IT CookBook, 쉽게 배우는 데이터 통신과 컴퓨터 네트워크(개정) 7장 연습문제 해답

1. ① 라우팅, ② 혼잡 제어, ③ 분할, ④ 병합
2. ① 정적 라우팅, ② 동적 라우팅
3. ① HELLO, ② ECHO
4. ① 소스 라우팅, ② 분산 라우팅, ③ 중앙 라우팅, ④ 계층 라우팅
5. ① 혼잡, ② 흐름
6. ① 트래픽 성형, ② 리키 버킷
7. 자원 예약
8. ① ECN
9. 홉
10. ① 거리 벡터, ② 링크 상태, ③ 경로 벡터
11. IP
12. Time To Live
13. DHCP

14. ②, ④

(설명②) 네트워크에 패킷 수가 과도하게 증가하는 현상을 혼잡이라 하고, 혼잡 현상을 예방하거나 제거하는 기능을 혼잡 제어라 한다.

(설명④) 송수신 호스트 사이의 패킷 전달 경로를 선택하는 과정을 라우팅이라 한다.

15. ①, ③, ④, ⑤

(설명②) 정적 라우팅 방식은 라우터에 보관된 경로 정보가 고정되어 변화된 정보를 갱신하기가 용이하지 않으며, 특히 네트워크 내부의 혼잡도를 반영할 수 없다.

16. ①, ②, ④, ⑤

(설명③) '목적지 호스트'에는 패킷의 최종 목적지가 되는 호스트의 주소 값을 지정

한다.

17. ②, ③

(설명②) 전송 중인 패킷이 버려지면 송신 호스트는 타임아웃 동작을 통해 패킷을 재전송하므로 네트워크로 유입되는 패킷의 양이 늘어나 혼잡의 정도가 증가된다.

(설명③) 패킷의 전송 지연 시간이 송신 호스트가 설정한 타임아웃 시간보다 크면 재전송 과정이 증가되어 혼잡의 정도가 증가된다.

18. ①, ②, ③, ④

(설명⑤) 자원 예약 방식은 혼잡 문제에는 도움이 되지만 통신 자원의 낭비를 초래할 수 있다.

19. ①, ③, ④

(설명②) 플러딩은 라우터가 자신에 입력된 패킷을 출력 가능한 모든 경로로 중개하는 방식이다.

(설명⑤) 외부 라우팅 프로토콜인 경로 벡터 프로토콜은 단순히 해당 라우터에서 어느 네트워크가 연결 가능한지에 대한 정보만을 제공한다.

20. ③, ④

(설명③) RIP 프로토콜은 거리 벡터 방식을 사용하는 내부 라우팅 프로토콜의 하나이다.

(설명④) RIP 프로토콜은 소규모 네트워크 환경에 적합하며, 가장 많이 사용되는 라우팅 프로토콜 중 하나다.

21. ①, ②, ⑤

(설명③) 오류 제어를 위하여 헤더 체크섬만 제공하고, 데이터 체크섬은 제공하지 않는다. 데이터 부분은 TCP 프로토콜에서 체크섬 기능을 제공하기 때문이다.

(설명④) Best Effort 원칙에 따른 전송 기능을 제공하기 때문에 물리적인 전송 오류를 100% 복구하지 못한다.

22. ②, ⑤

(설명②) Identification 필드는 송신 호스트가 지정하는 패킷 구분자로 분할된 패킷에 동일한 번호를 부여함으로써, 수신 호스트가 Identification 번호가 같은 패킷을 다시 병합할 수 있도록 해준다.

(설명⑤) 분할된 패킷들은 일정한 크기로 나누어지므로 모두 동일한 크기를 갖지만, 제일 마지막 패킷은 그렇지 않을 수 있다.

23. ④

(설명④) 상위 계층에서 내려온 데이터를 계층 2의 프레임 틀에 담을 수 있도록 IP 프로토콜에서 분할 과정을 거친 후에 전송하고, 수신 측에서는 이를 다시 합치는 작업을 수행해야 한다.

24. ①, ②, ③, ④, ⑤

25.

네트워크 계층(Network Layer)의 기본 기능은 송수신 호스트 사이의 패킷 전달 경로를 선택하는 라우팅이다. 라우팅 과정에 수반되는 부분도 네트워크 계층에서 처리하는데, 대표적인 것이 네트워크의 특정 지역에 트래픽이 몰리는 현상을 다루는 혼잡 제어와 라우터 사이의 패킷 중개 과정에서 다루는 패킷의 분할과 병합이다.

네트워크 계층에서 제공하는 주요 기능을 정리하면 다음과 같다.

■ 라우팅

네트워크의 구성 형태에 대한 정보는 라우팅 테이블(Routing Table)이라는 기억 장소에 보관된다. 그리고 이 정보를 이용해 패킷이 목적지까지 도달하기 위한 경로를 선택한다. 송수신 호스트 사이의 패킷 전달 경로를 선택하는 과정을 라우팅(Routing)이라 하고, 라우팅 테이블 정보는 네트워크 관리자나 네트워크 자신의 판단에 의해 계속 변경될 수 있다.

■ 혼잡 제어

네트워크에 패킷 수가 과도하게 증가하는 현상을 혼잡(Congestion)이라 하고, 혼잡 현상을 예방하거나 제거하는 기능을 혼잡 제어(Congestion Control)라 한다. 혼잡이 발생하면 네트워크 전체의 전송 속도가 급격히 떨어지므로 혼잡이 발생하지 않도록 관리해야 한다. 특히 네트워크의 특정 지역에서 혼잡이 발생하면, 그 혼잡이 주위로 빠르게 확산될 가능성이 높다.

■ 패킷의 분할과 병합

상위의 전송 계층에서 송신을 요구한 데이터는 최종적으로 MAC 계층의 프레임 구조에

정의된 형식으로 캡슐화되어 전송된다. 따라서 전송 계층에서 보낸 데이터가 너무 크면 패킷을 여러 개로 작게 쪼개 전송해야 한다. 큰 패킷 하나를 작게 나누는 과정을 패킷 분할(Segmentation)이라 하고, 반대로 목적지에서 분할된 패킷을 다시 모으는 과정을 병합(Reassembly)이라 한다. 패킷의 분할과 병합 과정이 양 끝단 시스템 사이에서 이루어지는 경우에는 전송 계층이 동작하는 종단 수신 시스템에 분할된 패킷을 보내고, 네트워크 계층에서 이루어지는 경우에는 중간의 각 라우터에서 분할과 병합을 반복한다.

26.

라우터 초기화 과정에서 가장 먼저 할 일은 주변 라우터의 경로 정보를 파악하는 것이다. 각 라우터가 주변에 연결된 라우터에 초기화하기 위해 사용하는 것이 HELLO 패킷이다. 라우터 사이의 전송 지연 시간을 측정하기 위해서 ECHO 패킷을 전송하는데, ECHO 패킷을 수신한 호스트는 송신 호스트에 즉각 회신하도록 설계되어 있다. 이런 과정을 반복하고, 측정값의 평균을 구해 해당 라우터까지의 전송 지연 시간을 유추할 수 있다.

27.

네트워크에 존재하는 전송 패킷 수가 많아질수록 네트워크 성능은 자연스럽게 감소한다. 이와 같은 성능 감소 현상이 급격하게 악화되는 현상을 혼잡(Congestion)이라고 하고, 혼잡 문제를 해결하기 위한 방안을 혼잡 제어(Congestion Control)라 한다.

송신호스트가 수신 호스트에서 처리할 수 있는 양보다 많은 데이터를 보내면 수신 호스트는 데이터를 정상적으로 처리하지 못한다. 이와 같이 송신 호스트와 수신 호스트 사이의 점대점 전송 속도를 조절하는 것을 흐름 제어(Flow Control)라 한다.

흐름 제어가 송신 호스트와 수신 호스트 사이의 문제를 다룬다면 혼잡 제어는 더 넓은 관점에서 호스트와 라우터를 포함한 서브넷에서 네트워크 전송능력 문제를 다룬다. 다음 그림은 흐름 제어와 혼잡 제어의 역할 차이를 설명한다.

그림 7-3 **흐름 제어와 혼잡 제어**

28.

혼잡이 발생하는 원인은 다양한데, 기본적으로 네트워크의 처리 능력보다 과도하게 많은 패킷이 입력되면 발생한다. 개별 라우터 관점에서 보면 라우터의 출력 선로를 통한 전송 용량이 부족해 아직 전송하지 못한 패킷이 버퍼에 저장되고, 입력 선로로 들어오는 패킷이 늘면서 버퍼 용량은 더 부족해진다. 결과적으로 라우터의 내부 버퍼 용량 부족이 심화되어 더 이상 패킷을 보관할 수 없어 버리게 된다. 그리고 전송 패킷이 버려지면 송신 호스트는 타임아웃(Timeout) 동작을 통해 패킷을 재전송하므로 네트워크로 송신되는 패킷의 양이 늘어난다.

29.

혼잡은 트래픽이 특정 시간에 집중되는 버스트(Burst) 현상에서 기인하는 경우가 많다. 즉, 송신 호스트에서 전송하는 패킷의 양이 시간대별로 일정하게 발생하는 경우보다 패킷이 짧은 시간에 많이 발생하는 경우에 혼잡이 일어날 확률이 높다. 따라서 송신 호스트가 전송하는 패킷의 발생 빈도가 네트워크에서 예측할 수 있는 전송률로 이루어지게 하는 기능이 필요한데, 이를 트래픽 성형(Traffic Shaping)이라 한다.

30.

초크(Choke) 패킷은 혼잡을 제거할 때 사용한다. 라우터는 자신의 출력 선로를 사용하는 빈도를 모니터할 수 있으므로 출력 선로의 사용정도가 한계치를 초과하면 주의 표시를 해둔다. 그리고 주의 표시한 방향의 경로는 혼잡이 발생할 가능성이 높기 때문에 특별 관리한다.

31.

거리 벡터(Distance Vector) 프로토콜은 직접 연결된 라우터 간에 라우팅 정보를 교환하는 방식이다. 정보를 교환하는 라우터는 거리 벡터 프로토콜을 사용하는 호스트나 라우터들이 다. 교환 정보는 각각의 라우터에서 전체 네트워크에 속하는 개별 네트워크까지 패킷을 전송하는데 걸리는 거리 정보다. 거리 벡터 알고리즘을 구현하려면 개별 라우터가 링크 벡터, 거리 벡터, 다음 홉 벡터라는 세 가지 필수 정보를 관리해야 한다.

■ 링크 벡터

링크 벡터 L(x)는 라우터x와 직접 연결된 주변 네트워크에 대한 연결 정보를 보관한다. 라우터 x와 직접 연결된 네트워크가M개일 때 링크 벡터 정보는 다음과 같이 나타낸다. 링크 벡터에 보관된 정보는 라우터x가 해당 네트워크와 연결하기 위해 할당한 라우터 포트 번호라고 생각할 수 있다.

링크 벡터 L(x) = [포트(1), 포트(2), ..., 포트(m), ..., 포트(M)]

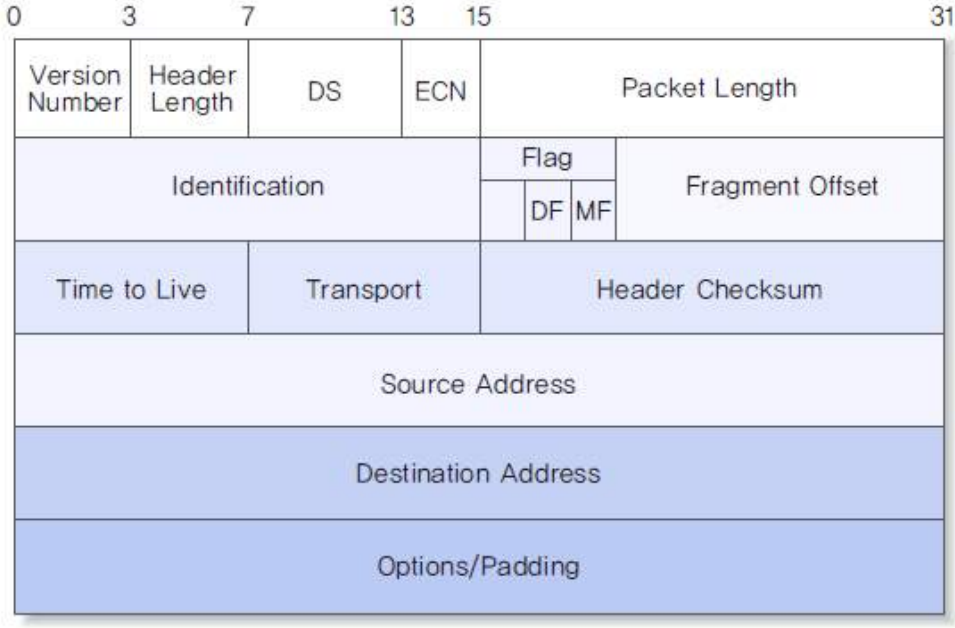
■ 거리 벡터

거리 벡터 D(x)는 전체 네트워크에 포함된 개별 네트워크들까지의 거리 정보를 관리한다. 네트워크가 N개라고 가정하면 거리 벡터 정보는 다음과 같이 표시할 수 있다. 거리 벡터에서 관리하는 거리 정보는 일반적으로 개별 네트워크까지 패킷을 전송하는 데 걸리는 최소 전송 지연 시간이다.
거리 벡터D(x) = [거리(1), 거리(2), ..., 거리(n), ..., 거리(N)]

■ 다음 홉 벡터
다음 홉 벡터 H(x)는 개별 네트워크까지 패킷을 전송하는 경로에 있는 다음 홉 정보를 관리한다. 보관하는 정보 수는 전체 네트워크에 속한 네트워크의 개수로, 거리 벡터의 경우와 같다.
다음 홉 벡터H(x) = [홉(1), 홉(2), ..., 홉(n), ..., 홉(N)]

32.
링크 상태(Link State) 프로토콜에서는 라우터 간의 정보 교환 원리가 거리 벡터 방식과 반대다. 개별 라우터가 주변 라우터까지의 거리 정보를 구한 후 이를 네트워크에 연결된 모든 라우터에 통보한다. 거리 벡터 프로토콜에서는 각 라우터가 상당 양의 정보 전송을 요구받고, 특히 링크 상태가 많이 변하면 동작 과정에서 시간이 많이 소요될 수 있다. 링크 상태 알고리즘은 이와 같은 단점을 극복하려고 고안된 방식이다. 거리 벡터 프로토콜은 알고리즘의 특성상 정보가 주기적으로 전달되지만 링크 상태 프로토콜에서는 주변 상황에 변화가 있을 때만 정보 전달이 이루어진다. 링크 상태 프로토콜은 정보 전달을 위해 플러딩(Flooding) 기법을 사용한다. 최초의 라우터가 주변의 모든 라우터에 정보를 전달하고, 다시 이들 라우터에서 주변의 모든 라우터에 정보를 전달하는 방식으로 동작한다. 이때 정보가 입력된 선로를 통해 출력되지 않도록 주의해야 한다. 플러딩 방식을 사용하면 특정 패킷이 네트워크에서 무한정 순환할 수 있다는 점을 고려해야 한다. 간단한 해결 방법은 각 라우터가 전송된 패킷을 기억했다가 해당 패킷이 다시 도착하면 무시하고 버리는 것이다. 링크 상태 방식을 사용하는 라우팅 프로토콜에는 TCP/IP 기반 인터넷에서 사용하는 OSPF(Open Shortest Path First)가 있다. 링크 상태 프로토콜과 거리 벡터 프로토콜 모두 다음과 같은 가정을 전제로 동작한다. 각 라우터는 주변 라우터의 주소 정보와 함께 이들 라우터까지 패킷을 전송하는 데 필요한 비용 정보를 알고 있으며, 이때 비용의 종류는 패킷 전송 지연 등을 비롯해 여러 가지가 될 수 있다.

33.
IP 프로토콜 헤더의 필드는 기능에 따라 Service Type 필드, 패킷 분할 필드, 주소 관련 필드, 그 외 필드로 나눌 수 있다. IP 프로토콜 헤더는 다음 그림과 같다.



■ Service Type 필드

Service Type 필드는 IP 프로토콜이 사용자에게 제공하는 서비스의 품질에 관련된 내용을 표현한다. 크기는 8비트며 각 비트 값의 의미는 다음과 같다.

표 7-4 Service Type

비트 번호	각 비트의 값	
	0	1
0~2	우선순위(111 : 가장 높음)	
3	보통의 지연	낮은 지연
4	보통의 전송률	높은 전송률
5	보통의 신뢰성	높은 신뢰성
6~7	예약	

■ 패킷 분할

IP 프로토콜에서는 상위 계층에서 내려온 전송 데이터가 패킷 하나로 전송하기에 너무 크면 분할(Fragmentation)해 전송하는 기능을 제공한다. 다음은 패킷 분할과 관련된 필드다.

• Identification(식별자 혹은 구분자) : IP 헤더의 두 번째 워드에는 패킷 분할과 관련된 정

보가 포함된다. 이중 Identification 필드는 송신 호스트가 지정하는 패킷 구분자 기능을 수행한다. IP 프로토콜이 분할한 패킷에 동일한 고유 번호를 부여함으로써, 수신 호스트가 Identification 번호가 같은 패킷을 다시 병합(Reassembly)할 수 있도록 해준다. 패킷을 분할하지 않으면 패킷을 전송할 때마다 이 필드의 값을 하나씩 증가시킨다.

- DF(Don't Fragment) : 패킷이 분할되지 않도록 한다. 즉, 값을 1로 지정하면 패킷 분할을 막을 수 있다.
- MF(More Fragment) : 분할된 패킷을 전송할 때는 여러 개의 분할 패킷이 연속해서 전송되므로 MF 필드 값을 1로 지정하여, 분할 패킷이 뒤에 계속 발생됨을 표시해주어야 한다. 분할 패킷 중 마지막 패킷은 MF 비트를 0으로 지정하여 분할 패킷이 더 없음을 표시한다.
- Fragment Offset(분할 옵셋) : 패킷 분할이 이루어지면 12비트의 Fragment Offset 필드를 사용한다. 저장되는 값은 분할된 패킷의 내용이 원래의 분할 전 데이터에서 위치하는 상대 주소 값이다. 값은 8바이트 배수므로, Fragment Offset 값이 64라면 원래 데이터에서 64×8 = 512번째에 위치한다.

■ 주소 관련 필드

Source Address와 Destination Address 필드는 송수신 호스트의 IP 주소다. Source Address는 송신 호스트 주소고, Destination Address는 수신 호스트 주소다.

■ 기타 필드

- Version Number(버전 번호) : IP 프로토콜의 버전 번호다. 현재 인터넷 환경에서 사용하는 IP 프로토콜은 버전 4다. IP 프로토콜의 새로운 버전인 IPv6과 구분하기 위해 기존 IP 프로토콜을 IPv4로 표현하기도 한다.
- Header Length(헤더 길이) : IP 프로토콜의 헤더 길이를 32비트 워드 단위로 표시한다. 일반 패킷을 전송하는 경우에 헤더의 Options 부분 이하가 빠지므로 IP 헤더의 최소 크기는 5다.
- Packet Length(패킷 길이) : IP 헤더를 포함하여 패킷의 전체 길이를 나타낸다. 필드의 크기가 16비트므로 IP 프로토콜에서 지원하는 패킷의 최대 크기는 216-1바이트다. 그러나 이는 이상적인 최댓값으로, IP 프로토콜에서 65,535바이트의 IP 패킷을 전송해도 대부분 데이터 링크 계층에서 분할해 전송한다. 따라서 실제 환경에서 IP 프로토콜은 IP 패킷을 더 작은 단위로 만든다. IP 패킷의 크기는 일반적으로 8,192바이트를 넘지 않는다.
- Time To Live(생존 시간) : 패킷 전송 과정에서 패킷이 올바른 목적지를 찾지 못하면 수신 호스트에게 제대로 도착하지 않고, 네트워크 내부에서 떠돈다. 이런 현상을 방지하려고 Time To Live 필드를 사용한다. 송신 호스트가 패킷을 전송하기 전에 네트워크에서 생존할 수 있는 시간을 이 필드에 지정하고, 각 라우터에서는 패킷이 지나갈 때마다

필드 값을 감소시키면서 패킷을 중개한다. 임의의 라우터에서 Time To Live 값이 0으로 감소하면 패킷은 자동으로 버려지고, 패킷 송신 호스트에게 ICMP 오류 메시지가 전달된다.

- Transport Protocol(전송 프로토콜) : IP 패킷을 생성하도록 IP 프로토콜에게 데이터 전송을 요구한 전송 계층의 프로토콜을 가리킨다. TCP는 6을, UDP는 17을, ICMP는 1을 지정한다.
- Header Checksum(헤더 체크섬) : 전송 과정에서 발생할 수 있는 헤더 오류를 검출하지만 데이터의 오류는 검출하지 않는다. 이와 달리 계층 4 프로토콜인 TCP, UDP 헤더는 데이터와 헤더 오류를 모두 검출한다. 체크섬 계산 과정은 다음과 같다. 먼저 Header Checksum 필드의 비트 값을 모두 0으로 설정한 후, 전체 헤더가 16비트 워드의 연속이라 가정하고 1의 보수 합을 수행한다. 이 값을 체크섬으로 하여 패킷을 전송하고, 수신 호스트는 1의 보수 합을 계산하여 계산 결과가 모두 1이면 전송 과정에 오류가 없다고 판단한다. 전송 과정에 오류가 발생하면 IP 프로토콜에서는 해당 패킷을 버리고, 이를 복원하는 일은 상위 계층에서 담당한다.
- Options(옵션) : 네트워크 관리나 보안처럼 특수 용도로 이용할 수 있다.
- Padding(패딩) : IP 헤더의 크기는 16비트 워드의 크기가 4의 배수가 되도록 설계되어 있다. 앞서 설명한 필드의 전체 크기가 이 조건에 맞지 않으면 이 필드를 사용해 조정할 수 있다.

34.

IP 클래스는 주소를 network와 host 필드로 구분해 관리함으로써, 클래스별로 네트워크 크기에 따라 주소 관리를 다르게 한다.

- network(네트워크) : 네트워크 주소다. 전 세계적으로 유일한 번호가 모든 컴퓨터 네트워크에 할당된다. 현재 이 주소의 할당은 NIC(Network Information Center)에서 담당한다.
- host(호스트) : 네트워크 주소가 결정되면 하위의 호스트 주소를 의미하는 host 비트 값을 개별 네트워크의 관리자가 할당한다. A 클래스는 host 비트의 크기가 크기 때문에 규모가 큰 네트워크에서 사용하고, C 클래스는 규모가 작은 네트워크에서 사용한다.

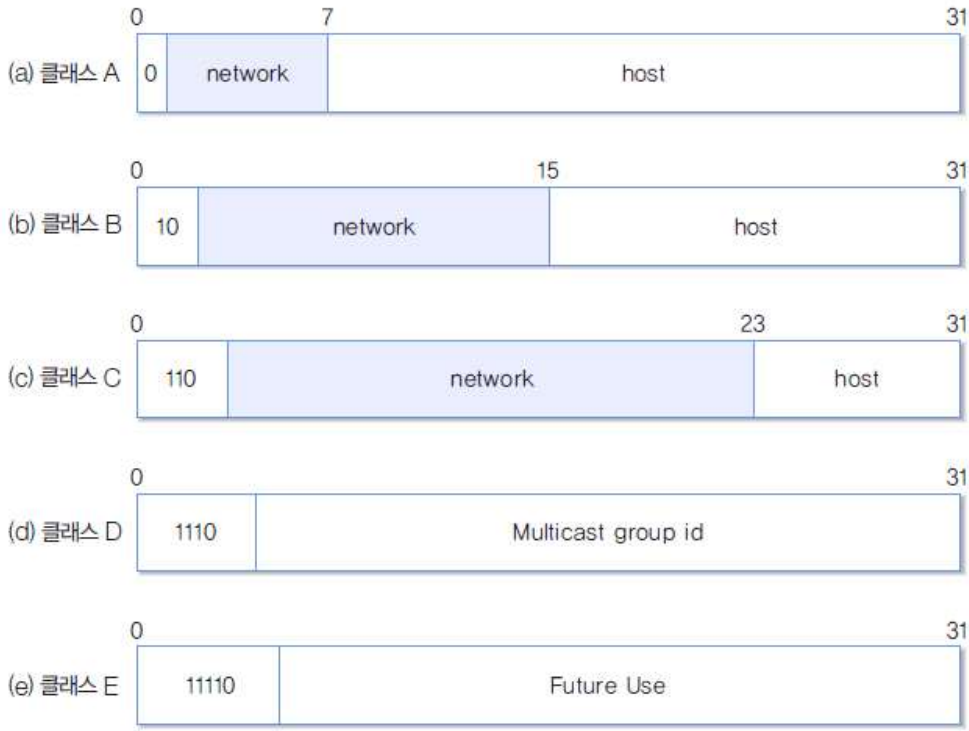


그림 7-13 IP 주소 체계

그림에서 유추할 수 있는 것처럼 IP 주소 값을 통해 이 주소가 속한 클래스를 알 수 있다. 이진수이면 그림에서 맨 왼쪽 비트와 비교해 바로 알 수 있다. 십진수이면 A1.A2.A3.A4의 표현에서 맨 왼쪽의A1 값이 아래의 주소 대역 중 어디에 속하는지에 따라 결정된다. 예를 들어, 211.223.201.30은 클래스 C의 주소다.

표 7-6 IP 주소 값에 따른 주소 체계

IP 주소 값	주소 체계
0.0.0.0 ~ 127.255.255.255	클래스 A의 주소 대역
128.0.0.0 ~ 191.255.255.255	클래스 B의 주소 대역
192.0.0.0 ~ 223.255.255.255	클래스 C의 주소 대역
224.0.0.0 ~ 239.255.255.255	클래스 D의 주소 대역
240.0.0.0 ~ 255.255.255.255	클래스 E의 주소 대역

맨 왼쪽 4비트가 1111인 경우로 예약 주소 대역이다.

35.
다양한 유형의 네트워크를 통해 패킷을 중개하려면, IP 프로토콜이 패킷을 각 네트워크에서 처리하기 편한 크기로 분할(Fragmentation)해야 한다. 예를 들어, X.25 프로토콜에서의 프레임 크기와 이더넷에서의 프레임 크기는 다르다. 따라서 상위 계층에서 더 큰 데이터 전송을 요구하면 IP 프로토콜에서 패킷 분할 과정을 먼저 수행해 전송한다.

IP 헤더를 제외한 전송 데이터의 크기는 380바이트고, 패킷은 최대 크기가 128바이트라고 가정하였다.

그림 7-16 **패킷 분할의 예**

먼저 Fragment Offset 필드를 계산해야 하는데, 이 값에 8을 곱한 크기가 분할 전의 데이터 위치다. 패킷 전체의 최대 크기인 128바이트에서 헤더인 20바이트를 빼면 108바이트가 되므로 분할 패킷에 보관할 수 있는 데이터의 최대 크기는 (108을 8로 나눈 몫(정수값))×8=104바이트다. 따라서 분할된 패킷의 개수는 4개(=380을 108로 나눈 몫+1)며, 각 패킷의 Fragment Offset 필드 값은 0, 13, 26, 39가 된다.
분할 패킷인 분할 1, 분할 2, 분할 3은 데이터 크기 104바이트에 헤더 크기 20바이트를 더해 124가 되므로 패킷의 전체 크기 Packet Length=124바이트다. 마지막 분할 패킷은, 전체 데이터의 크기 380에서 세 개의 분할 패킷 크기 3×104를 빼면 68바이트의 여분을 얻을 수 있는데 이 값에 헤더 크기인 20바이트를 더해 Packet Length=88바이트다. 분할한 패킷의 Identification 필드에는 동일한 번호를 부여해야 한다. 그림에서는 임의로 1254번을 가정하였다. MF 필드는 마지막 패킷만 제외하고 1을 지정해 분할 패킷이 이어지고 있음을 표시해 주어야 한다. DF 비트는 지정되지 않은 것으로 가정하였다.