 네트워크에서 셀을 라우팅하기 위해 가상 회선 식별자를 필요로 하지만, IP 데이터그램에는 발신지와 목적지 IP만을 포함하기 때문임
      - 과정
        1. 진입점 라우터는 IP 데이터그램을 수신하며, 진출점 라우터의 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 (48/62)

- 보안 (1/2)
  - 문제점
    - 패킷 스니핑 (Packet Sniffing)
      - 침입자가 패킷을 가로채어 복사본을 만들 수 있음
      - 암호화를 통해 내용을 알아낼 수 없도록 무력화 가능함
    - 패킷 수정 (Packet Modification)
      - 패킷을 가로채고 내용을 고친 후, 새로운 패킷을 수신자에게 보낼 수 있음
      - 무결성 메커니즘을 통해 탐지될 수 있음
    - IP 스푸핑 (IP Spoofing)
      - 다른 컴퓨터의 발신지 주소를 포함한 IP 패킷을 생성할 수 있음
      - 발신지 인증 (Origin Authentication)메커니즘을 통해 방지될 수 있음

IPv4 (49/62)

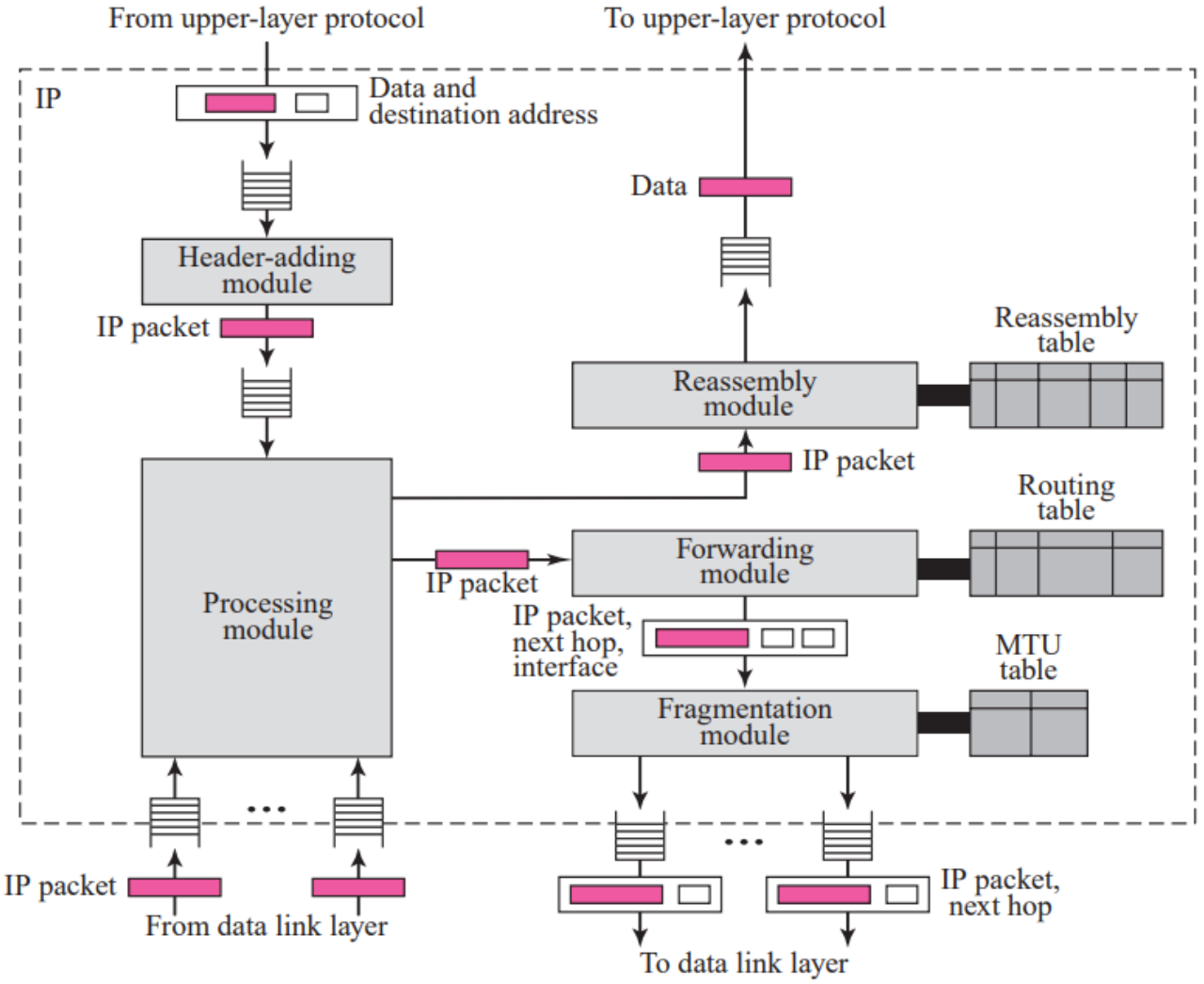
- 보안 (2/2)
  - IPsec (IP Security)
    - 제공 서비스
      - 알고리즘과 키의 결정
        - 안전한 채널을 생성하기 위해 사용 가능한 알고리즘과 키에 대해 합의함.
      - 패킷 암호화
        - 패킷 기밀성을 유지하기 위해 합의된 암호화 알고리즘과 키를 사용하여 암호화를 수행함
        - 패킷 스니핑 공격을 무력화함
      - 데이터 무결성
        - 패킷이 변경되지 않았음을 확인할 수 있도록 함
        - 패킷 수정 공격을 방지함
      - 발신지 인증
        - 발신지를 인증하여 패킷이 공격자에 의해 생성되지 않았음을 확인하도록 함
        - IP 스푸핑 공격을 방지함

IPv4 (50/62)

- IP 패키지 (1/13)
  - 특징
    - 데이터링크 계층이나 상위 계층 프로토콜로부터 IP 패킷을 받음
      - 상위 계층에서 패킷을 받으면 루프백 패킷을 제외하고는 데이터링크 계층으로 보냄
      - 데이터링크 계층으로부터 패킷이 오면, 패킷을 전달하기 위해 다시 데이터링크 계층으로 보내거나, 상위 계층 프로토콜로 보냄

IPv4 (51/62)

- IP 패키지 (2/13)
  - 구성 요소 (1/12)
    - 헤더 추가 모듈
    - 처리 모듈
    - 포워딩 모듈
    - 단편화 모듈
    - 재조립 모듈
    - 라우팅 테이블
    - MTU 테이블
    - 재조립 테이블
    - 큐



IPv4 (52/62)

- IP 패키지 (3/13)
  - 구성 요소 (2/12)
    - 헤더 추가 모듈
      - 동작
        - 상위 계층에서 전달받은 L4 패킷 데이터와 여러 정보들을 통해 L3 헤더를 구축하여 L4 패킷을 IP-Datagram으로 만듦
        - 완성된 IP-Datagram을 목적에 맞는 포워딩 모듈 혹은 재조립 모듈의 Queue에 전달함
      - 의사 코드 (Pseudo code)

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

IPv4 (53/62)

- IP 패키지 (4/13)
  - 구성 요소 (3/12)
    - 처리 모듈 (1/2)
      - 동작
        - 입력 큐에서 데이터그램을 제거함
        - 목적지 주소가 로컬 주소와 일치하는지 확인함
          - 일치하면 재조립 모듈로 데이터그램을 보낸 후 종료함
        - 장치가 라우터인지 확인함
          - 라우터이면 TTL을 감소시킴
        - TTL이 0 이하인지 확인함
          - 0 이하이면 데이터그램을 폐기하고 ICMP 오류 메시지를 전송한 후 종료함
        - 데이터그램을 포워딩 모듈로 보냄

IPv4 (54/62)

- IP 패키지 (5/13)
  - 구성 요소 (4/12)
    - 처리 모듈 (2/2)
      - 의사 코드

```
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 (55/62)

- IP 패키지 (6/13)
  - 구성 요소 (5/12)
    - 큐
      - 입력 큐
        - 데이터링크나 상위 계층 프로토콜로부터 온 데이터그램을 저장함
        - 처리 모듈은 입력 큐로부터 데이터그램을 가져옴
      - 출력 큐
        - 데이터링크 계층이나 상위 계층 프로토콜로 가는 데이터그램을 저장함
        - 단편화 모듈과 재조립 모듈은 출력 큐에 데이터그램을 넣음

IPv4 (56/62)

- IP 패키지 (7/13)
  - 구성 요소 (6/12)
    - 라우팅 테이블
      - 패킷의 다음 홉 수를 결정하기 위해 포워딩 모듈에서 사용됨
    - 포워딩 모듈
      - 처리 모듈로부터 IP 패킷을 받아, 보내져야 할 노드의 주소와 패킷이 보내져야 하는 인터페이스의 번호를 찾고 이 정보와 함께 패킷을 단편화 모듈로 보냄
    - MTU 테이블
      - 단편화 모듈이 특정 인터페이스의 MTU를 찾기 위해 사용됨
      - 인터페이스와 MTU 두 열로 구성됨

IPv4 (57/62)

- IP 패키지 (8/13)
  - 구성 요소 (7/12)
    - 단편화 모듈 (1/2)
      - 동작
        - 데이터그램 크기를 추출함
        - 크기가 해당 네트워크의 MTU보다 큰지 확인함
        - 크기가 클 경우, D 비트가 설정되었으면 데이터그램을 폐기하고 ICMP 오류 메시지를 전송한 후 종료함
        - 크기가 클 경우, D 비트가 설정되지 않았으면 단편화를 수행하고 종료함
        - 크기가 작은 경우, 데이터그램을 전송함

IPv4 (58/62)

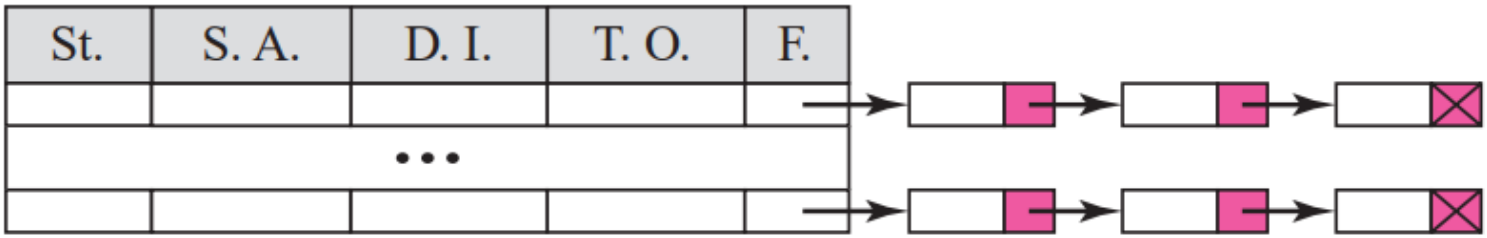
- IP 패키지 (9/13)
  - 구성 요소 (8/12)
    - 단편화 모듈 (2/2)
      - 의사 코드

```
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.
}
```

IPv4 (59/62)

- IP 패키지 (10/13)
  - 구성 요소 (9/12)
    - 재조립 테이블 (1/2)
      - 정의
        - 재조립 모듈에서 쓰이는 시스템 레벨에서 사용되는 자료구조
      - 특징
        - 일반적으로, 연결 리스트 혹은 속도 측면에서 유리한 고정 길이 배열로 구현됨

St.: State
S. A.: Source address
D. I.: Datagram ID
T. O.: Time-out
F.: Fragments



IPv4 (60/62)

- IP 패키지 (11/13)
  - 구성 요소 (10/12)
    - 재조립 테이블 (2/2)
      - 구조
        - State : Entry의 사용 가능 여부를 저장하며, 빈 Entry의 St. 값은 Free임
        - Source address : 소스 노드의 주솟값을 저장함
        - Datagram ID : 단편화 조각 패킷들이 한 패킷에서 파생된 경우, 소스 주소와 더불어 데이터그램 ID또한 동일할 것임
        - Time-out : 첫 번째 단편화 패킷이 도착하면 타이머가 실행되고, 설정한 시간이 넘어가기까지 모든 단편화 조각들을 전달받지 못하면 Time-out이 발생됨
        - Fragments : 단편화 조각들을 지칭하는 포인터로, 단편화 조각들은 연결 리스트에 Offset-Field에 명시된 순서를 고려하여 저장됨

St.: State
S. A.: Source address
D. I.: Datagram ID
T. O.: Time-out
F.: Fragments

IPv4 (61/62)

- IP 패키지 (12/13)
  - 구성 요소 (11/12)
    - 재조립 모듈 (1/2)
      - 동작
        - 오프셋 값이 0이고 M 비트가 0인지 확인함
          - 그렇다면 데이터그램을 적절한 큐로 전송하고 종료함
        - 재조립 테이블에서 해당 항목을 검색함
          - 항목이 없으면 새 항목을 생성함
        - 데이터그램을 연결 리스트에 삽입하고 모든 단편이 도착했는지 확인함
          - 모든 단편이 도착했으면, 단편을 재조립하고 상위 계층 프로토콜로 전달한 후 종료함
          - 단편이 모두 도착하지 않았으면, 타임아웃이 발생했는지 확인하고, 발생했으면 모든 단편을 폐기하고 ICMP 오류 메시지를 전송함

IPv4 (62/62)

- IP 패키지 (13/13)
  - 구성 요소 (12/12)
    - 재조립 모듈 (2/2)
      - 의사 코드

```
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

- 보충
- IPv4
- ARP

ARP (1/33)

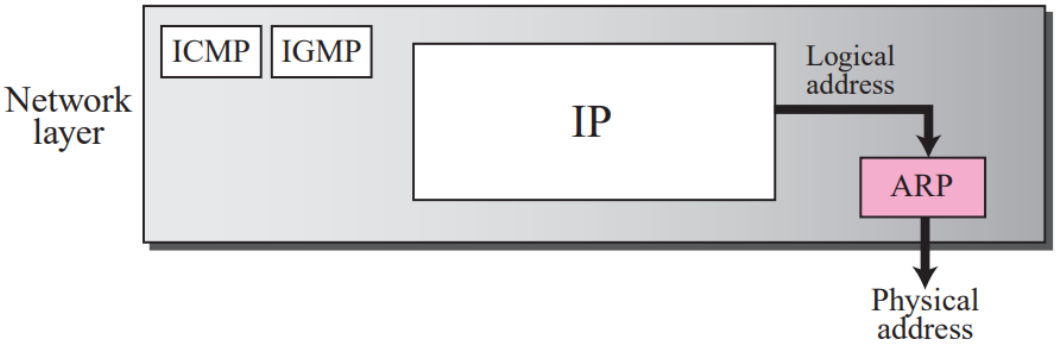
- 주소 변환 (1/3)
  - 필요성
    - 호스트나 라우터로 패킷을 전달하기 위해선, 해당 패킷에 논리 주소와 물리 주소가 모두 저장되어 있어야 함
    - 논리 주소를 물리 주소로, 물리 주소를 논리 주소로 변환하는 작업이 요구됨
  - 종류
    - 정적 변환
    - 동적 변환

ARP (2/33)

- 주소 변환 (2/3)
  - 종류 (1/2)
    - 정적 변환
      - 원리
        - 논리 주소와 물리 주소가 Mapping되어 있는 Table을 이용하여 변환함
        - 이러한 Table은 Network Device들에 저장되어 있음
      - 한계점
        1. Network Device의 NIC(Network Interface Card)를 교체하여 새 물리 주소가 생기는 경우
        2. LocalTalk과 같은 LAN에서와 같이 컴퓨터가 부팅될 때 마다 물리 주소가 변경되는 경우
        3. Mobile 컴퓨터와 같이 네트워크를 이동하면서 물리 주소가 계속 변경되는 경우
        - 위와 같은 한계에 대응하여, 테이블을 주기적으로 업데이트해야하는데, 이러한 업데이트는 네트워크에 매우 큰 오버헤드로 작용하게 됨

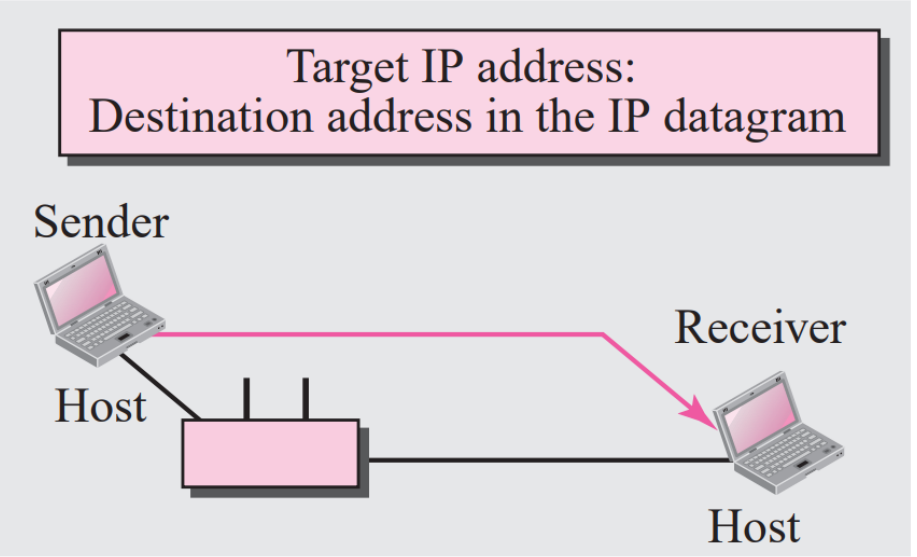
ARP (3/33)

- 주소 변환 (3/3)
  - 종류 (2/2)
    - 동적 변환
      - 원리
        - 한 호스트가 다른 기기의 논리 주소만 알고 있는 경우, 프로토콜을 통해 물리 주소를 알아내는 방식임
      - 종류
        - ARP 프로토콜
          - 논리 주소를 물리 주소로 동적 매핑하는 프로토콜
        - RARP 프로토콜
          - 물리 주소를 논리 주소로 동적 매핑하는 프로토콜
          - 현재 RARP는 DHCP로 대치되어 사용되지 않는 추세임



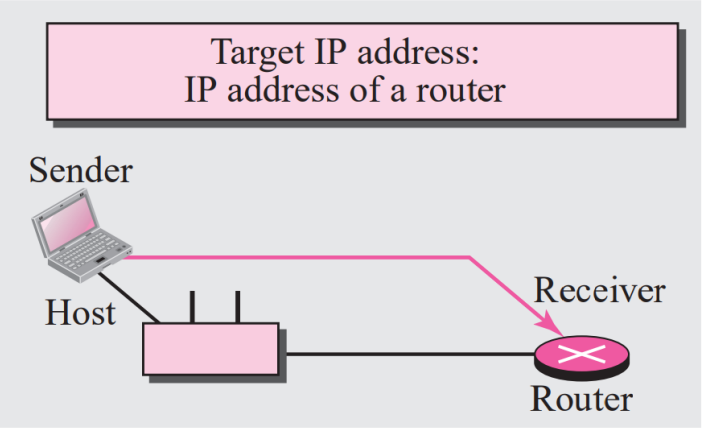
ARP (4/33)

- ARP 프로토콜 (1/11)
  - ARP가 사용되는 경우 (1/4)
    - 한 호스트가 같은 물리 네트워크상의 다른 호스트에게 메시지를 전송하고자 하는 경우
      - 두 호스트는 같은 물리 네트워크상에 있기 때문에 Receiver Host의 논리주소는 이미 알고 있는 상태임
      - 수신자의 물리 주소만 알고있는 상황이므로, ARP 프로토콜이 사용되어야 함



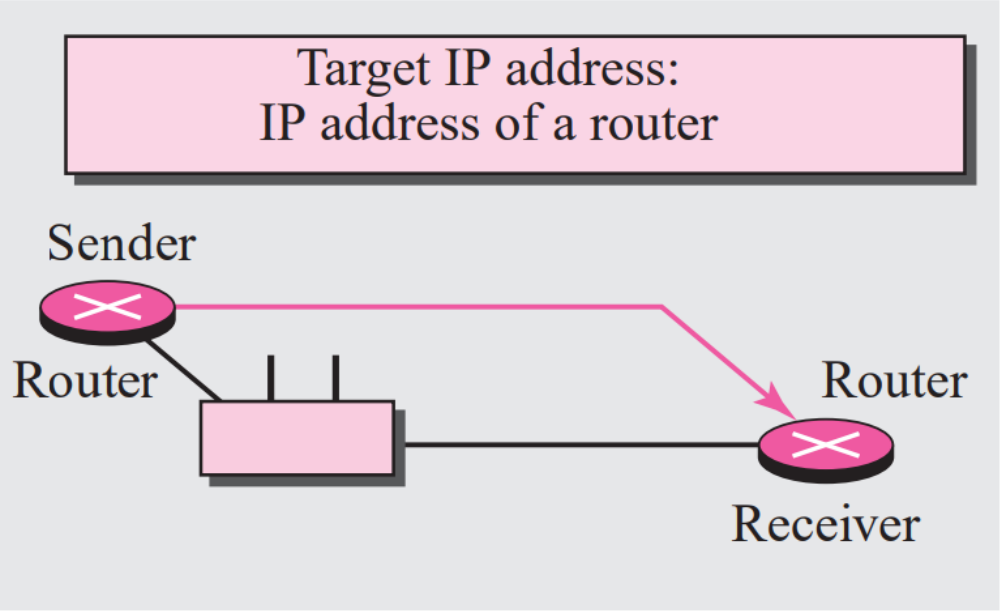
ARP (5/33)

- ARP 프로토콜 (2/11)
  - ARP가 사용되는 경우 (2/4)
    - 한 호스트가 다른 물리 네트워크상의 호스트에게 메시지를 전송하고자 하는 경우
      - 송신자는 자신의 라우팅 테이블(정적 테이블)을 참조하여 인접한 라우터로의 논리주소를 알아내고, 이를 ARP를 통해 물리 주소로 변환함
      - 송신자에게 라우팅 테이블이 없다면, 디폴트 라우터의 논리 주소를 ARP를 이용하여 물리 주소로 변환함
      - 결과적으로 인접한 Router의 논리 주소가 ARP를 통해 물리 주소로 변환됨



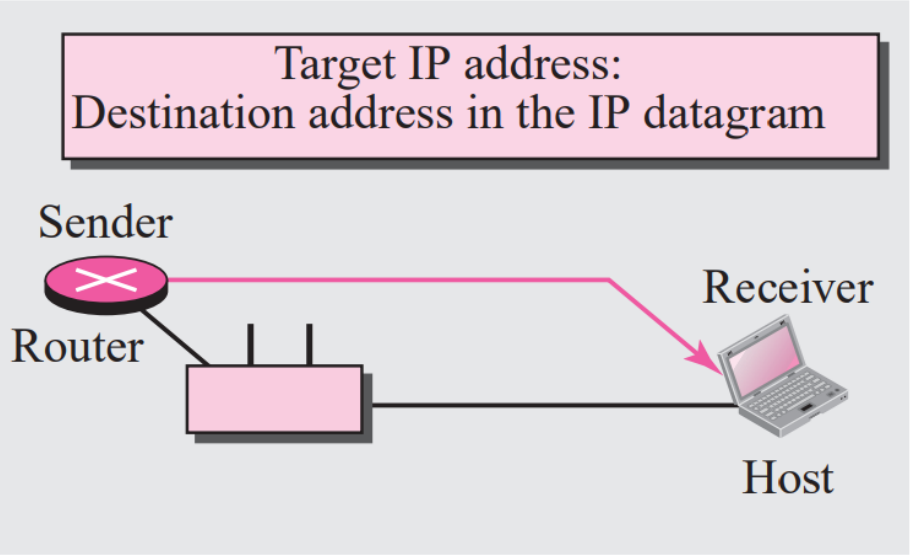
ARP (6/33)

- ARP 프로토콜 (3/11)
  - ARP가 사용되는 경우 (3/4)
    - 한 라우터가 다른 라우터에게로 메시지를 전송하는 경우
      - 메시지가 네트워크 상의 중간경로에 위치한 경우임
      - 송신 라우터는 라우팅 테이블(동적 라우팅 테이블)을 참고하여 수신 라우터의 논리 주소를 알아내고, 이 논리 주소를 ARP를 이용하여 물리 주소로 변환함



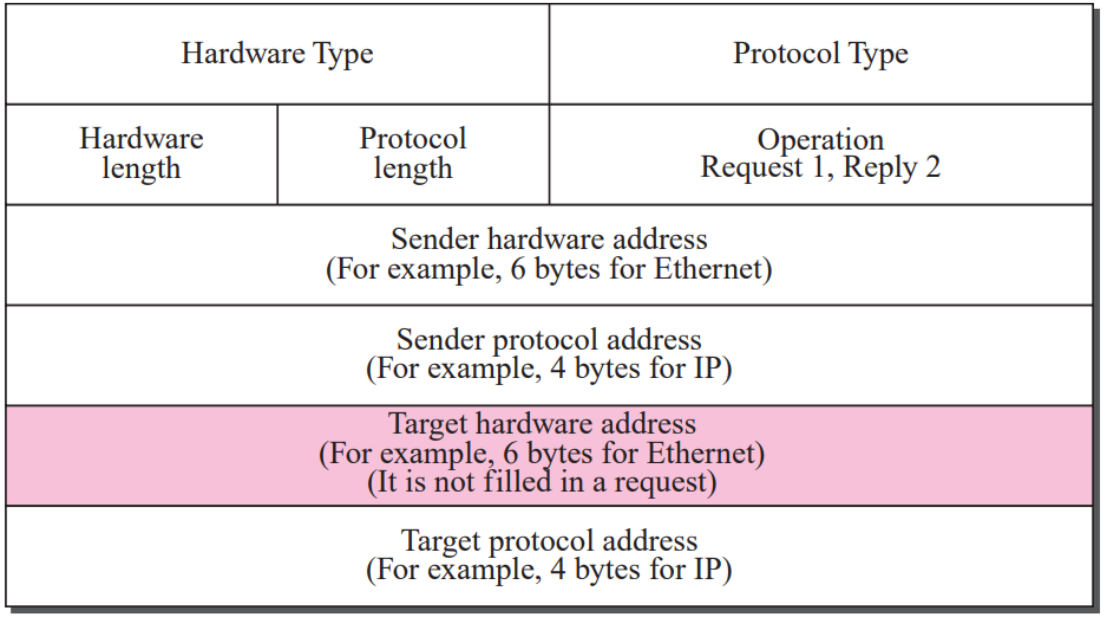
ARP (7/33)

- ARP 프로토콜 (4/11)
  - ARP가 사용되는 경우 (4/4)
    - 한 라우터가 자신의 관할내에 있는 호스트에게로 메시지를 송신하는 경우
      - 메시지가 다수의 물리 네트워크를 거쳐 수신자에게로 도착하는 경우임
      - 수신자의 논리 주소가 ARP를 통해 물리 주소로 변환됨



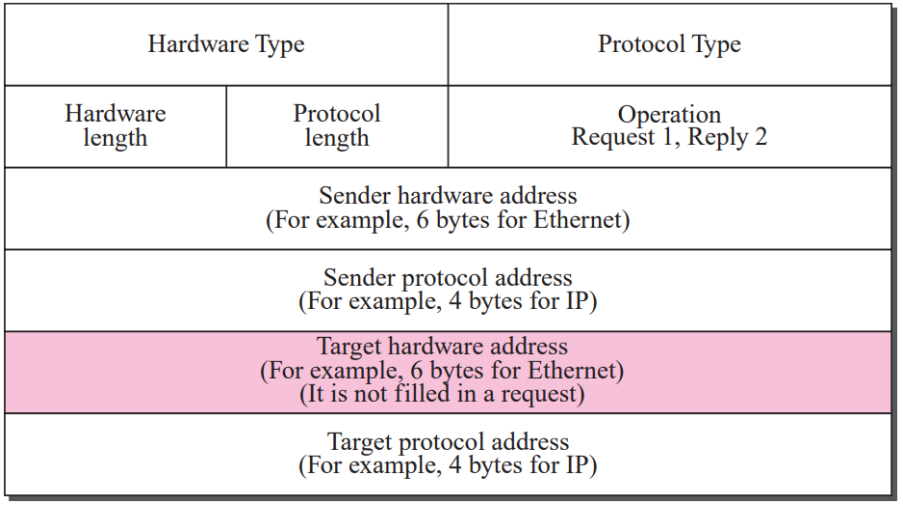
ARP (8/33)

- ARP 프로토콜 (5/11)
  - 패킷 형식 (1/4)
    - 하드웨어 유형
      - ARP가 수행되고 있는 네트워크의 유형을 정의하는 필드임
    - 프로토콜 유형
      - 프로토콜을 정의하는 필드임



ARP (9/33)

- ARP 프로토콜 (6/11)
  - 패킷 형식 (2/4)
    - 하드웨어 길이
      - 물리 주소의 길이를 바이트 단위로 정의하여 저장하는 필드임
      - e.g., Ethernet의 경우, 하드웨어 길이 값은 6임
    - 프로토콜 길이
      - 논리 주소의 길이를 바이트 단위로 정의하여 저장하는 필드임
      - e.g., IPv4의 프로토콜 길이는 4임



ARP (10/33)

- ARP 프로토콜 (7/11)
  - 패킷 형식 (3/4)
    - 동작
      - ARP 요청의 경우, 해당 필드에 1이 저장됨
      - ARP 응답의 경우, 해당 필드에 2가 저장됨
    - 발신지 하드웨어 주소
      - 송신자의 물리 주소가 저장되는 가변 길이 필드임
      - e.g., Ethernet에서 해당 필드는 6바이트임
    - 발신지 프로토콜 주소
      - 송신자의 논리 주소가 저장되는 가변 길이 필드임
      - e.g., IPv4에서 해당 필드는 4바이트임

Hardware Type		Protocol Type
Hardware length	Protocol length	Operation Request 1, Reply 2
Sender hardware address (For example, 6 bytes for Ethernet)		
Sender protocol address (For example, 4 bytes for IP)		
Target hardware address (For example, 6 bytes for Ethernet) (It is not filled in a request)		
Target protocol address (For example, 4 bytes for IP)		

ARP (11/33)

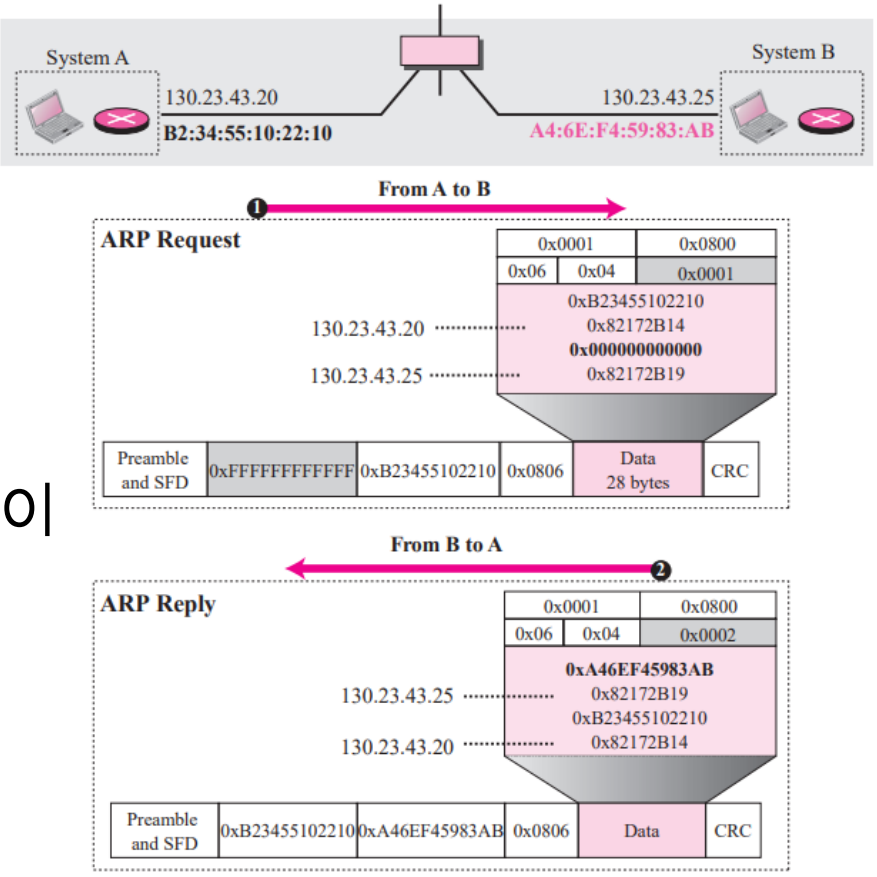
- ARP 프로토콜 (8/11)
  - 패킷 형식 (4/4)
    - 타깃 하드웨어 주소
      - 수신자의 물리 주소가 저장되는 가변 길이 필드임
      - 초기에는 0으로 채워져 있음
    - 타깃 프로토콜 주소
      - 수신자의 논리 주소가 저장되는 가변 길이 필드임

<table>
<tr><td colspan="2">Hardware Type</td><td>Protocol Type</td></tr>
<tr><td>Hardware length</td><td>Protocol length</td><td>Operation
Request 1, Reply 2</td></tr>
<tr><td colspan="3">Sender hardware address
(For example, 6 bytes for Ethernet)</td></tr>
<tr><td colspan="3">Sender protocol address
(For example, 4 bytes for IP)</td></tr>
<tr><td colspan="3">Target hardware address
(For example, 6 bytes for Ethernet)
(It is not filled in a request)</td></tr>
<tr><td colspan="3">Target protocol address
(For example, 4 bytes for IP)</td></tr>
</table>

ARP (12/33)

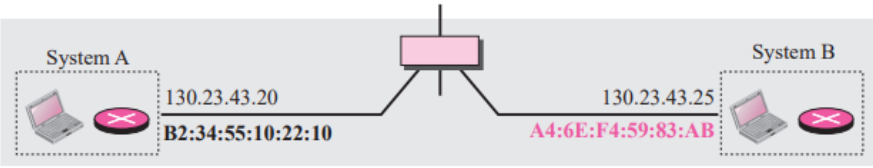
- ARP 프로토콜 (9/11)
  - 전송 예시 (1/2)
    - 가정 사항
      - A에 의한 ARP 요청과 B에 의한 ARP 응답이 수행됨
    - ARP 요청
      - 수신자의 IP는 알고 있으나, 물리 주소는 모르기 때문에 0으로 타깃 하드웨어 주소 필드를 채움
      - 브로트캐스트 방식으로 신호를 전송함
      - ARP 캐시*를 통해 ARP 요청 과정이 생략될 수 있음

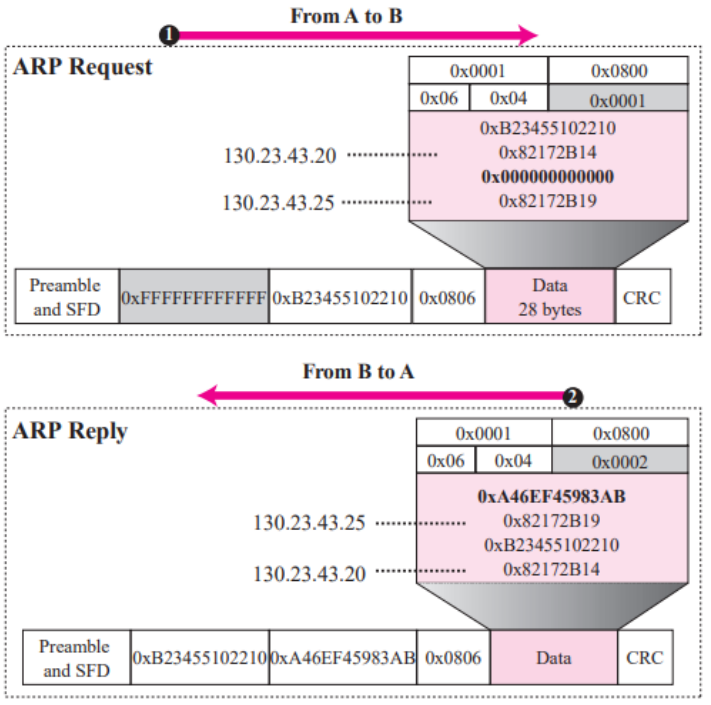
*ARP 캐시(ARP Cache): 기존에 통신했던 호스트의 물리 주소를 저장하는 메모리로, 유효 시간이 존재함



ARP (13/33)

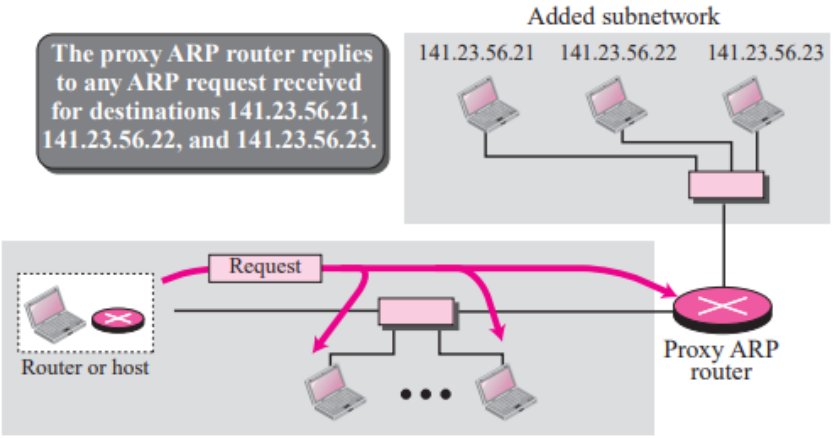
- ARP 프로토콜 (10/11)
  - 전송 예시 (2/2)
    - ARP 응답
      - ARP 요청과 달리, 유니캐스트 방식으로 신호를 전송함
      - 수신자는 송산자의 논리 주소와 물리 주소를 알고 있음





ARP (14/33)

- ARP 프로토콜 (11/11)
  - 프록시 ARP
    - 정의
      - 호스트 집합을 대행해 수행하는 ARP임
    - 목적
      - 서브넷팅 효과를 만들기 위해 사용함
    - 동작 방식
      - 프록시 ARP를 수행하는 라우터가 자신의 호스트 집합으로 향하는 메시지를 수신하면, 라우터는 ARP 응답 패킷에 라우터 자신의 물리 주소를 입력하여 송신자에게 응답함
      - 이후에 자신의 호스트 집합내에 한 호스트에게로 가는 메시지를 라우터가 수신하면, 라우터는 적절히 메시지를 전달함

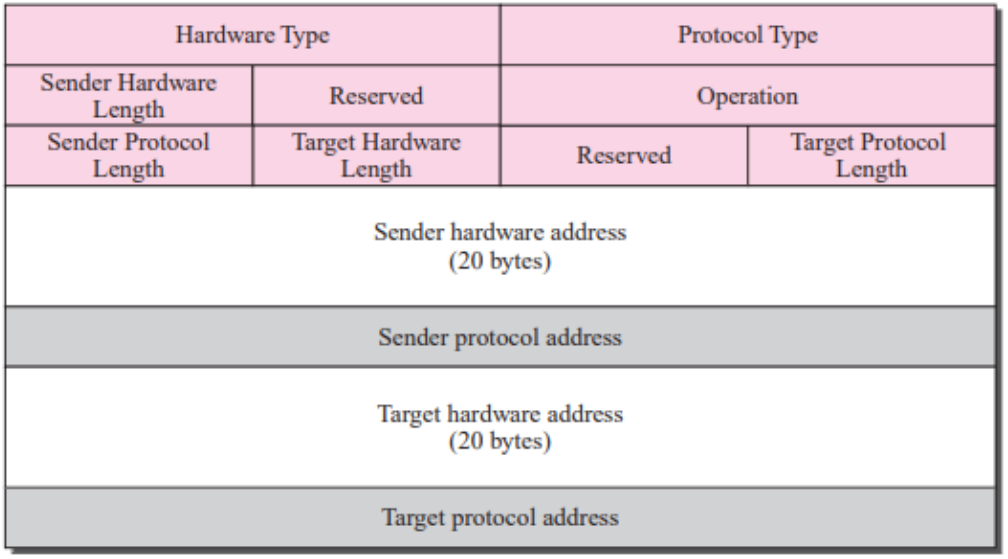


ARP (15/33)

- ATMARP (1/12)
  - 정의
    - ATMWAN 내 진출점 라우터의 물리 주소를 찾기 위해 사용되는 프로토콜
  - LAN 상에서 ARP와의 차이점
    - LAN은 브로드캐스트 네트워크이고 ARP 전송에서 브로드캐스트를 사용함
    - ATM은 브로드캐스트 네트워크가 아니므로, ATMARP를 위해 새로운 해결책이 필요함

ARP (16/33)

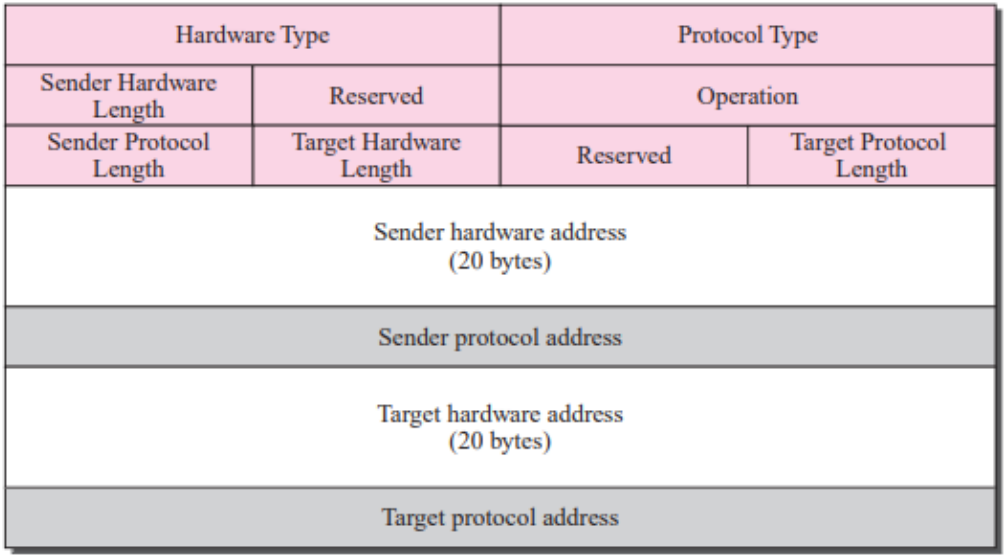
- ATMARP (2/12)
  - 패킷 형식 (1/4)
    - 하드웨어 유형
      - 물리적 네트워크 유형을 정의함
      - e.g., ATM 네트워크의 경우 0x13을 가짐
    - 프로토콜 유형
      - 프로토콜 유형을 정의함



ARP (17/33)

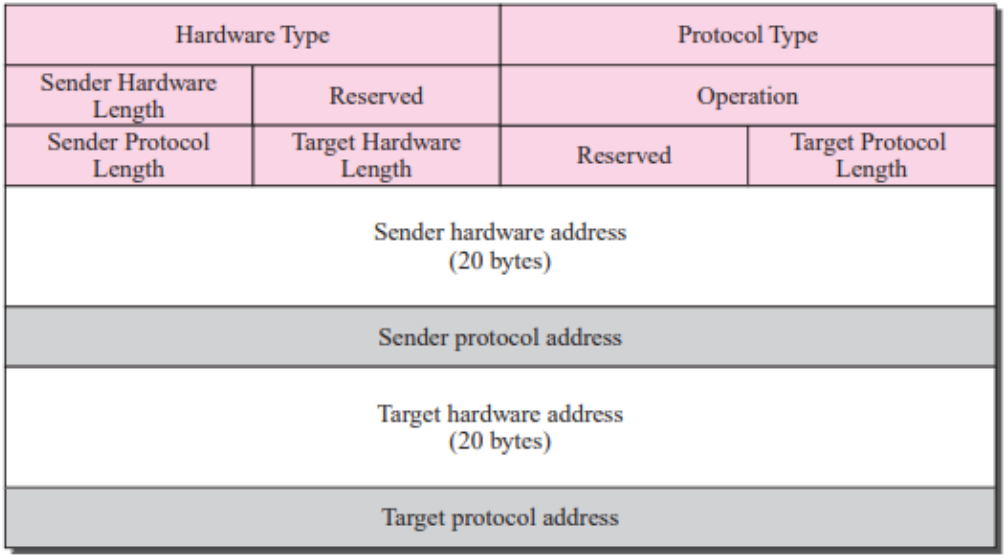
- ATMARP (3/12)
  - 패킷 형식 (2/4)
    - 송신자 하드웨어 길이
      - 송신자 물리 주소의 길이를 바이트 단위로 정의함
      - e.g., ATM 네트워크의 경우 20의 값을 가짐
    - 동작
      - 패킷의 유형을 정의함

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



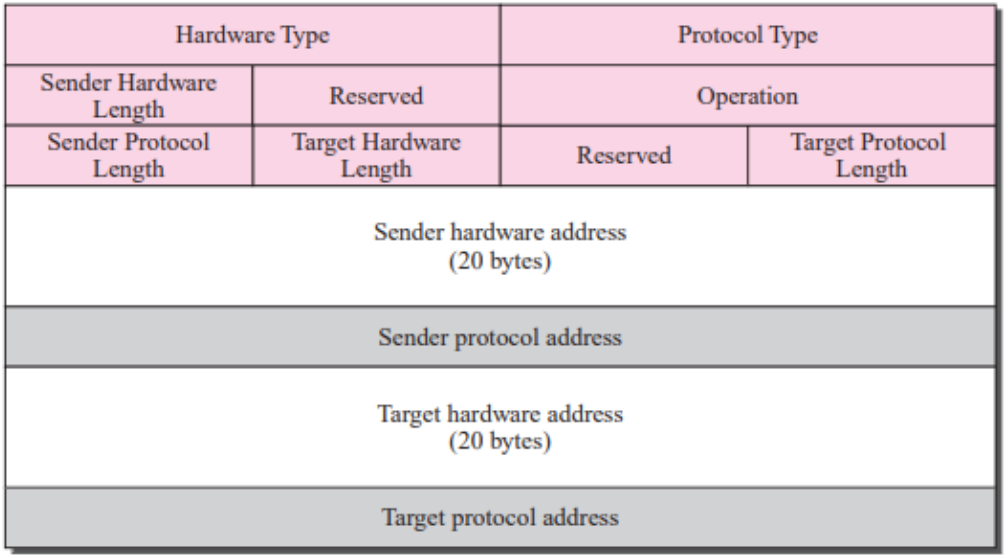
ARP (18/33)

- ATMARP (4/12)
  - 패킷 형식 (3/4)
    - 송신자 프로토콜 길이
      - 송신자 프로토콜 주소의 길이를 바이트 단위로 정의함
    - 타깃 하드웨어 길이
      - 수신자 물리 주소의 길이를 바이트 단위로 정의함
      - ATM 네트워크를 가로질러 결합이 발생하고, 두 개의 하드웨어 주소가 필요하면 reserved 필드가 두 번째 주소의 길이를 정의하기 위해 사용됨
    - 타깃 프로토콜 길이
      - 수신자 프로토콜 주소의 길이를 바이트 단위로 정의함



ARP (19/33)

- ATMARP (5/12)
  - 패킷 형식 (4/4)
    - 송신자 하드웨어 주소
      - 송신자의 물리 주소를 정의하는 가변 길이 필드
    - 송신자 프로토콜 주소
      - 송신자 프로토콜 주소를 정의하는 가변 길이 필드
    - 타깃 하드웨어 주소
      - 수신자 물리 주소를 정의하는 가변 길이 필드
      - 요청 메시지에서는 비어 있고, 응답과 NACK 메시지에서는 채워짐
    - 타깃 프로토콜 주소
      - 수신자의 프로토콜 주소를 정의하는 가변 길이 필드

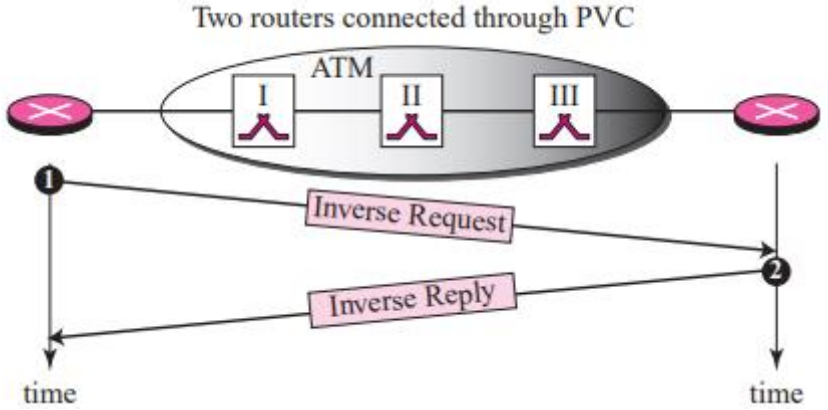


ARP (20/33)

- ATMARP (6/12)
  - 동작 방식 (1/6)
    - PVC 연결 (1/2)
      - 정의
        - 미리 설정된 고정된 경로를 따라 데이터를 전송하는 회선 연결 방식
      - 특징
        - VPI와 VCI가 영구적인 연결을 위해 정의되고, 이 값들이 교환기의 테이블에 입력됨
        - ATMARP 서버가 필요하지 않지만, 라우터가 물리 주소를 IP와 결합할 수 있어야 함
      - PVC가 라우터에서 설정될 때, 라우터는 인버스 요청 메시지를 송신함
      - 수신 라우터는 메시지를 수신하고 인버스 응답을 되돌려 보냄
        - 인버스 요청/응답에는 메시지 송신자의 물리 주소와, IP 주소가 포함됨
      - 인버스 메시지 교환 후, 양 라우터들은 물리 주소를 PVC에 매핑하는 테이블 항목을 추가함
      - 라우터가 IP 데이터그램을 수신하면, 테이블은 라우터가 가상 회선 식별자를 사용하여 데이터그램을 캡슐화할 수 있는 정보를 제공함

ARP (21/33)

- ATMARP (7/12)
  - 동작 방식 (2/6)
    - PVC 연결 (2/2)
      - 절차
        1. PVC가 라우터에서 설정될 때, 라우터는 인버스 요청 메시지를 송신함
        2. 수신 라우터는 메시지를 수신하고 인버스 응답을 되돌려 보냄
           - 인버스 요청/응답에는 메시지 송신자의 물리 주소와, IP 주소가 포함됨
        3. 인버스 메시지 교환 후, 양 라우터들은 물리 주소를 PVC에 매핑하는 테이블 항목을 추가함
        4. 라우터가 IP 데이터그램을 수신하면, 테이블은 라우터가 가상 회선 식별자를 사용하여 데이터그램을 캡슐화할 수 있는 정보를 제공함

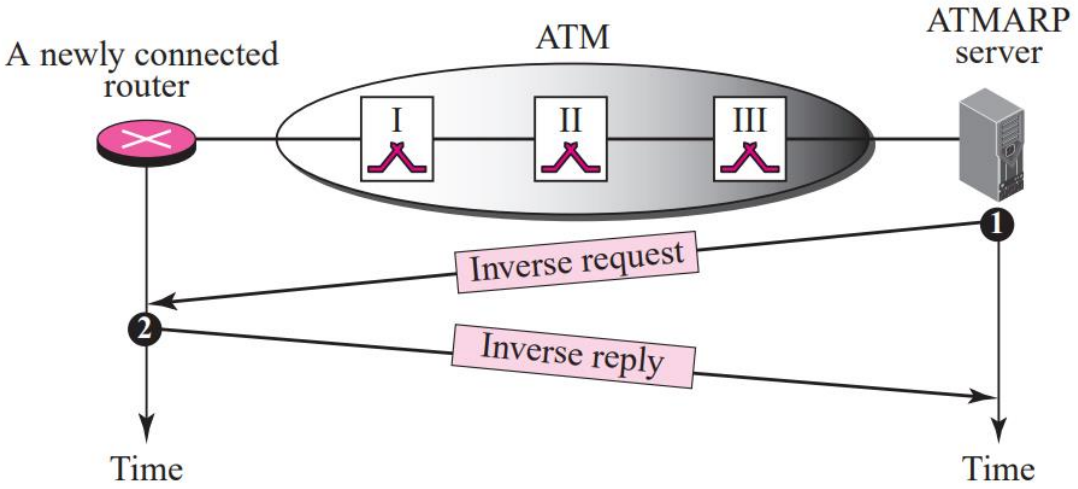


ARP (22/33)

- ATMARP (8/12)
  - 동작 방식 (3/6)
    - SVC 연결 (1/4)
      - 정의
        - 데이터 전송이 필요할 때마다 설정되고 종료되는 동적 연결 방식
      - 특징
        - 한 라우터가 다른 라우터와 연결 생성을 원할 때마다 새로운 가상 회선이 설립됨
        - 각 라우터는 클라이언트 ATMARP 프로그램을 수행하고, 하나의 컴퓨터가 ATMARP 서버 프로그램을 수행함

ARP (23/33)

- ATMARP (9/12)
  - 동작 방식 (4/6)
    - SVC 연결 (2/4)
      - 초기 설정
        - ATM 서버 테이블 만들기
          - 라우터가 ATM 네트워크에 처음 연결되고 PVC 연결이 라우터와 서버 사이에 설립되면, 서버는 라우터에게 인버스 요청을 보냄
          - 라우터는 자신의 IP 주소와 물리 주소를 포함하여 인버스 응답을 수행함
          - 서버는 라우팅 테이블에 해당 라우터가 미래에 진출점 라우터로 수행되는 경우에 사용할 항목을 추가함



ARP (24/33)

- ATMARP (10/12)
  - 동작 방식 (5/6)
    - SVC 연결 (3/4)
      - 절차

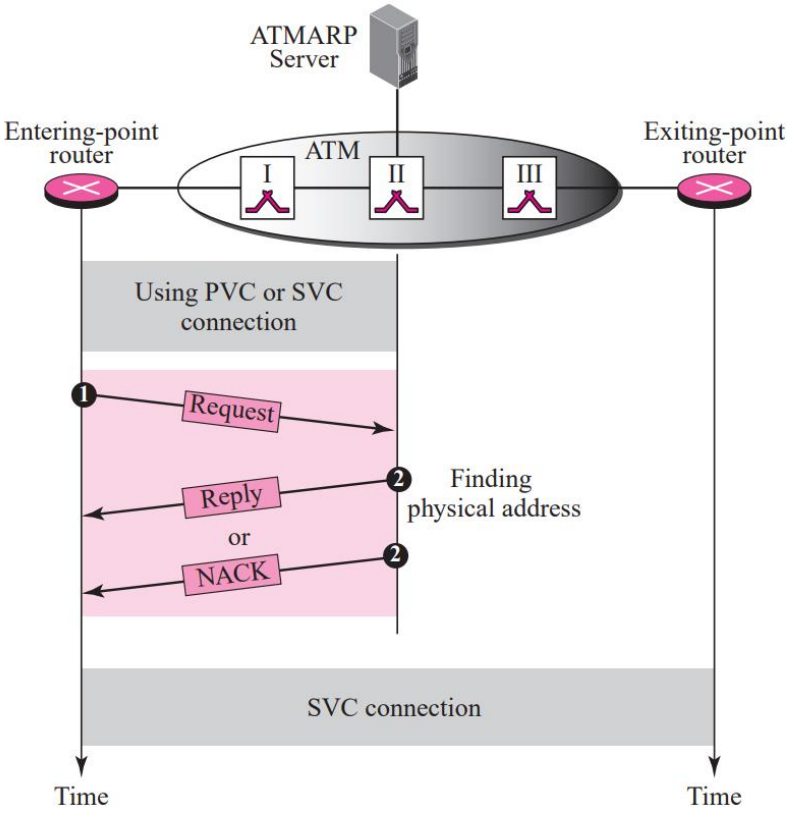
1. 서버 연결
   - 라우터와 서버 사이에 연결된 PVC를 통해 서버에 연결함
   - 라우터와 서버 사이에 PVC가 없으면, 서버는 최소한 ATMARP 요청 및 응답 메시지 교환을 위한 SVC 연결을 생성하기 위해 라우터의 물리 주소를 알아야 함

2. 물리 주소 받기
   - 진입점 라우터와 서버 사이에 연결이 있을 때, 라우터는 ATMARP 요청을 서버로 보냄
   - 서버는 물리 주소를 찾을 수 있으면 ATMARP 응답을, 그렇지 않으면 ATMARP NACK 응답을 수행함
   - 진입점 라우터가 NACK를 받으면 데이터그램은 폐기됨

ARP (25/33)

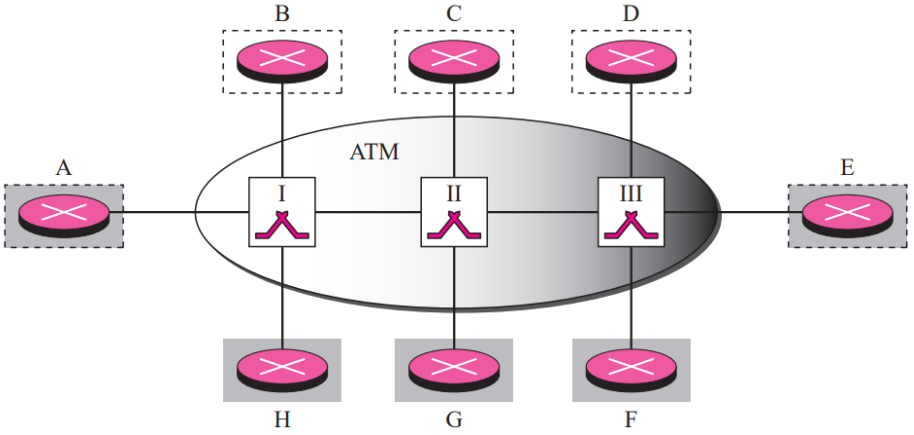
- ATMARP (11/12)
  - 동작 방식 (6/6)
    - SVC 연결 (4/4)
      - 절차

3. 가상 회선 설립
   - 진출점 라우터의 물리 주소를 받은 후, 진입점 라우터는 자신과 진출점 라우터간의 SVC를 요청함
   - ATM 네트워크는 진입점 라우터가 해제를 요구할 때까지 유지되는 가상 회선을 설립하기 위해 두 개의 물리주소를 사용함
   - 네트워크 내의 각 교환기는 IP 데이터그램을 운반하는 셀들이 라우팅되도록 테이블에 항목을 추가함



ARP (26/33)

- ATMARP (12/12)
  - 논리적 IP 서브넷 (LIS, Logical IP Subnet)
    - 정의
      - ATM 네트워크를 여러 개의 논리적 서브넷으로 나눌 수 있게 하는 기술
    - 특징
      - ATM 네트워크에서 브로드캐스트 기능을 모방하는 것이 필요한 ATMARP와 다른 프로토콜들이 운용되도록 만들어줌
      - 동일한 논리적 서브넷에 속한 라우터는 동일한 프리픽스와 서브넷 마스크를 공유함
      - 다른 서브넷에 속한 라우터에 패킷을 송신해야 하면, 두 서브넷에 모두 속한 라우터로 패킷을 보내야 함
      - ATMARP를 사용하기 위해 각 서브넷에는 각각의 ATMARP 서버가 필요함



ARP (27/33)

- ARP 패키지 (1/7)
  - 수행
    - 3계층으로부터 IP 데이터그램을 넘겨받아 수신지 물리 주소를 알아내어 2계층의 프레임으로 캡슐화를 수행함
  - 구성요소
    - 캐쉬 테이블
    - 큐
    - 출력 모듈
    - 입력 모듈
    - 캐쉬 제어 모듈

ARP (28/33)

- ARP 패키지 (2/7)
  - 캐쉬 테이블
    - 엔트리의 배열로 구성되며, 각 엔트리는 다음의 필드를 가짐

Cache Table Entry	Description
State	FREE, PENDING, RESOLVED 셋 중 하나로 지정됨 FREE값은 해당 엔트리의 TTL이 만료되었으므로, 다른 Entry가 입력될 수 있음을 의미함 PENDING값은 해당 엔트리에 대한 ARP 요청이 전송되었으나, ARP 응답을 아직 수신하지 못했음을 의미함 RESOLVED값은 해당 엔트리가 ARP 응답까지 수신하여 물리 주소까지 알고 있고 캐쉬 정보가 만료되지 않은 상태임을 의미함 (송신을 준비중인 패킷은 캐쉬 테이블에서 RESOLVED 상태인 엔트리에 대한 정보만 활용할 수 있음)
Hardware Type	ARP 패킷에서의 하드웨어 유형과 같음
Protocol Type	ARP 패킷에서의 프로토콜 유형과 같음
Hardware Length	ARP 패킷에서의 하드웨어 길이와 같음
Protocol Length	ARP 패킷에서의 프로토콜 길이와 같음
Interface Number	다수의 인터페이스를 가진 장비에서는 인터페이스의 번호가 지정되어야 함 장치의 주소는 인터페이스마다 부여됨
Queue Number	ARP 응답을 기다리는 패킷들이 저장되는 Queue 보통, 같은 목적지 주소를 가진 패킷들은 같은 큐에 저장됨
Attempts	ARP 요청을 전송한 횟수
Time-Out	해당 엔트리의 TTL을 초단위로 환산한 값
Hardware Address	수신자의 물리 주소 값 ARP Reply를 받아야만 채워짐
Protocol Address	수신자의 논리 주소 값

ARP (29/33)

- ARP 패키지 (3/7)
  - 큐
    - 정의
      - ARP 응답을 기다리는 IP 데이터그램들이 저장되는 자료구조
    - 특징
      - Destination마다 하나씩 유지됨
      - 출력 모듈에서 해결되지 않은 패킷들이 큐에 들어옴
      - 입력 모듈은 큐에서 패킷을 해결된 물리 주소와 같이 가져와 데이터링크 계층으로 보내어 전달될 수 있도록 함

ARP (30/33)

- ARP 패키지 (4/7)
  - 출력 모듈 (1/2)
    - 동작
      - IP 패킷이 수신될 때까지 대기함
      - 캐시 테이블에서 IP 패킷의 목적지에 해당하는 항목을 찾음
        - 항목이 발견된 경우
          - 상태가 RESOLVED라면 하드웨어 주소를 추출하여 데이터 링크 계층으로 패킷과 함께 전송함
          - 상태가 PENDING이라면 패킷을 큐에 삽입함
        - 항목이 발견되지 않은 경우
          - 상태가 PENDING으로 설정된 캐시 항목을 생성하고 시도 횟수(ATTEMPTS)를 1로 설정함
          - 큐를 생성하고 패킷을 삽입함
          - ARP 요청을 전송함

ARP (31/33)

- ARP 패키지 (5/7)
  - 출력 모듈 (2/2)
    - 의사 코드

```
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 (32/33)

- ARP 패키지 (6/7)
  - 입력 모듈
    - 동작 및 의사코드
      - ARP 패킷(요청 또는 응답)이 도착할 때까지 대기함
      - 해당하는 항목을 찾기 위해 캐시 테이블을 검사함
        - 항목이 발견된 경우
          - 항목을 업데이트함
          - 상태가 PENDING이라면 큐가 비어있지 않은 동안 패킷을 하나씩 꺼내서 하드웨어 주소와 함께 전송함
        - 항목이 발견되지 않은 경우
          - 새 항목을 생성하고 테이블에 추가함
        - 패킷이 요청인 경우
          - ARP 응답을 전송함

```
1  ARP_Input_Module ( )
2  {
3      Sleep until an ARP packet (request or reply) arrives.
4      Check the cache table to find the corresponding entry.
5
6      If (found)
7      {
8          Update the entry.
9
10         If (the state is PENDING)
11         {
12             While (the queue is not empty)
13             {
14                 Dequeue one packet.
15                 Send the packet and the hardware address.
16             }//end while
17         }//end if
18
19     }//end if
20
21     If (not found)
22     {
23         Create an entry.
24         Add the entry to the table.
25     }//end if
26
27     If (the packet is a request)
28     {
29         Send an ARP reply.
30     }//end if
31
32     Return
33
34 }//end module
```

ARP (33/33)

- ARP 패키지 (7/7)
  - 캐쉬 제어 모듈
    - 동작 및 의사코드
      - 주기적 타이머가 만료될 때까지 대기함
      - 캐시 테이블의 모든 항목에 대해 반복함
        - 상태가 FREE인 경우
          - 다음 항목으로 계속 진행함
        - 상태가 PENDING인 경우
          - 시도 횟수를 1 증가시킴
          - 시도 횟수가 최대값을 초과하면 상태를 FREE로 변경하고 해당 큐를 삭제함
          - 그렇지 않으면 ARP 요청을 보냄
          - 다음 항목으로 계속 진행함
        - 상태가 RESOLVED인 경우
          - 타임아웃 값을 감소시킴
          - 타임아웃이 0 이하로 떨어지면 상태를 FREE로 변경하고 해당 큐를 삭제함

```
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 an 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
```

Thanks!

이 정 민(jeongmin@pel.sejong.ac.kr)