네트워크프로그래밍(주경호)

Lee Jun Hyeok (wnsx0000@gmail.com)

June 11, 2025

목차

1. Review

데이터통신과네트워크(주경호)에서 다뤘던 내용 중 본 수업에 필요한 부분을 review함.

1.1. Data Communication

1.1.1. Data Communication

1. Data Communication
통신(Telecommunication)은 전화/정보/텔레비전 등을 포함하는 원거리 의사소통을 의미함.
데이터통신(Data communication)은 두 기기 간에 데이터를 주고받는 통신을 의미함.

data communication는 하드웨어(chipset, NIC chip 등)와 소프트웨어(소켓, 펌웨어 등)로 구현됨.

웹, wifi, bluetooth 등 data communication을 수행하는 다양한 시스템들이 있음.

2. 주요 특성들
data communication은 아래와 같이 4가지 주요 특성들을 가짐.

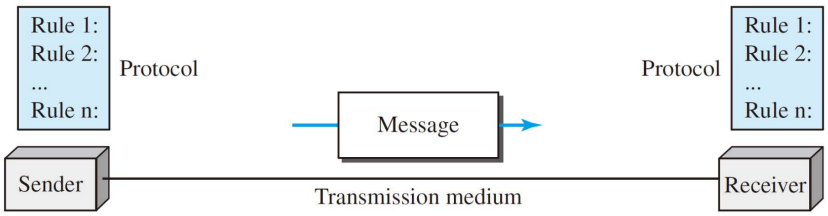
전달(delivery) : 시스템은 데이터를 올바른 목적지에 전달해야 함.
정확성(accuracy) : 시스템은 데이터를 변동 없이 정확하게 전달해야 함.
적시성(timeliness) : 시스템은 데이터를 고려된 시간 내에 전달해야 함.
파형 난조(jitter) : 시스템은 jitter가 최대한 일어나지 않도록 해야 함.

여기서 jitter는 패킷 도착 시간 지연에 따른 딜레이나 버벅거림을 의미함.

3. 구성 요소
data communication은 아래와 같이 5가지 구성 요소들을 가짐.

메시지(message) : 주고받는 데이터.
송신자(sender) : message를 보내는 사람.
수신자(receiver) : message를 받는 사람.
전송 매체(transmission medium) : 메시지가 이동하는 물리적인 경로. (cable, 공기)
프로토콜(protocol) : data communication을 제어하는 규약의 집합.

정보를 전달하는 물리적/논리적 경로를 통틀어 채널(channel)이라고도 함.



1.1.2. Data Representation

데이터 표현 방법(Data Representation)에는 아래와 같은 것들이 있음.

1. text
bit pattern의 집합으로 표현함. 이 집합을 code라고 하고, 텍스트를 나타내는 과정을 coding이라고 함.
code에는 Unicode, ASCII 등이 있음.

Unicode는 32bit로 pattern을 구성하고, 세상의 모든 언어를 표현할 수 있음. ASCII는 8bit로 pattern을 구성하고, Unicode의 첫 127개(로마자 등)에 해당하는 문자를 표현할 수 있음.

2. number
number는 이진수로 단순 변환함.

3. image, video

image는 픽셀(pixel)에 대한 행렬로 표현함. image의 색은 3가지 정보(RGB)로 표현함.

pixel은 디지털 image를 구성하는 가장 작은 단위로, 각 pixel은 하나의 색상을 나타냄. image의 해상도(resolution)는 pixel의 개수에 의해 결정됨.

video는 image의 집합으로 표현함.

4. audio
이산적으로 표현이 가능한 text, number, image와는 달리, 연속적인 데이터를 가지므로 별개의 표현 방식을 가짐.

1.1.3. Data Flow

데이터가 전송될 수 있는 방향에 따라 데이터 흐름(Data Flow)을 분류할 수 있음. 여기서의 data flow는 일대일 통신을 가정함.

1. simplex
단방향(simplex)는 단방향 통신을 의미함. 연결된 두 장치 중 하나에서는 송신만 가능하고, 다른 하나에서는 수신만 가능함.

예를 들어, 키보드, 단순 모니터 등이 있음.

2. half-duplex
반이중(half-duplex)는 각 장치에서 송/수신을 동시에 수행하지 못하는 양방향 통신을 의미함. 연결된 두 장치는 송/수신이 모두 가능하지만, 한쪽이 송신하고 있으면 다른 쪽은 수신만 할 수 있음.

예를 들어, 무전기, 개인 라디오(CD radio) 등이 있음.

3. full-duplex
전이중(full-duplex)는 각 장치에서 송/수신을 동시에 수행할 수 있는 양방향 통신을 의미함. 연결된 두 장치는 데이터를 송신하면서 수신받을 수 있음.

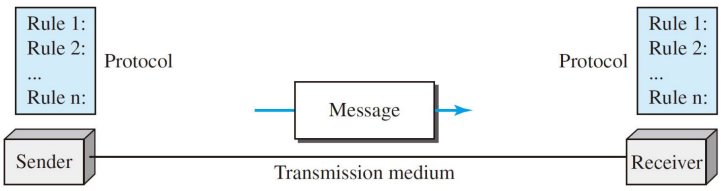
예를 들어, 전화 등이 있음.

1.2. Network

1.2.1. Network

1. Network
네트워크(Network)는 소통 가능한 기기들의 상호 연결 집합임. host-client 구조에서 각 기기들을 host가 될 수도 있고, client가 될 수도 있음.

2. connecting device
network에서는 연결 장치(connecting device)가 각 layer에 대해 동작하여 데이터 전송 및 장치 연결을 수행함.

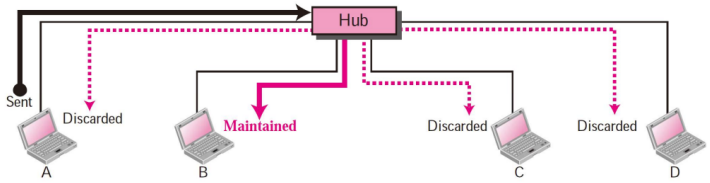


아래와 같이 3가지 종류의 connecting device가 존재함.

1) Repeater/Hub
Repeater는 physical layer에 대한 connecting device임. 전달받은 signal이 너무 작아지거나 corrupt되지 않도록 원본 bit 패턴 재생성 및 타이밍 조정 등을 수행함.

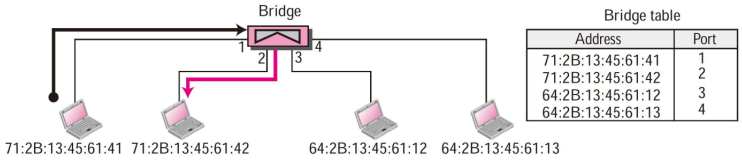
현재 ethernet에서는 star topology를 주로 사용하는데, 여기에서의 repeater는 기존의 기능만이 아니라 각 point를 연결하는 역할도 수행함. 이때의 repeater를 Hub라고도 함. 이 경우 주소에 따른 forwarding

이 아니라 모든 port로 단순 전송하므로 hub는 switch라고 볼 수 없음.



2) Bridge
Bridege 또는 L2 switch는 physical/data-link layer에 대한 connecting device임. physical layer 측면에서는 입력받은 signal을 재생성해서 전송하고, data-link layer 측면에서는 physical address를 확인하여 자신의 port 중 어디로 전송할지를 결정함. 다시 말해, network 내부에서 MAC 주소를 기반으로 여러 장치를 연결하고 forwarding하는 장치.

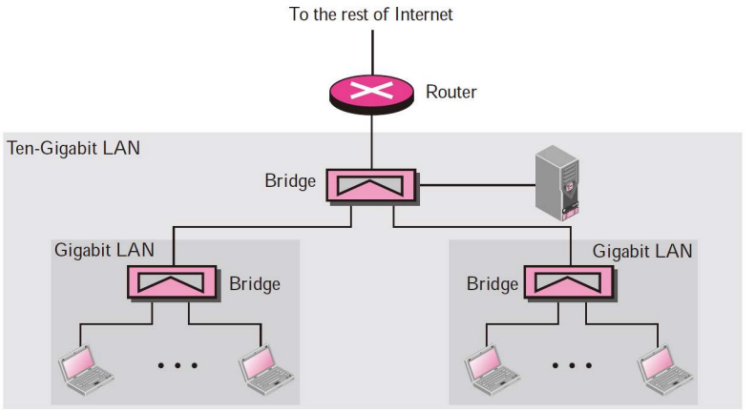
단순히 switch라고 하면 bridge를 의미함.



3) Router
Router는 또는 L3 switch는 physical/data-link/network layer에 대한 connecting device임. physical layer 측면에서는 입력받은 signal을 재생성해서 전송하고, data-link layer 측면에서는 physical address를 확인하고, network layer 측면에서는 network layer address를 확인함. 다시 말해, routing table을 사용하여 network를 다른 network와 연결하는 장치로 packet 전달, 경로 선택, 보안 등을 처리함.

router는 physical address와 logical address 모두를 가짐. 또한 router는 연결된 여러 network에 대해 interface(논리적/물리적 port)를 가지는데, 각 interface는 해당 network에 포함된 IP address를 할당받음. 즉, router는 여러 개의 IP address를 가지게 됨. 예를 들어, 뒤에서 설명하는 option 관련 그림에서 router는 network별로 IP address를 가짐.

repeater와 bridge는 LAN 내부에서의 통신을 처리한다면, router는 network(LAN, WAN)끼리의 연결을 처리함. 즉, internetwork를 구성함.



3. Network 평가 기준
Network의 평가 기준에는 아래와 같은 것들이 있음.

성능(Performace) : 데이터 처리량과 지연 정도를 평가.
신뢰성(Reliability) : 실패 빈도, 실패 시 복구 시간, 재해 등에 대한 견고함 등으로 평가.
보안(Security) : 비인가된 접근, 데이터 보호, 데이터 손실에 대한 복구 정책/과정 등으로 평가.

1.3. Network Types

1.3.1. LAN

LAN(Local Area Network)은 가정, 사무실, 건물, 학교 단위의 network를 말함. LAN은 host들을 서로 연결함.

여러 방식으로 구현할 수 있음. 과거에는 bus topology로 구현하기도 했지만, 지금은 주로 star topology로 구현함. 물론 wifi 등 무선 LAN도 존재함.

1.3.2. WAN

1. WAN

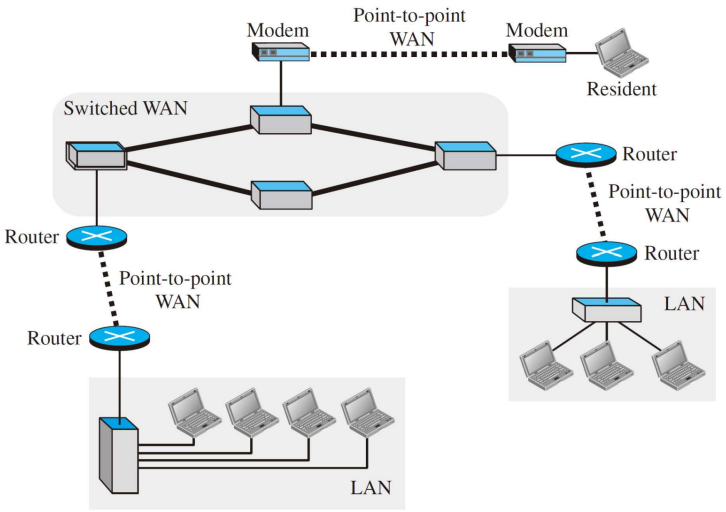
WAN(Wide Area Network)은 도시, 국가 단위의 network를 말함. WAN은 connecting device(router, switch, modem 등)를 활용해 구성될 수 있음.

WAN은 여러 방식으로 구현됨. connecting device를 활용하여 switched WAN을 구성하기도 하고, point-to-point WAN을 구성하기도 함.

2. internetwork

internetwork(internet)는 2개 이상의 network가 연결된 것을 말함.

여기서의 internet은 Internet과는 구분되는 개념임.



3. Internet

인터넷(Internet)은 전세계의 장치들이 서로 연결되어 데이터를 주고받는 internetwork임.

Internet은 backbone, provider network, customer network, peering point 등으로 구성됨.

최상위 layer에 있는 backbone은 주로 거대 통신 회사(AT&T, NTT 등)들에 의해 운영되는 대규모 network임. 그 다음 layer에 있는 provider network는 backbone의 서비스를 하위 newwork에 제공하는 network임. backbone과 provider network를 통틀어 인터넷 서비스 제공자(ISP, Internet Service Provider)라고 부름.

customer network는 하위 layer으로서 Internet으로부터 제공받는 서비스를 이용하는 network임.

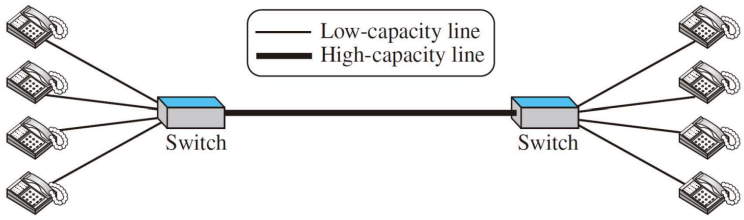
peering point는 두 대상을 연결하여 데이터를 직접적으로 교환할 수 있도록 하는 연결점임.

1.3.3. Switching

스위칭(Switching)은 데이터를 적절한 위치로 보내주는(forwarding하는) 작업을 의미함. network에는 switching의 방식에 따라 Circuit-Switched Network와 Packet-Switched Network가 있음.

1. Circuit-Switched Network
switch가 두 장치 사이의 서킷(circuit)이라는 전용 회선(dedicated connection)을 제공하는 방식. switch는 각 회선에 대한 활성화/비활성화를 수행함.

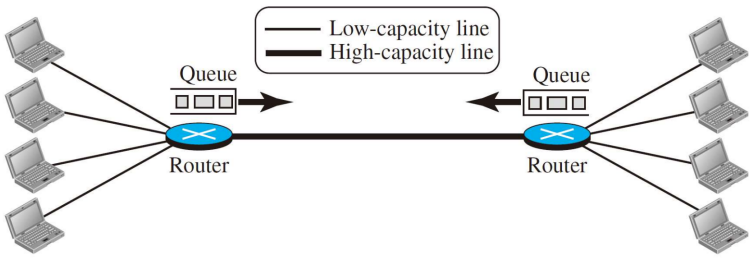
주로 과거 전화선 등에 사용되었음. 현재의 전화는 packet-switched network를 주로 활용함.



2. Packet-Switched Network
두 장치가 패킷(packet)이라는 단위로 데이터를 주고받는 방식. dedicated circuit을 사용한 연속적인 연결 대신, router를 사용하여 독립적인 packet의 교환으로 통신이 이루어짐.

packet-switched network는 packet을 저장하고 forwarding하기 위한 queue를 사용함.

음성 데이터 등을 교환하기 위해서는 데이터를 인코딩/디코딩하고 packet으로 만들어야 함.



3G 전화에서는 circuit switching을 썼지만, voLTE에서부터는 packet switiching을 씀. VoLTE는 LTE(4G) 네트워크를 통해 음성 통화를 제공하는 기술임. 또한 VoIP는 일반적인 internet 연결(Wi-Fi, ethernet 등)을 통해 음성 통화를 제공하는 기술임. 즉, 통신사의 이동통신망이 아닌, 인터넷 서비스 제공자(ISP)의 네트워크를 기반으로 동작함.

1.4. Protocol

1.4.1. Protocol

1. Protocol
프로토콜(Protocol)은 효율적인 통신을 위해 수신/송신/연결 장치들이 지켜야 하는 규칙의 집합임.

2. Protocol Layering
복잡한 통신에 대한 protocol은 각 작업에 대한 계층화(Layering)를 통해 설계/구현/유지보수를 단순화할 수 있음. 이때 각 layer에서는 인접한 layer과의 상호작용만 고려하면 됨.

protocol layering에 의한 network model로는 TCP/IP, OSI Model 등이 있음.

3. principles of protocol layering
protocol layering 시에 각 layer가 지켜야 하는 원칙은 아래와 같음.

1) 각 layer는 자신에게 책임이 있는 작업과 그 반대 작업을 모두 수행할 수 있어야 함.
2) 통신 중인 두 장치에 대해, 동일한 layer에 위치하면서 서로 대응되는 두 packet은 그 값이 동일해야 함.

예를 들어, Encrypt/Decrypt layer는 encrypt/decrypt 작업을 모두 수행할 수 있어야 하고, 한쪽에서 encrypt해서 보낸 데이터를 decrypt하면 원래의 데이터와 동일해야 함.

1.4.2. Internet 주소 체계

Internet에서는 특정 장치를 식별하기 위해 아래와 같은 주소들을 사용함. 자세한 내용은 '데이터통신과네트워크(주경호)' 필기와, 본 필기의 뒤에서 설명함.

1. MAC 주소
MAC(Media Access Control) 주소는 네트워크의 각 장치가 가지고 있는 NIC(Network Interface Card)에 할당된 주소로, network 내에서 장치를 식별하기 위해 사용됨. 16진수로 이뤄져 있음.

이더넷(Ethernet)은 LAN에서의 MAC 주소 관련 처리에 대한 프로토콜임. ethernet을 사용하는 network를 ethernet network라고 함.

2. IP 주소
network 내에서 장치를 식별하기 위해 사용되는 주소. 길이에 따라 IPv4, IPv6가 있고, 사설 ip와 공인 ip 등의 분류가 존재함.

1.4.3. TCP/IP

1. TCP/IP
TCP/IP(Transmission Contorl Protocol/Internet Protocol)는 오늘날 Internet에서 사용되는 프로토콜 집합임.

TCP/IP는 아래와 같이 5개의 layer로 구성되어 있고, 각 layer는 나름의 프로토콜을 포함하고 있음.

1) Layer1. 물리(Physical) layer : 물리적인 연결을 통해 비트를 전송하는 layer.

2) Layer2. 데이터 링크(Data link) layer : 적절한 링크의 탐색을 수행하는 layer. (ethernet 등.)

3) Layer3. 네트워크(Network) layer : 송신자와 수신자의 연결을 생성하는 layer. (IP 등.)

4) Layer4. 전송(Transport) layer : port 선택, 데이터 reliability, 흐름 관리 등을 수행하는 layer. (TCP, UDP 등.)

5) Layer5. 응용(Application) layer : 사용자와의 상호작용을 처리하는 layer. (HTTP, HTTPS, DNS 등.)

위에서 표현한 5개의 layer는 OSI 7계층에서 5개의 layer를 가져와 설명한 것이고, 4개의 layer로 설명하기도 함. 이 경우 network layer를 Internet layer로 부르고, data link layer와 physical layer를 묶어서 네트워크 액세스 계층(Network Access layer)으로 부름. 물론 의미는 동일함.

2. 캡슐화/역캡슐화
송신자 측에서 데이터는 application layer에서 physical layer로 흐르게 되는데, 이때 각 layer에 대한 헤더가 추가되며 캡슐화(encapsulation)됨. 이 헤더는 수신자 측에서 layer를 거치며 역캡슐화(decapsulation)됨. 이때 데이터는 encapsulation된 상태에 따라 다른 명칭으로 부름.

encapsulation에 의한 데이터 교환 단위는 아래와 같이 layer별 명칭이 존재하고, 단순히 각 단위를 통틀어 패킷(packet)이라고도 부름.

1) 비트(bit) : physical layer에서는 frame을 비트로 변환하여 전송함.

2) 프레임(frame) : data link layer에서 ethernet 헤더를 추가한 것. MAC 주소 등을 포함함.

3) 데이터그램(datagram) : network layer에서 segment에 ip 헤더를 추가한 것. ip 주소 등을 포함함.

4) 세그먼트(segment, TCP)/데이터그램(datagram, UDP) : transport layer에서 message에 transport layer 헤더를 추가한 것. port 번호, 시퀀스 번호, 흐름 제어 정보 등을 포함함.

5) 메시지(message) : http의 경우 http 메시지. application layer에서 요청/응답에 http 헤더 등을 추가한 것. uri 경로, http 메서드 등을 포함함.

3. Addressing
bit로 표현되므로 주소를 가질 수 없는 physical layer를 제외한 layer들은 출발 주소와 도착 주소를 활용하여 통신함.

layer별로 가지는 대표적인 주소는 아래와 같음.

1) data link layer : physical address(MAC 주소).

2) network layer : logical address(ip 주소). network에 따라 다를 수 있는 physical address에 독립적인 통신을 가능하게 함.

3) transport layer : port address(port 번호). 기기 내의 process 식별을 위한 주소.

4) application layer : application specific address(email, URL 등). 사용자 친화적인 주소.

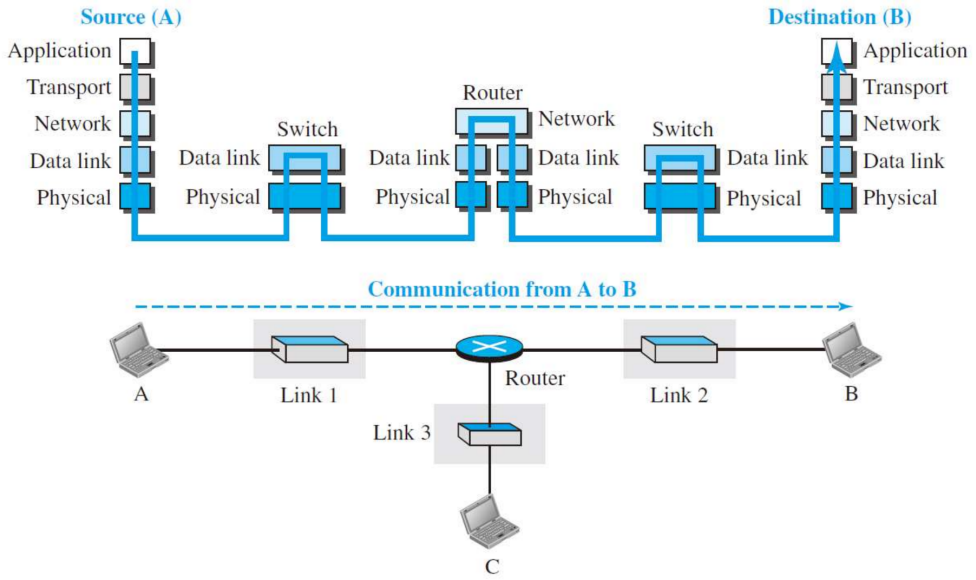
4. 데이터 전달 과정

encapsulation된 데이터는 connecting device(switch, router 등)들을 거치며 목적지로 전달됨.

switch는 data link layer를 처리함. 즉, LAN에 대해 동작하고, MAC 주소를 확인하여 적절한 장치(swtich, router, 목적지 등)에 데이터를 전송함.

router는 network layer를 처리함. 즉, WAN에 대해 동작하고, ip 주소를 확인하여 적절한 장치(switch, router, 목적지 등)에 데이터를 전송함.

목적지를 발견했으면 데이터의 헤더가 layer 순서대로 사용/삭제되며 처리됨.



5. end-to-end/hop-to-hop

3, 4, 5 layer는 end-to-end를 보장해야 함. 즉, 3, 4, 5 layer에 대한 데이터는 중간 장치(hop, host와 router 등)의 개입 없이 송신자와 수신자끼리만 사용해야 하고, hop에 의해 패킷이 수정되면 안 됨. 이 layer들은 internet에서의 통신을 처리함.

1, 2 layer는 hop-to-hop을 보장해야 함. 즉, 1, 2 layer에 대한 데이터는 hop에서 사용해야 함. 이 layer들은 하드웨어적인 link에서의 통신을 처리함.

여기에서 hop은 host나 router를 의미함.

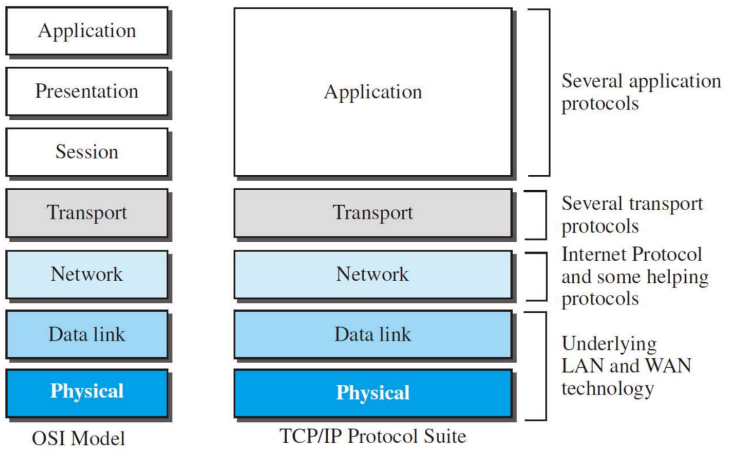
router는 data link layer 데이터(frame)에 대해 decapsulation을 수행하고, swtich는 데이터를 수정하지 않음.

1.4.4. OSI Model

OSI(Open System Interconnection) 7계층 model은 ISO(International Organization for Standardization)에서 제정한 프로토콜 집합임.

ISO는 국제 표준을 지정하는 국제 기관임.

모든 통신과 장치에 각 layer를 구현하는 대신, 필요한 layer만 구현하기도 함. 또한 실제로는 TCP/IP 프로토콜을 주로 사용함.

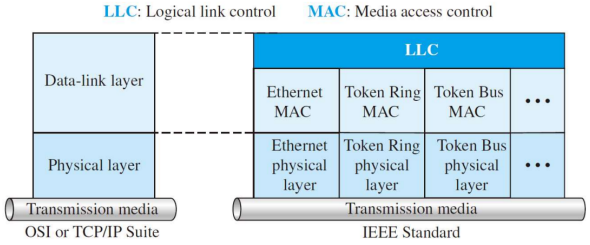


1.5. Ethernet

1.5.1. IEEE project 802

IEEE project 802는 ethernet을 포함한 physical/data-link layer protocol의 구체적인 표준을 제공한 프로젝트임.

IEEE project에서는 data-link layer를 LLC(Logical Link Control. DLC와 대응되는 개념)와 MAC으로 나누었음. LLC는 framing, flow/error contorl 등 모든 IEEE LAN에서 단일 link 통신을 지원하는 부분임. MAC는 각 LAN에 따라 특정 기능을 수행하는 부분임.



1.5.2. Ethernet

Ethernet은 wired network에 대한 physical/data-link layer protocol임.

ethernet은 data rate에 따라 standard/fast/gigabit/10-gigabit ethernet으로 나뉨.

과거부터 여러 protocol들이 있었지만, ethernet을 제외한 대다수가 사라졌음.

1.5.3. Standard Ethernet

Standard Ethernet은 가장 표준적인 ethernet으로, 아래와 같은 특징들을 가짐.

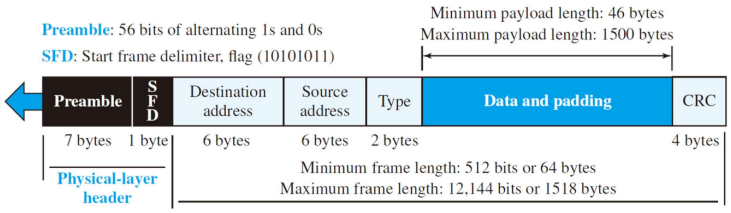
1. Connectionless/Unreliable
standard ethernet에서 frame을 주고받는 통신에서는 connection이 존재하거나, reliable하지 않음.

각 frame은 독립적으로 전송되며, 송신자는 실제로 전송되었는지를 확인하지 않음. 수신자도 frame이 제대로 전달되지 않았으면 이를 단순히 무시함. statndard ethernet에서는 이런 책임을 상위 layer에게 넘김.

2. Frame Format
standard ethernet의 frame foramt은 아래와 같음.

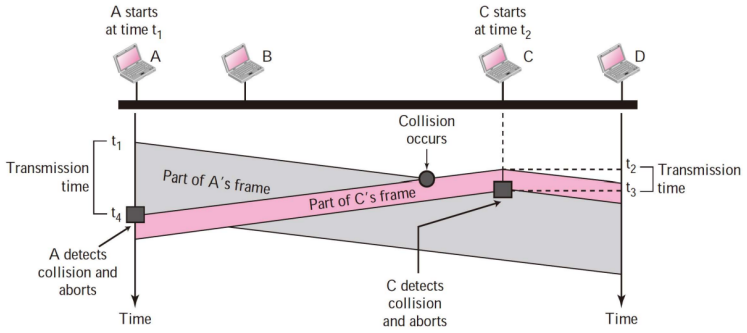
1) Preamble : synchronization을 위한 부분. 010101···로 구성됨.
2) Start Frame Delimiter(SFD) : frame 시작점에 대한 delimiter. 01010111임.

3) DA/SA : destination과 source에 대한 link-layer address임.
4) Type : frame이 encapsulation한 packet이 상위 layer에서 사용한 protocol 정보임. (ex. IP)
5) Data : 상위 layer로부터 받은 packet. packet 길이에 따라 길이가 가변적임.
6) CRC : CRC의 redundant bit.



4. CSMA/CD

ethernet은 CSMA/CD를 사용함.



1.5.4. MAC address

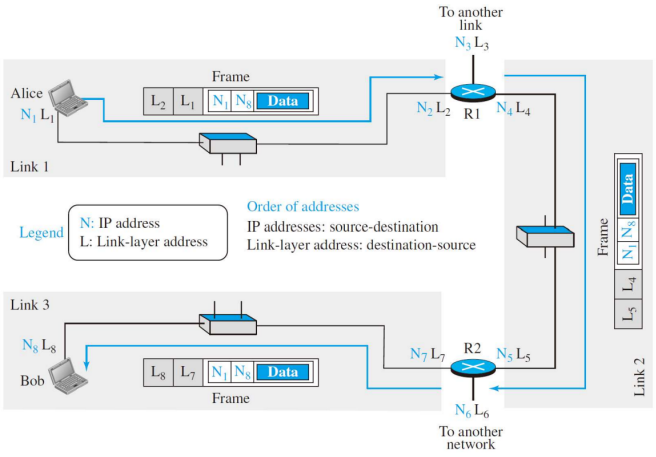
1. MAC address

MAC address는 data-link layer에서 사용하는 address로, physical address, link address라고도 함. 이는 48bit 길이의 address로, 아래와 같이 16진수로 두 자리씩 끊어서 나타냄.

$$0A : 00 : 27 : 00 : 00 : 04$$

data-link layer에서는 하나의 link에서의 통신만을 고려함. MAC address는 하나의 link에서 출발 node와 도착 node에 대한 address이고, 하나의 link에서 다음 link로 넘어갈 때 그 값이 새롭게 지정됨.

ethernet에서 각 장치는 NIC(Network Interface Chip)를 가지고 있고, NIC는 고유한 MAC address를 할당받음.



2. 종류
MAC address에는 3가지 종류가 있음.

1) Unicast Address : link에 존재하는 하나의 node에 대한 address.
ethernet에서 사용되는 unicast address는 48bit 크기이며, 이는 주로 12개의 16진수를 :으로 구분하여 표기함. 예를 들어, A3:34:45:11:92:F1 등으로 표기함.

2) Multicast Address : link에 존재하는 여러 node에 대한 address.

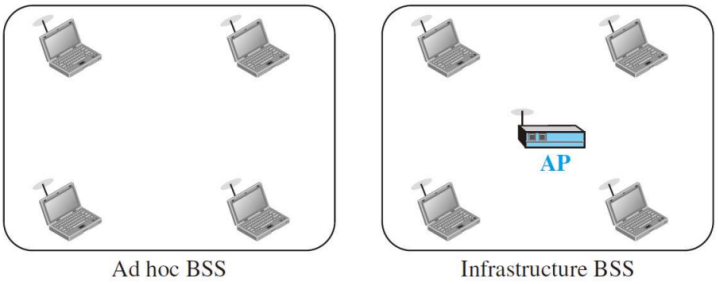
3) Broadcast Address : link에 존재하는 모든 node에 대한 address.
ethernet에서 사용되는 broadcast address는 48bit 크기이며, 모든 값이 1임. 즉, FF:FF:FF:FF:FF:FF 임.

1.5.5. IEEE 802.11

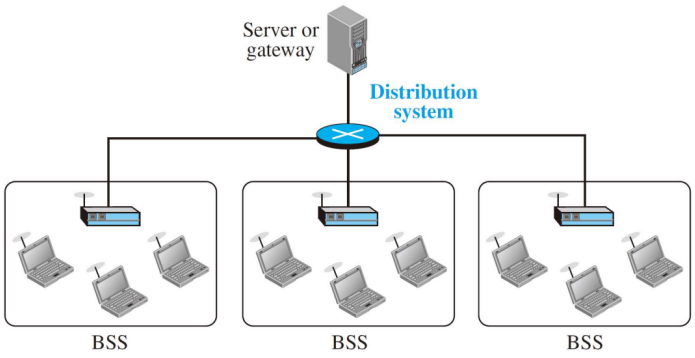
IEEE 802.11은 wireless LAN에 대한 physical/data-link layer protocol의 구체적인 표준임. WIFI 등이 정의됨.

1. Architecture
architectrue는 basic service set과 extended service set으로 나눌 수 있음.

basic service set은 station끼리 통신하거나(ad hoc BSS), station들에 추가로 Access Point(AP)라고 하는 중앙 station을 둬서 통신함(infrastructure BSS). 이때 station들을 묶은 단위를 BSS라고 부름.



extended service set은 AP를 통해 여러 BSS를 distribution system으로 연결함. distribution system 자체는 wired/wireless LAN일 수 있음.



1.5.6. Hidden State Problem

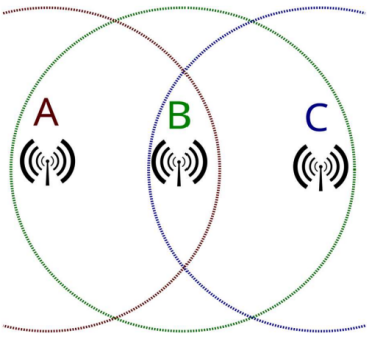
Hidden State Problem은 station a, b, c가 존재할 때, a에서 b로 전송한 데이터가 b에는 전달되지만 c에는 전달되지 않아, c에서도 b에 데이터를 전송하여 collison이 발생할 수 있는 현상임.

CSMA/CA에서 사용하는 RTS와 CTS는 Hidden-Station problem을 해결함.

RTS(request to send)는 송신자가 수신자에게 데이터를 전송해도 되는지를 물어보는 frame이고, CTS(clear to send)는 수신자가 송신자에게 데이터를 전송해도 된다고 응답하는 frame임. RTS에는 해당 통신이 channel을 점유해야 하는 시간 정보를 포함하고 있음. RTS를 전달받은 다른 station들은 Network Al-

location Vector(NAV)라고 하는 timer를 시작하여 RTS에 적힌 시간만큼 대기한 뒤, channel에 접근을 시도함.

RTS/CTS를 사용하면 RTS가 다른 어떤 station에 전달되지 않더라도 CTS는 수신자와 연결된 모든 station에 닿을 수 있으므로 hidden-station problem에 따른 collision이 발생하지 않음.



2. Network Layer

2.1. 서론

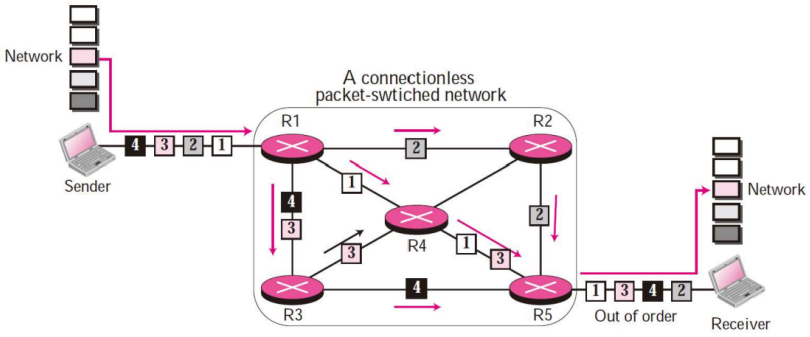
2.1.1. Packet Switching

network layer에서의 packet switching 방식에 대해 알아보자.

network layer는 기본적으로 connectless로 설계되었으나, connection-oriented로 전환되려는 시도 또한 존재한다고 함.

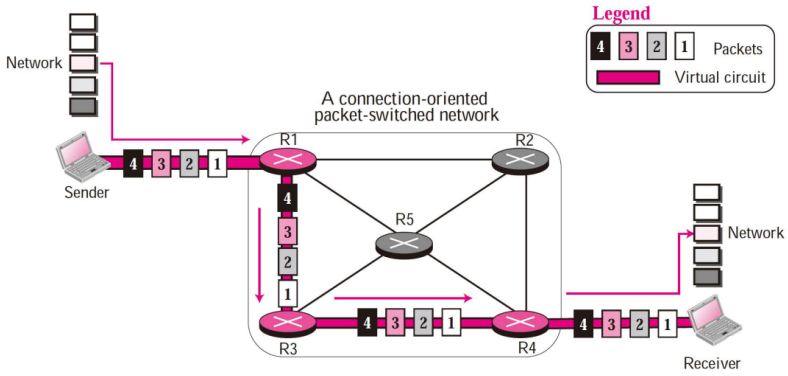
1. Conenctless Service
Connectless Service에서는 network layer에서 각 packet을 독립적으로 취급하고, 각 packet에 대해 destination으로의 전달만을 고려함.

이에 따라 각 packet은 서로 다른 경로로 전송되거나, 순서가 뒤바뀐 채로 도착하거나, 일정하지 않은 시간 간격을 두고 도착하거나, 목적지에 도착하지 못할 수도 있음. 이에 대한 처리는 상위 layer에게 있음.



2. Connection-oriented Service
Connection-oriented Service에서는 network layer에서 서로 같은 message에 속하는 각 packet에 대한 관계성을 고려함.

packet들을 전송하기 전에 해당 message에 대한 virtual connection을 생성해 path를 결정함. 이후 전송 시에 이 packet들은 동일한 path를 따라서 전송됨.



2.1.2. Network Layer Service

network layer에서 제공되는 service를 source computer, router, destination computer의 관점에서 각각 살펴보자. 구체적인 내용은 뒤에 자세히 정리함.

1. Source Computer

source computer에서는 아래와 같은 순서로 작업들이 수행됨. destination logcal address와 next-hop logical address를 잘 구분하자.

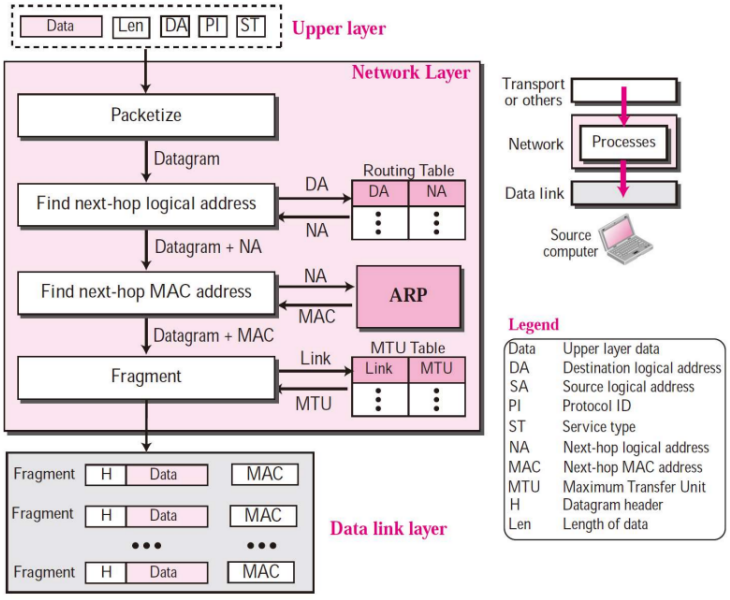
1) 상위 layer로부터 받은 정보로 packet을 구성함.

2) routing table에서 destination logical address를 활용해 next-hop(destination 또는 router) logical address를 얻음. 현재 network에 destination이 존재하지 않는다면 router의 address를 얻게 됨.

3) ARP protocol에 의해 next-hop logical address에 대한 physical address를 얻음.

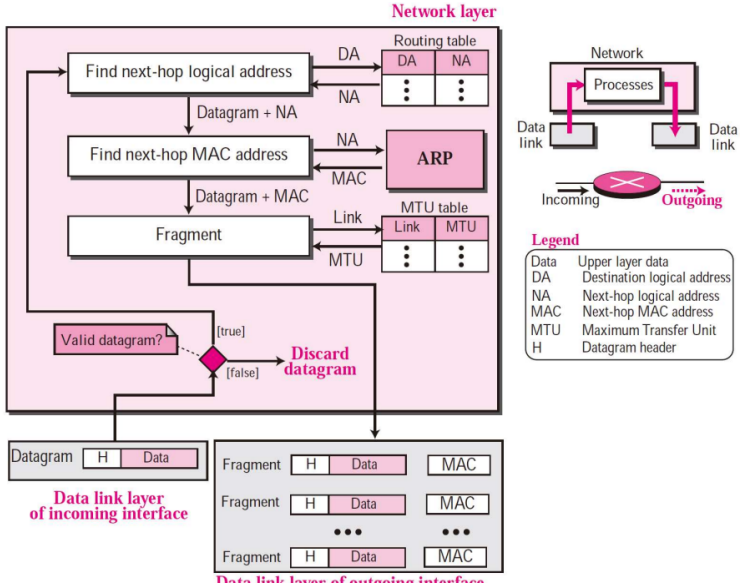
4) MTU table로부터 MTU를 받아 fragmentation함. 뒤에서 설명할 것처럼 대부분의 nework에는 link에 대해 하나의 frame이 가질 수 있는 데이터 최대 크기인 MTU가 정의되어 있고, MTU를 넘어가는 크기의 packet은 작은 크기로 fragmentation해야 함.

5) packet을 data-link layer로 넘김.



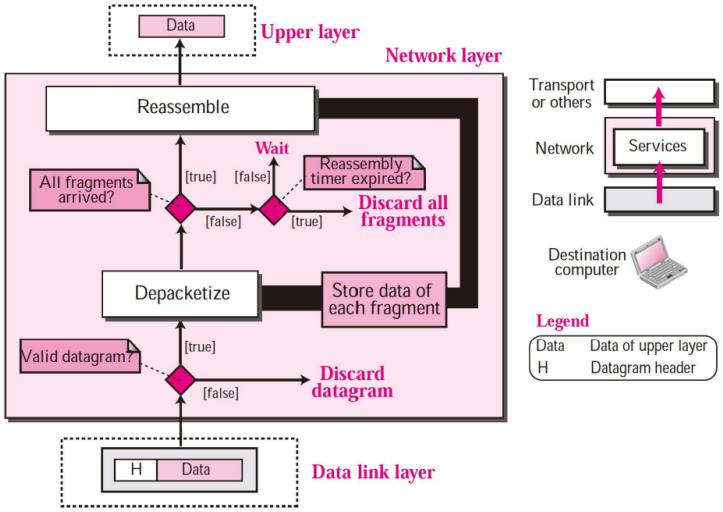
2. Router

data-link layer로부터 packet을 받아 source computer와 유사한 작업을 수행함. 이때 수신받은 packet에 대해 checksum을 계산하여 유효성 검사를 함.



3. Destination Computer

수신받은 packet들은 fragment일 수 있으므로 이를 reassemble함. 이때 reassemble timer를 사용해 timer가 expire되면 해당 fragment 전체를 discard함.



2.1.3. 추가 issue들

network layer에서 고려하는 추가 issue들은 아래와 같음.

1. Error Control

network layer에서도 error control(error detection/correction)을 수행함.

network layer에서는 checksum field를 사용해 header에 대한 error detection을 수행하고, 전체 packet에 대한 control은 수행되지 않음. header는 각 router에서 수정되므로 error control도 각 router 및 host에서 수행됨.

물론 data-link layer에서도 error contorl을 수행하지만 이는 hop-to-hop에 대한 것으로, 각 router에서 발생할 수 있는 error를 방지하지는 못함.

2. Routing

Routing은 connectless service에서 packet forwarding을 위해 routing table에 대한 생성/유지/수정을 수행하는 작업으로, network layer에서 가장 중요한 issue 중 하나임.

3. Security
network layer는 기본적으로 security에 대한 처리를 하지 않도록 설계되었음. 하지만 connection-oriented service으로의 변경에 의해 IPSec이라는 새로운 가상 계층이 필요하게 되었음.

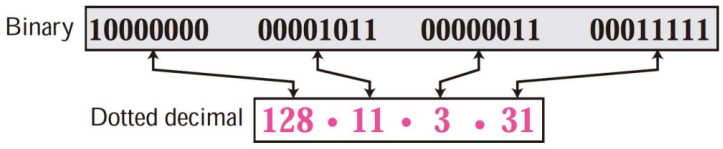
2.2. IP Address

여기에서의 IP address는 IPv4 address임.

2.2.1. IP Address

IP address는 internet에서 각 기기(host, router)를 식별하는 데에 사용되는 network layer에서의 address임. 이는 32비트 길이로, internet에서 unique(유일함.)하고 universal(보편적으로 인식됨.)하게 정의됨.

IP address의 표기법(notation)은 아래와 같이 8비트 단위로 끊어서 표기하는 것임. Binary Notation에서는 각 8비트를 이진수(base 2)로 단순히 끊어서 표기하고, Dotted-Decimal Notation에서는 각 8비트를 10진수(base 256)로 .으로 구분해 표기함.



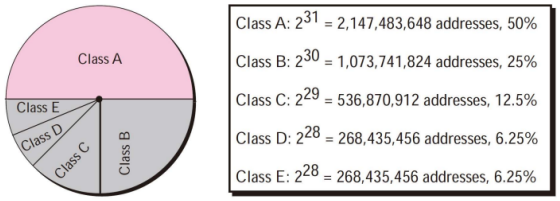
IP address는 NOT, AND, OR과 같은 bit-wise operation으로 조작할 수 있음.

IP address에 대한 addressing(주소 지정) 방식은 classful addressing, classless addressing이 있음.

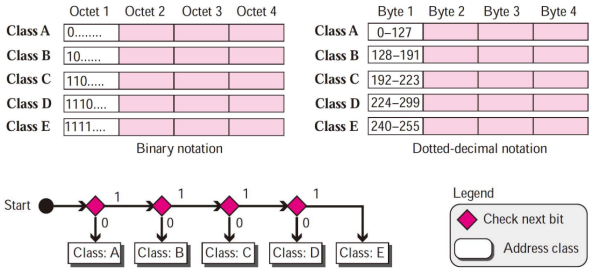
어떤 protocol에 대해 Address Space(주소 공간)는 해당 protocol에서 제공하는 전체 address의 집합임. IP address의 address space는 당연하게도 2^{32}개의 원소를 가짐.

2.2.2. Classful Addressing

1. Classful Addressing
Classful Addressing은 address space를 5개의 class(A, B, C, D, E)로 분류하고, block을 각 기관(organization)별로 할당하는 방식으로, two-level addressing을 구현함.



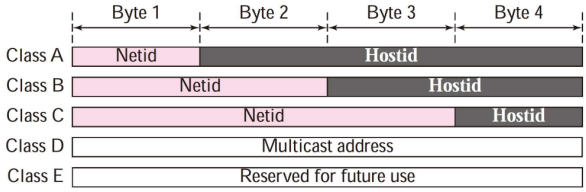
각 class는 맨 앞 4개의 비트로 구분됨. 이에 따라 맨 앞 비트 4개를 순차적으로 검사(0인지 1인지.)하여 class를 분류할 수 있음. dotted-decimal notation으로 보면 아래와 같이 10진수 값의 범위에 따라 class가 구분됨.



2. Class 별 Address Space

아래 그림과 같이 class A, B, C는 IP address를 서로 다른 크기의 Netid와 Hostid로 나누고, D와 E는 각각 multicast address와 추후 사용을 위한 address로 활용함.

Netid는 특정 기관이 가진 network에 대한 id이고, Hostid는 해당 network의 host들이 가지는 id임. 이때 Netid에 의해 구분되는 각 network의 주소 범위를 Block이라고 함. classful addressing에서 각 기관은 그 규모에 따라 class A/B/C의 block을 할당받아 IP address를 활용함.

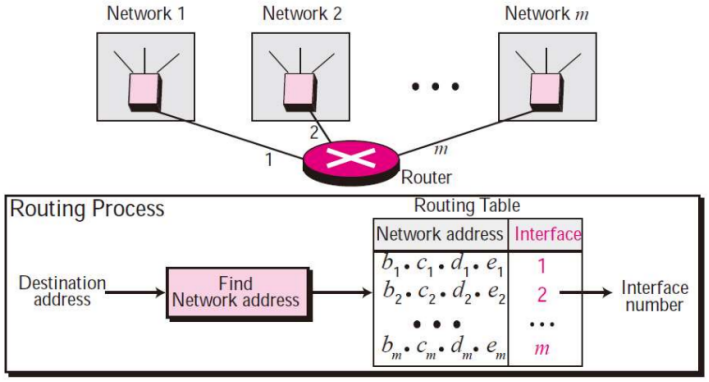


여기에서 IP address를 작성할 때는 아래와 같이 해당되는 class의 Netid 비트 수(n)를 /로 뒤에 명시함. 이는 classful addressing에서 일반적으로 사용되는 표기법은 아니지만, 여기에서는 이해를 위해 사용함.

180.8.0.1/16

network에서의 첫 번째 address(Hostid가 모두 0인 address)는 Network Address라고 하며 host에 할당되지 않음. 즉, network address는 해당 network에 대한 식별자임. router는 network address를 활용해 router table에서 수신받은 packet이 어떤 network로 전달되어야 하는지를 결정함.

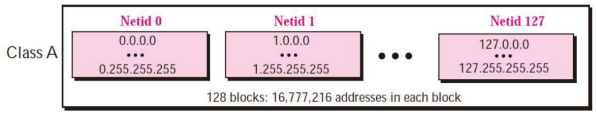
Network Mask는 class(A/B/C)별로 Netid를 추출하기 위해 존재하는 32비트 길이의 비트열임. Netid에 해당하는 부분은 1, Hostid에 해당하는 부분은 0임. 단순히 bit-wise AND로 Netid를 추출할 수 있음.



각 class별 block과 block의 크기는 아래와 같음.

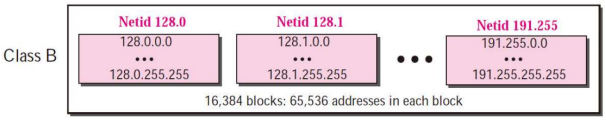
1) Class A

class A에서는 총 8비트가 Netid로 사용되는데, 맨 앞 1비트가 class 구분에 의해 0으로 고정되므로 총 $2^7 = 127$개의 block이 존재하고, 각 block은 $2^{24} = 16,777,216$개의 address를 포함함.

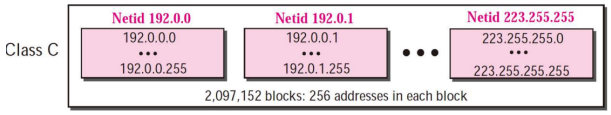


2) Class B

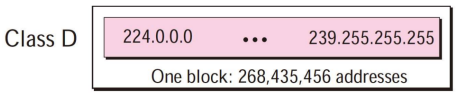
class B에서는 총 16비트가 Netid로 사용되는데, 맨 앞 2비트가 class 구분에 의해 10으로 고정되므로 총 $2^{14} = 16,384$개의 block이 존재하고, 각 block은 $2^{16} = 65,536$개의 address를 포함함.



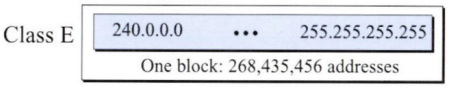
3) Class C
class C에서는 총 24비트가 Netid로 사용되는데, 맨 앞 3비트가 class 구분에 의해 110으로 고정되므로 총 $2^{21} = 2,097,152$개의 block이 존재하고, 각 block은 $2^8 = 256$개의 address를 포함함.



4) Class D
class D는 단일 block으로 구성되어 있으며, 각 address는 multicasting을 위한 host group을 정의하는 데에 사용됨. 즉, 해당 group의 host들은 기본(unicast) address에 추가로 이 multicast address를 가지게 되고, 이 address를 사용해 group 내 host들에게 multicast할 수 있음.



5) Class E
class E는 단일 block으로 구성되어 있으며, 각 address는 예약되어 일반적인 목적이 아닌 연구, 실험, protocol 개발 등에 사용됨.



classful addressing에서는 이와 같이 고정된 크기의 block을 기관에 할당하므로 address가 낭비되거나 (A, B) 부족할 수 있음(C). 이는 각각 subnetting과 supernetting으로 해결할 수 있음. 하지만 이 경우 routing이 복잡해지고, class 사용에 따른 낭비/부족을 원천적으로 해결하지는 못함.

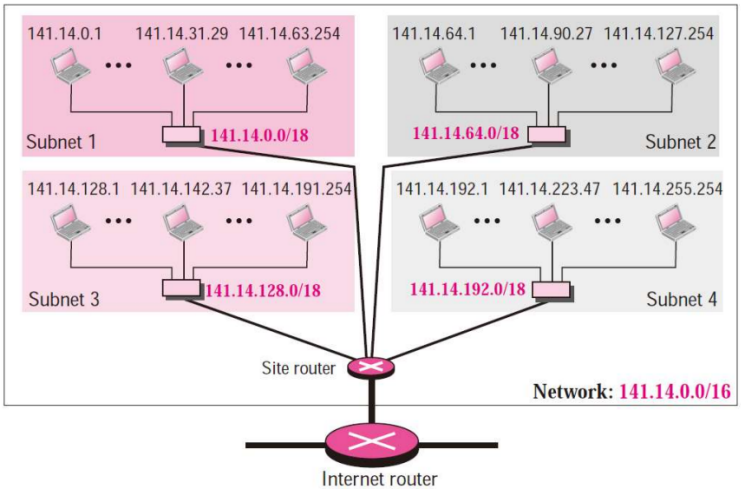
처음 IP가 제시되었을 때는 internet이 지금과 같은 정도로 활용될 것이라고 예상되지 못했음. 그래서 그냥 단순하게 이런 class로 나누어 각 기관에게 제공했음.

2.2.3. Classful Addressing : Subnetting and Supernetting

1. Subnetting
Subnetting은 너무 많은 address를 가지는 class A/B network를 여러 개의 작은 subnet으로 나눠서 활용하는 기법임. 앞에서 다룬 방식은 network와 host로 구성되는 2-level addressing이었다면, subnetting을 적용하면 subnetwork가 추가된 3-level addressing이 됨.

subnetting에서는 생성할 subnet의 개수에 대응되는 비트를 Netid 뒤에 추가로 사용하고, 내부적으로 Site Router(사설 라우터)를 사용하여 network를 나눔. Site는 특정 기관 등에 대해 존재하는 network을 의미함. 이때 기존 Netid(n)와 추가한 비트를 포함하는 비트열을 Subnetid(n_{sub})라고 하고, 그 길이는 아래와 같이 계산할 수 있음. s는 subnet의 개수로, 2의 거듭제곱임.

$$n_{sub} = n + \log_2 s$$



network address와 마찬가지로, subnet이 가지는 첫 번째 address(Hostid가 모두 0인 address)는 Subnet Address라고 함. 이는 site router에서 해당 packet이 어떤 subnet으로 송신되어야 하는지 결정하는 데 활용됨.

또한 network mask와 같이 Subnet Mask는 subnet 별로 Subnetid를 추출하기 위해 존재하는 32비트 길이의 비트열임. Subnetid에 해당하는 부분은 1, Hostid에 해당하는 부분은 0임. 단순히 bit-wise AND로 Subnetid를 추출할 수 있음.

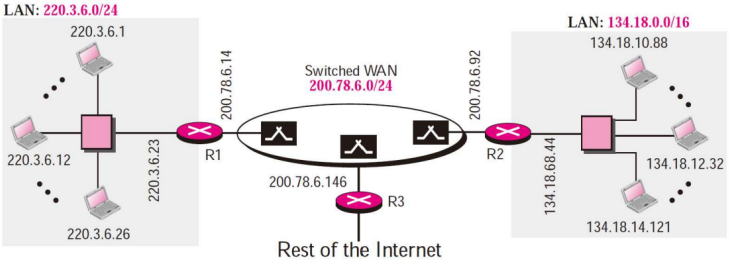
2. Supernetting
Supernetting은 너무 적은 address를 가지는 class C network 여러 개를 하나의 큰 supernet으로 묶어서 활용하는 기법임.

supernetting에서는 생성할 subnet의 개수에 대응되는 비트를 Hostid 앞에 추가로 사용함. 이때 기존 Netid(n)에서 비트를 덜 사용하게 된 비트열을 Supernetid(n_{super})라고 하고, 그 길이는 아래와 같이 계산할 수 있음. s는 subnet의 개수로, 2의 거듭제곱임.

$$n_{super} = n - \log_2 s$$

또한 network mask와 같이 Supernet Mask는 supernet 별로 Supernetid를 추출하기 위해 존재하는 32비트 길이의 비트열임. Supernetid에 해당하는 부분은 1, Hostid에 해당하는 부분은 0임. 단순히 bit-wise AND로 Supernetid를 추출할 수 있음.

아래와 같이 classful addressing을 활용해 internet이 구성될 수 있음. 여기에서 switched WAN은 각 network 사이에서 swtiching을 수행함.



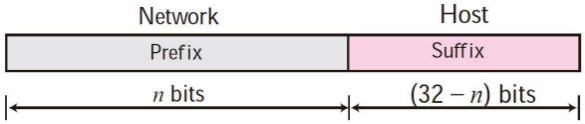
2.2.4. Classless Addressing

1. Classless Addressing
Classless Addressing은 임의의 길이를 가질 수 있는 prefix와 suffix로 IP address를 나누고, 이에 따라 CIDR을 활용하는 방식으로, two-level addressing을 구현함.

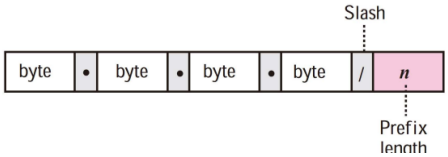
subnetting과 supernetting은 classful addressing에서의 address 낭비/부족 문제를 완화시킴. 하지만 결

과적으로 routing이 더 복잡해지고, class 사용에 따른 낭비/부족을 원천적으로 해결하지는 못함. 이를 해결하기 위해서는 IPv6와 같이 더 큰 address space를 가지는 방식을 제안할 수 있지만 이는 packet format의 수정이 필요하다는 문제가 있음. 반면 classless addressing은 이를 잘 해결할 수 있음.

classless addressing에서도 block 단위로 address를 할당함. 대신 아래와 같이 IP address를 Netid에 대응되는 Prefix와 Hostid에 대응되는 Suffix로 나눔. classful addressing에서는 class에 따라 Netid의 길이가 몇 가지로 고정되어 있었지만, classless addressing에서는 prefix의 길이로 0부터 32까지 모든 수가 가능함. 즉, network 크기에 따른 block을 할당할 수 있음.



prefix가 임의의 길이(n)를 가질 수 있으므로, 각 IP address에 대해 n을 명시해줘야 해당 address가 어떤 network에 속하는지 알 수 있음. classless addressing에서는 CIDR을 활용하여 이를 해결함. CIDR(Classless Interdomain Routing)은 아래와 같이 address 뒤에 slash notation(/)으로 구분하여 n을 명시하는 기법임.



classless addressing에서도 suffix가 모두 0인 address(첫 번째 address)는 Network Address로, network 식별에 활용되며 host에게 할당되지 않음.

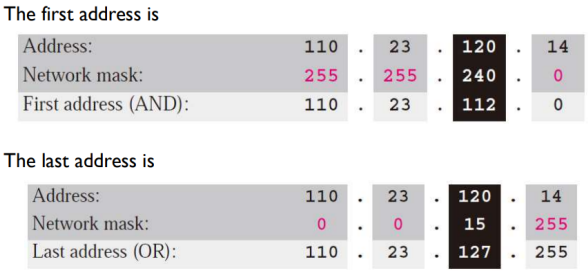
반면 suffix가 모두 1인 address(마지막 address)는 Direct Broadcast Address로, destination address로 지정하여 해당 prefix로 식별되는 network에 속한 모든 host에게 broadcast하는 데에 사용함.

2. Block 정보 추출

CIDR에 의해 prefix의 길이 n을 안다면 network mask(prefix 부분만 1인 32비트 비트열)를 구성할 수 있으므로, address 처리에 필요한 block의 모든 정보를 얻을 수 있음.

network address 또는 시작 주소는 임의의 한 address와 mask를 AND 연산하여 알 수 있고, network가 가진 전체 address의 개수(N)는 $N = 2^{32-n}$으로 알 수 있음. 또한 network의 마지막 address는 network address에 address의 개수를 더하는 식으로 구하거나, 단순히 mask에 NOT 연산을 적용한 뒤(prefix 부분만 1인 32비트 비트열) 임의의 address와 OR 연산하여 알 수 있음(전부 1이 되므로.).

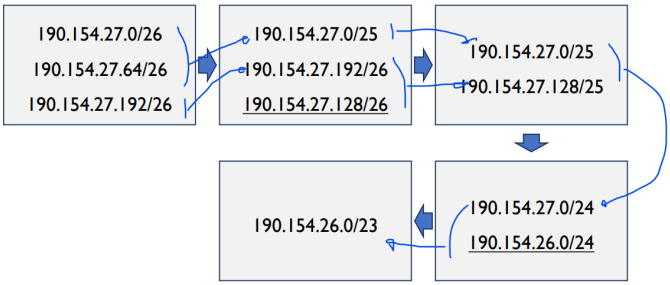
참고로, mask와 AND/OR 연산을 하는 경우 전체 address에 대해 연산하지 않아도, prefix와 suffix의 경계에 있는 바이트에 대해서만 비트 단위 연산을 적용하면 됨. 나머지 부분은 값이 유지되거나, 삭제되거나 둘 중 하나임.



3. Aggregation

classless addressing에서 block은 아래 그림과 같이 서로 합쳐질(aggregate) 수 있음. 이는 address에 대한 논리적인 결합이며, 뒤에서 aggregation에 따른 router 구성을 보면 이해할 수 있듯이 실제로 각 network가 하나로 합쳐지는 것은 아님.

이 그림에서 각 address는 network address임. 각 network address를 이진수로 나타내고, CIDR이 같은 두 block의 address space를 합칠 수 있는지 확인하면 쉽게 이해할 수 있음. CIDR로 나타낸 n을 k만큼 줄이려면, 2^k 개의 각 block이 존재해야 함.



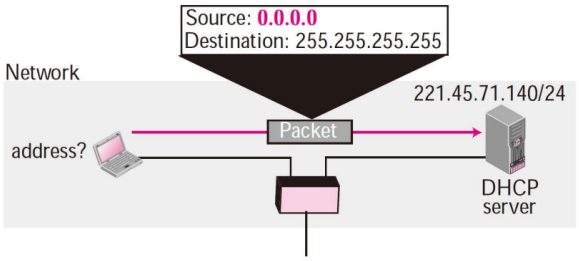
2.2.5. Classless Addressing : Special Blocks

classless addressing에서 block 중 일부는 아래와 같이 특수한 역할을 가지고 있음.

1. All-zeros Address

All-zeros Address는 말 그대로 모든 값이 0인 IP address(0.0.0.0/32)로, host가 자신의 IP address를 모를 때 사용함. n이 32이므로 하나의 단일 주소를 가지는 block임.

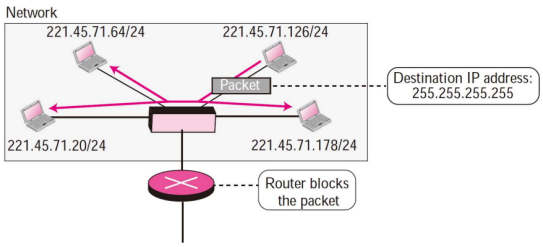
즉, 이는 bootstrap(internet 연결을 위한 부팅 과정)에서 사용되며, 해당 host는 source address를 all-zero address로 해서 bootstrap(DHCP) 서버에 전송함.



2. All-ones Address

All-ones Address는 말 그대로 모든 값이 1인 IP address(255.255.255.255/32)로, host가 포함된 local network에 대한 제한된 broadcast를 수행할 때 사용함. n이 32이므로 하나의 단일 주소를 가짐.

즉, network 내의 모든 host에게 전송하려는 경우 destination address를 all-ones address로 해서 전송함. router는 이런 packet을 수신받으면 block하여 local network에서만 broadcast되도록 함.

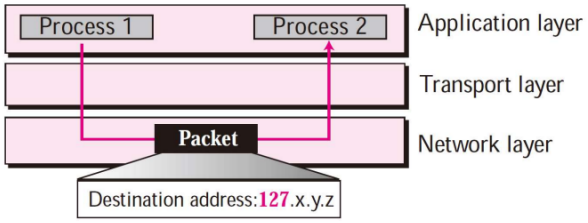


direct broadcast address에서도 broadcast를 수행하지만 이는 특정 network에 대한 broadcast이고, all-ones address는 local network에서의 broadcast에 사용됨.

3. Loopback Address

Loopback Address는 host에 설치된 소프트웨어에 대한 테스트 등에 사용되는 IP address(127.0.0.0/8)임. 즉, loopback address를 destination adress로 하는 packet은 외부로 전송되지 않고, network layer에서 해당 host로 다시 반환됨.

예를 들어, ping(network 연결이 제대로 이뤄지는지를 확인하는 application)에서는 동일한 host에 위치한 소프트웨어가 packet을 잘 받아서 처리하는지 확인하기 위해 이를 사용하고, client 프로그램에서는 동일한 host에 위치한 server 프로그램에 packet을 전송할 때 사용함.



4. Private Address

Private Address(사설 주소)는 global network에게 인식되지 않고, network 내부적으로만 사용이 가능한 address임. 다양한 형태의 block이 존재함.

이를 global network가 인식할 수 있도록 하려면 뒤에서 설명할 NAT 등을 추가로 활용해야 함.

참고로, block은 ICANN이라는 국제 관리 기구에서 ISP(Internet Service Provider)에서 할당하고, ISP는 해당 block을 subnetting하는 등의 처리를 적용한 뒤 host에게 IP address를 할당함.

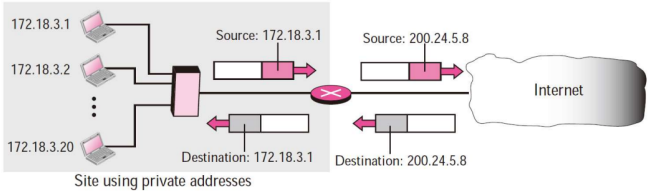
이 필기에서는 기본적으로 private ip가 아니라, public ip임을 가정하고 설명함.

2.2.6. NAT

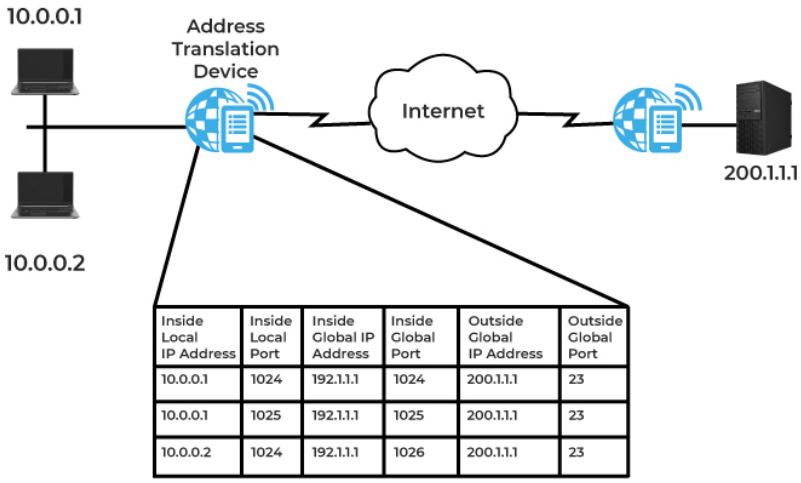
NAT(Network Address Translation)은 내부 통신을 위해 private address를 활용하고, 외부와의 통신은 private address를 global address로 변환해 수행하도록 한 기법임.

대부분의 작은 규모의 network에서는 일부 host들만이 internet으로의 동시 통신을 필요로 함. 이에 따라 각 host 모두에 global address가 할당될 필요가 없음. NAT의 활용에 따라 IPv4의 적은 address 개수에도 불구하고, IPv6의 필요성이 줄어들었음.

NAT에서는 NAT router가 translation table을 활용해 private address와 global address 간의 변환을 수행함. Translation Table은 private IP address와 그에 대응되는 public IP address로 구성된 table임. 이때 각 host 간의 many-to-many 관계를 저장해야 하므로 port number를 포함하도록 translation table을 확장함.



translation table은 private에 대한 정보와, 해당 private이 변환된 public에 대한 정보, 그리고 통신 중인 외부(external)에 대한 정보를 모두 포함하도록 구성됨.



Inside Local IP Address	Inside Local Port	Inside Global IP Address	Inside Global Port	Outside Global IP Address	Outside Global Port
10.0.0.1	1024	192.1.1.1	1024	200.1.1.1	23
10.0.0.1	1025	192.1.1.1	1025	200.1.1.1	23
10.0.0.2	1024	192.1.1.1	1026	200.1.1.1	23

교재에서 소개하는 translation table은 아래와 같이 구성됨. 여기에서 private address/port는 private host의 것이고, external address/port는 통신 중인 외부 host의 것임. 또한 외부에서 내부에 접근할 때 private port를 쓴다고 설명함. 즉, public port와 private port를 구분하지 않는데, 위에 정리한 내용이 더 정확하다고 판단했음.

교수님께서도 "이전에 배운 NAT는 baisc NAT였다. private IP/port, dest의 IP/port. 실제로는 여기에 추가로 NAT router의 IP/port도 존재한다"라고 언급하셨음.

Private Address	*Private Port*	*External Address*	*External Port*	*Transport Protocol*
172.18.3.1	1400	25.8.3.2	80	TCP
172.18.3.2	1401	25.8.3.2	80	TCP
…	…	…	…	…

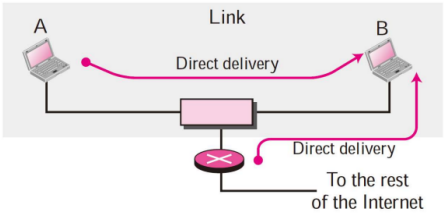
2.3. Delivery/Forwarding

IP address를 활용한 network layer에서의 delivery와 forwarding에 대해 알아보자.

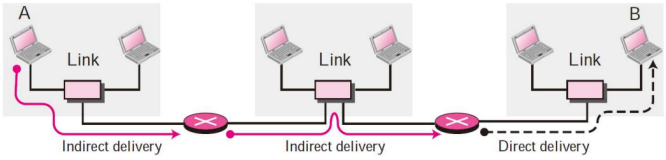
2.3.1. Delivery

Delivery(전달)는 하위 layer들에 대한 handling을 통해 packet을 source로부터 destination으로 전송하는 network layer에서의 과정임. delivery는 아래와 같이 두 가지로 구분될 수 있음.

1) Direct Delivery
Driect Delivery는 destination이 동일한 physical network로 연결되어 있는 상황에서의 delivery임. 즉, source가 destination, 또는 마지막 router와 destination이 동일한 physical network에 위치한 경우임.



2) Indirect Delivery
Driect Delivery는 destination이 동일한 physical network로 연결되어 있지 않은 상황에서의 delivery임. 즉, 하나 이상의 router를 거쳐야 destination에 도달하게 되고, 이에 따라 IP address와 routing table를 활용해 다음 router로 이동해야 함.



2.3.2. Forwarding

Forwarding은 packet을 다음 hop(destination/router 등)으로 전달하는 것임.

connectless service에서는 destination address 기반의 forwarding이, connection-oriented service에서는 label 기반의 forwarding이 수행됨.

2.3.3. Destination Address Based Forwarding : Concept

Destination Address 기반의 forwarding은 말 그대로 destination address에 대한 routing table을 활용해 forwarding하는 방식임. 이는 connectless service의 기법으로, 가장 많이 사용되는 전통적 방법임.

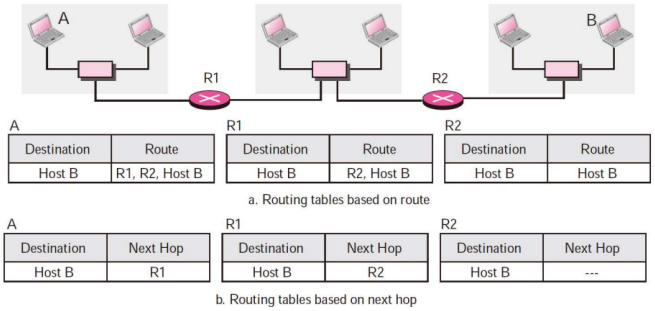
Internet과 같은 큰 규모의 internet에서는 routing table이 많은 entry를 가져 그 크기가 커지게 되는데, 이 경우 단순히 table을 활용하는 데에서 그친다면 table lookup 과정이 비효율적임. 아래의 기법들을 활용하여 table을 크기를 충분히 작게 만들 수 있음. 여기에서 entry란 table에서 각 행(레코드)를 말함.

router에 대한 성능에 있어서는 table의 크기를 줄이는 것 외에도 table을 update하는 방법, searching을 수행하는 방법도 중요한데, 이는 뒤의 router 부분에서 다룸.

1. Next-hop Method

Next-hop Method는 destination까지의 전체 route(경로)가 아닌 바로 다음 hop에 대한 address만을 저장하는 기법임. 이에 따라 각 entry가 가지는 데이터를 줄일 수 있음.

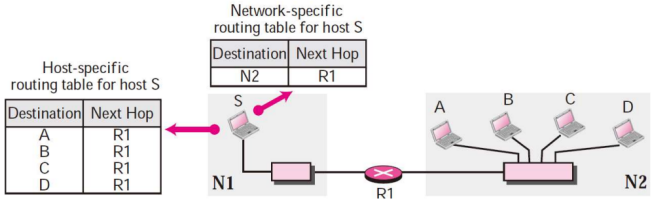
routing table은 기본적으로 net-hop method를 활용함.



2. Network-specific Method

Network-specific Method는 각 destination host별로 entry를 생성하는 대신, 해당 network에 대한 entry를 생성하는 기법임. 동일한 network에 속한 host들을 하나의 entry로 처리하므로 table을 줄일 수 있음.

뒤에서 설명할 routing 기법들은 network-specific method를 전제로 함.

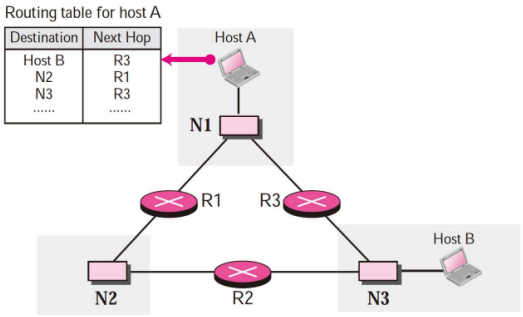


3. Host-specific Method

Host-specific Method는 각 destination host별로 entry를 생성하는 기법임.

이는 network-specific method에 비해 더 큰 table을 사용하게 되지만, 관리자가 각 host에 따른 제어를

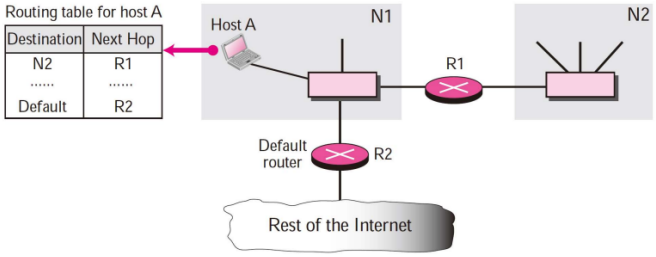
수행할 수 있다는 이점이 있음. 예를 들어, 어떤 host로의 packet이 특정 router를 거치거나 거치지 않도록 table을 지정할 수 있음. 즉, 보안성을 제공하거나 경로를 점검하기 위한 경우에 사용하는 기법임.



4. Default Method

Default Method는 destination address가 table에 존재하지 않는 경우, default로 설정된 hop으로 forwarding하는 기법임.

당연하게도 internet에 존재하는 모든 대상에 대한 정보를 table에 저장할 수는 없으므로, 모르는 대상은 default로 처리함.



2.3.4. Destination Address Based Forwarding : Classful Addressing

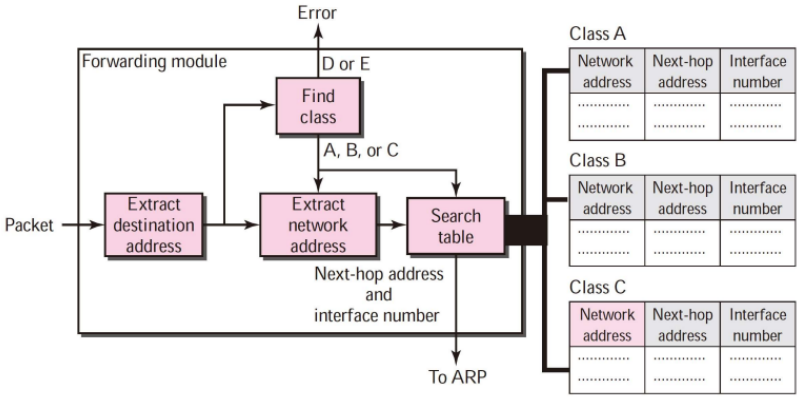
Classful Addressing에서의 Destination Address 기반 forwarding을 살펴보자. classful addressing에서는 address에 대한 낭비/부족이 발생할 수 있지만, forwarding에서의 단순함을 제공할 수 있음.

1. Subnetting이 없는 경우

subnetting은 기관이 소유한 network 내부에서 수행되고, internet에 존재하는 대부분의 router는 관여하지 않음. 이런 대부분의 router들은 class(A/B/C) 별로 table을 하나씩 사용하도록 설계됨. 물론 multicasting을 지원하려면 class D에 대한 table도 별도로 필요한데 여기에서는 다루지 않음.

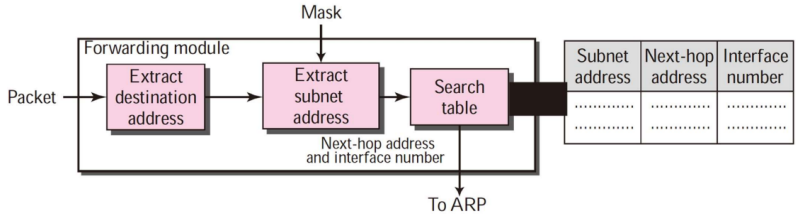
class에 대한 각 table은 network address(destination), next-hop address, interface number(router의 outgoing port)를 포함함.

forwarding module에서는 아래 그림와 같은 작업이 수행됨.
1) packet에서 destination address를 추출함.
2) class를 찾음.
3) destination address로부터 network address를 추출하여 적절한 class의 table을 활용해 결과를 얻음.



2. Subnetting이 있는 경우

subnetting은 기관이 소유한 network의 내부의 있는 router에서 수행됨. 아래와 같이 고정 길이를 가지는 subnetting의 경우 단순히 고정 길이의 mask(n_{sub})와 하나의 table을 사용해 forwarding할 수 있음.



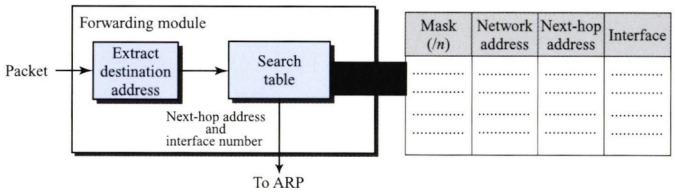
물론 가변 길이의 subnetting의 경우 여러 mask와 table을 사용해야 함.

참고로, network에서 interface는 장치가 network에 물리적 또는 논리적으로 연결되는 지점을 의미함.

2.3.5. Destination Address Based Forwarding : Classless Addressing

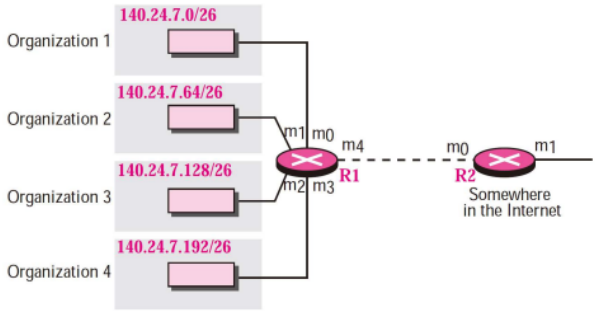
Classless Addressing에서의 Destination Address 기반 forwarding을 살펴보자.

classless addressing을 사용하는 경우 아래와 같이 mask(n)에 따라 network가 구분되므로 routing table은 추가로 mask(n) 정보를 포함하게 됨. 즉, classless에 비해 table의 크기가 커짐. 이에 따라 address aggregation 등을 활용함.



1. Aggregation

앞에서 다룬 것처럼, classless addressing에서는 aggregation을 통해 여러 block을 하나로 합칠 수 있음. 이는 아래 그림과 같이 실제로 network를 합치는 것이 아니라, 특정 router에서 하나의 address로 여러 network들에 대한 forwarding을 수행할 수 있도록 해서 table 크기를 줄이는 것임.



Mask	Network address	Next-hop address	Interface
/26	140.24.7.0	----------	m0
/26	140.24.7.64	----------	m1
/26	140.24.7.128	----------	m2
/26	140.24.7.192	----------	m3
/0	0.0.0.0	default router	m4

Routing table for R1

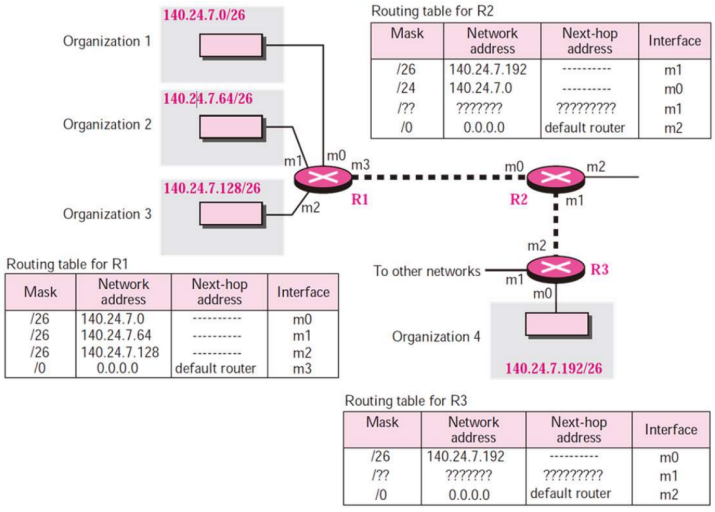
Mask	Network address	Next-hop address	Interface
/24	140.24.7.0	----------	m0
/0	0.0.0.0	default router	m1

Routing table for R2

2. Longest Mask Matching

Longest Mask Matching은 classless addressing에서 table이 mask의 길이(n)에 따라 내림차순으로 정렬되어 있고, destination address에 해당되는 entry가 여러 개라면 mask의 길이가 가장 긴 것을 선택하도록 하는 기법임. 즉, 긴 mask에 대한 entry가 우선적으로 선택되어 forwarding됨.

이에 따라 aggregation이 적용되어 있는 경우에도 더 구체적인(prefix가 더 긴) network부터 선택하여 forwarding할 수 있음. 예를 들어, 아래의 그림과 같이 internet이 구성된 경우 더 짧은 mask의 entry로 forwarding한다면, organization 4로 가야 하는 packet이 잘못 전달되게 됨.



Routing table for R2

Mask	Network address	Next-hop address	Interface
/26	140.24.7.192	----------	m1
/24	140.24.7.0	----------	m0
/??	???????	?????????	m1
/0	0.0.0.0	default router	m2

Routing table for R1

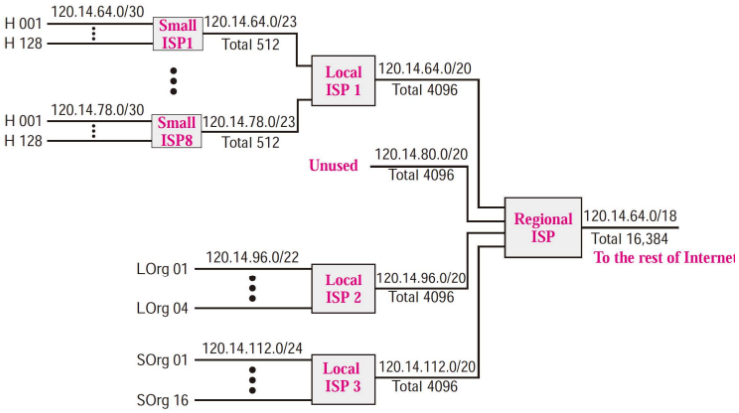
Mask	Network address	Next-hop address	Interface
/26	140.24.7.0	----------	m0
/26	140.24.7.64	----------	m1
/26	140.24.7.128	----------	m2
/0	0.0.0.0	default router	m3

Routing table for R3

Mask	Network address	Next-hop address	Interface
/26	140.24.7.192	----------	m0
/??	???????	?????????	m1
/0	0.0.0.0	default router	m2

3. Hierarchical Routing

Hiearchical Routing은 routing table의 크기를 줄이기 위해 aggregation으로 router에 대한 계층적 구조를 적용한 것을 말함. 즉, 각 router가 모든 network에 대한 address를 가지고 있는 대신, 일부 network들로 forwarding을 수행하는 하나의 rotuer에 대한 address를 가지고 있도록 함.



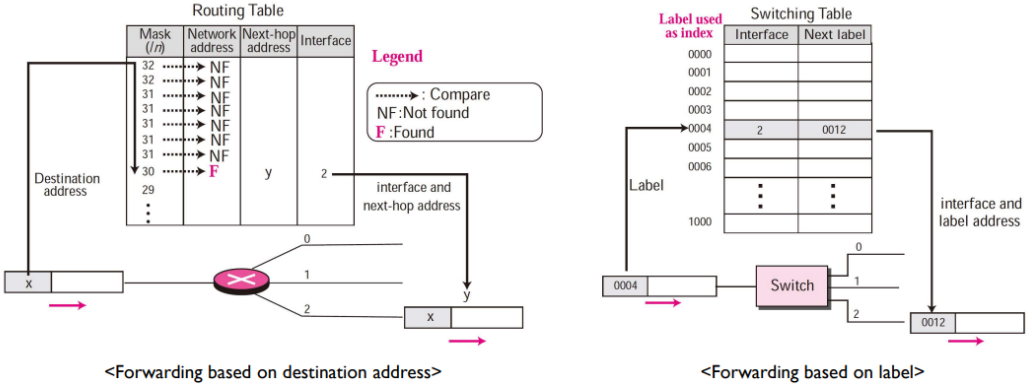
또한 이를 확장한 Geographical Routing은 지리적인 위치를 고려하여 아메리카, 유럽, 아시아 등의 지역별로 큰 block을 할당하고 hierarchical routing을 적용한 것임. 즉, 특정 지역의 router는 다른 지역에 대해 하나의 entry만 가지고 있으면 됨.

2.3.6. Label Based Forwarding

1. Label Based Forwarding

Label 기반의 forwarding은 말 그대로 packet에 포함된 label에 따라 forwarding하는 방식으로, routing을 switching으로 대체함. 이는 connection-oriented service(virtual-circuit approach)에서의 기법임.

destination address 기반의 forwarding에서는 routing table을 특정 알고리즘에 따라 확인하는 searching(탐색)을 수행했지만, label 기반의 forwarding에서는 label에 따라 switching table의 특정 값을 얻는 accessing(접근)을 수행함.



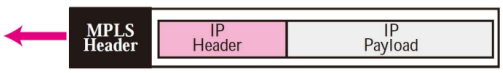
<Forwarding based on destination address> <Forwarding based on label>

2. MPLS

MPLS(Multiprotocol Label Switching)은 label 기반 forwarding을 구현할 수 있도록 하는 표준 protocol임. 이는 network layer와 data-link layer의 중간 layer로 간주됨.

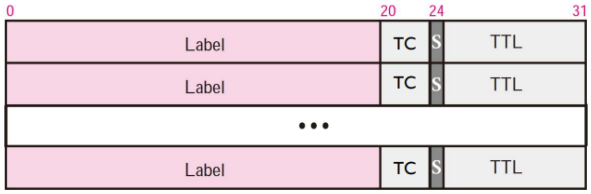
MPLS에 의해 정의되는 MPLS router는 destination address를 활용한 routing을 수행하는 router로서의 기능과, label을 활용한 switching을 수행하는 swtich로서의 기능을 모두 수행함.

MPLS에서는 IP packet을 payload로 하고, 앞에 MPLS header를 추가하는 encapsulation을 적용함.



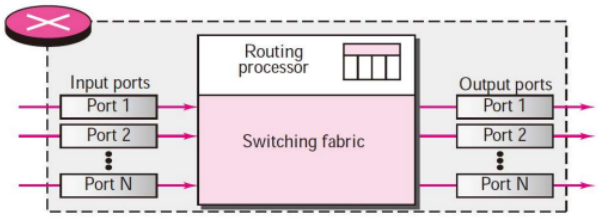
MPLS header는 구체적으로 아래와 같이 구성됨. 이는 32비트 길이의 subheader 스택으로 구성되며, 이를 활용하여 계층적인 switching이 가능함. label은 MPLS router의 routing table에서의 인덱싱을 위한 값이고, TC(Traffic Class)는 QoS와 ECN을 위한 값, S는 subheader의 situation을 정의하는 값임

(1이면 해당 subheader가 가장 마지막 subheader인 것.).



2.3.7. Router의 구조

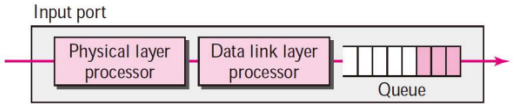
router는 아래와 같이 input port, output port, routing processor, switching fabric으로 구성됨.



1. Input Port

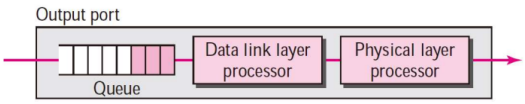
Input Port는 router에 대한 physical/data-link layer 기능을 수행함. 즉, 수신받은 signal로부터 비트열 구성, frame에 대한 decapulation, error detection/correction을 수행함.

input port는 buffer(queue)를 사용해 packet을 저장해 두고, 이를 switching fabric으로 넘김.



2. Output Port

Output Port는 input port의 반대 작업을 수행함.



3. Routing Processor

Routing Processor는 rotuer에 대한 network layer 기능을 수행함. 즉, destination address를 활용해 routing table에 대한 table lookup을 수행하여, next-hop address와 output port number를 얻음.

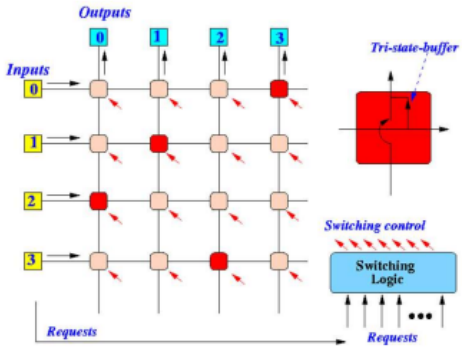
4. Switching Fabric

Switching Fabric(교환 조직)은 packet을 입력 queue에서 출력 queue로 이동시키는 부분임. 이는 packet delivery에 대한 전체 지연 시간에 큰 영향을 끼침.

과거 router는 해당 기능을 수행하는 컴퓨터였어서, 메모리 등의 자원이 switching fabric으로 사용되었지만, 현재는 아래와 같이 다양한 종류의 switching fabric이 있음.

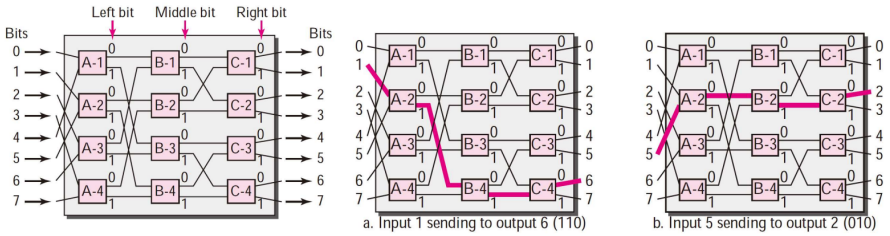
1) Crossbar Switch

Crossbar switch는 n개의 input port를 n개의 output port로 연결하는 switching fabric으로, 각 cross-point에서 microswitch를 사용함.



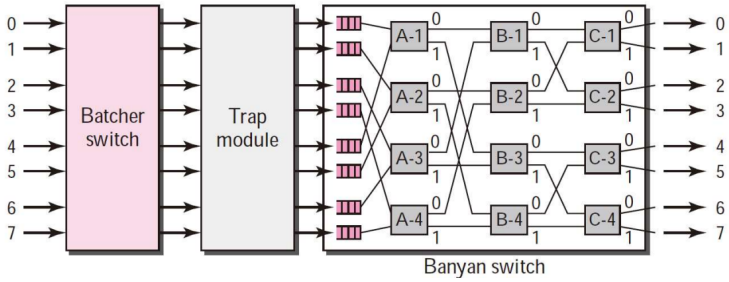
2) Banyan Switch
Banyan Switch는 n개의 input port를 n개의 output port로 연결하는 switching fabric으로, n/2개의 microswitch를 가진 $\log_2 n$개의 stage로 구성됨. output port number를 이진수로 표현했을 때의 각 비트를 순서대로 stage에서 사용하여 switching함.

crossbar switch에 비해 더 현실적인 방법임.



3) Batcher-Banyan Switch
Batcher-Banyan Switch는 banyan switch에 Batcher Switch를 수신받은 packet들을 output port에 따라 정렬하는 swtiching fabric임.

banyan switch는 동일한 port로 가지 않는 packet들끼리도 collision이 발생할 수 있다는 문제점이 존재하는데, batcher-banyan switch에서는 이를 개선함. 자세한 내용은 다루지 않음.



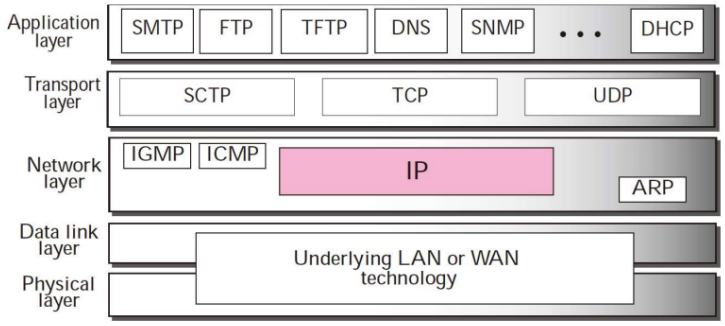
2.4. IPv4

IPv4(Internet Protocol Version 4)를 알아보자.

2.4.1. IP

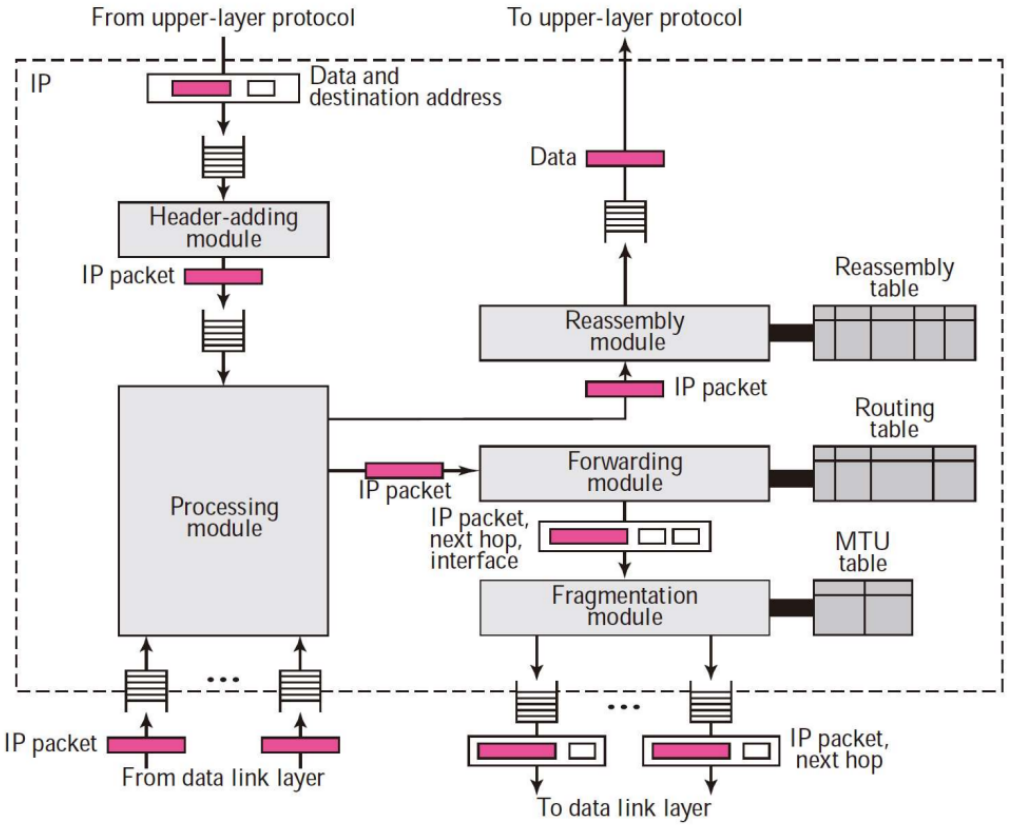
1. IP
IP(Internet Protocol)은 TCP/IP의 network layer에 속하는 protocol임.

IP는 unreliable하고 connectionless한 datagram protocol로, best-effort delivery service임. 여기에서 best-effort는 IP packet이 corrupt되거나, 손실되거나, 순서가 바뀌어 전달되거나, 지연되거나, network의 혼잡도를 높일 수 있음을 의미함. 대신 IP는 상위 layer의 protocol에게 관련된 책임을 넘김.



2. IP Package

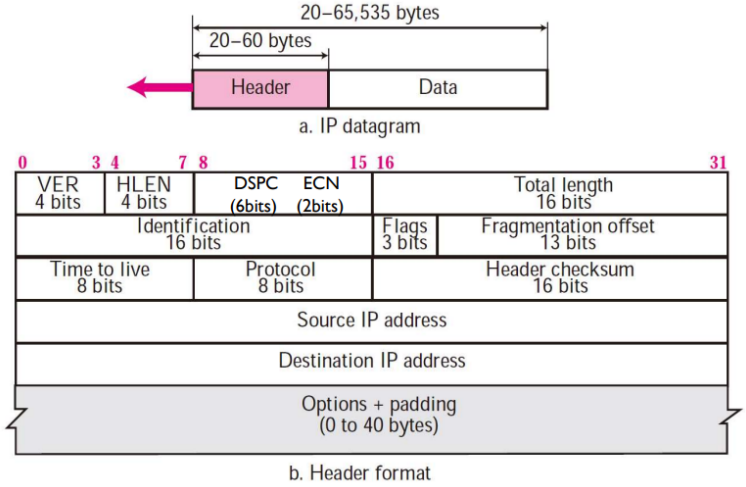
IP Package는 IP에 해당되는 기능을 구현하는 데에 필요한 소프트웨어 모듈의 집합임. queue 등을 생략하고 간단히 나타내면 아래와 같음.



2.4.2. Datagram

Datagram은 network layer에서의 packet을 말함.

Datagram은 가변 길이를 가지며, header와 data로 구성됨. header는 20에서 60바이트 크기로 routing과 delivery를 위한 정보를 담은 field들을 포함함. 이때 option을 제외한 부분은 고정 부분으로 20바이트이고, option은 가변 부분으로 0에서 40바이트까지의 크기를 가질 수 있음.



datagram header의 각 field가 의미하는 바를 알아보자.

1) VER(Version)
: IP의 protocol 버전을 나타내는 field. 해당 datagram을 처리하는 IP software에게 버전(4 또는 6)을 알려주고, 버전이 호환되지 않으면 해당 datagram을 discard함.

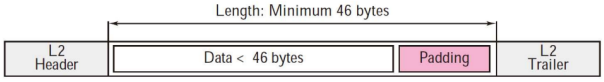
2) HLEN(Header length)
: datagram header의 길이를 4byte 크기의 word 단위로 나타내는 field. header의 길이는 가변적이므로 이를 명시해 줘야 함. option이 없다면 길이는 20바이트로 HLEN 값은 5이고, option이 최대 크기로 지정되어 있다면 길이는 60바이트로 HLEN 값은 15임.

3) DSCP(Differentiated Services Code Point)
: 원래는 해당 datagram이 어떻게 처리되어야 하는지에 대한 ToS(Type of Service)를 나타내는 field이지만, 현재는 DiffServ(Differentiated Service)를 나타냄. 실시간 통신에 활용된다고 함.

4) ECN(Explict Congestion Notification)
: network congestion(혼잡도) 확인에 사용되는 부가적인 field임.

5) Total length
: datagram의 header와 data를 모두 포함하는 전체 길이를 byte 단위로 나타내는 field. 해당 field는 16비트 크기이므로, datagram의 최대 total length는 $2^{16} - 1 = 65,535$바이트임. 또한 total length에서 HLEN을 빼면(물론 word 단위이므로 4를 곱해줘야 함.) data의 길이도 알 수 있음.

data-link layer에서는 protocol의 종류에 따라 frame의 크기가 제한됨. ethernet의 경우 frame에 encapsulation될 데이터의 크기는 46에서 1500바이트 사이여야 함. datagram이 그것보다 크다면 아래에서 설명할 fragmentation을 적용해야 하고, 그것보다 작다면 padding을 추가해야 함. padding을 추가하는 경우 decapsulation 시에 total length를 활용해 어느 지점부터 padding인지를 알 수 있음.



6) Identification/Flags/Fragmentation offset
: fragmentation에서 사용함.

7) Time to live
: 해당 datagram이 앞으로 거칠 수 있는 최대 hop(router)의 개수를 나타내는 field. 각 router에 도착할 때마다 이 값을 1씩 줄이고, 도착했을 때 값이 0이라면 router는 해당 datagram을 discard함.

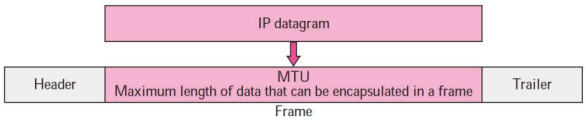
8) Protocol
: 해당 datagram에 대해 IP의 service를 활용하는 상위 layer의 protocol(TCP, UDP 등)을 나타내는 field.

9) Checksum
: error detection을 위한 checksum을 저장하는 field.

10) Source/Destination Address
: 32비트 길이의 source/destination address를 저장하는 field. source에서 destination으로 가는 동안 변경되지 않음.

2.4.3. Fragmentation

1. Fragmentation
Fragmentation은 해당 data-link layer protocol의 MTU보다 큰 datagram(total length)에 대해, data field를 나눠 더 작은 여러 개의 datagram을 생성하는 기법임. 이때 MTU(Maximum Transfer Unit)는 data-link layer protocol에서 frame이 가질 수 있는 data field(datagram의 total length)의 최대 크기임.



router에서 전달받은 frame은 이전 link에서의 protocol에 따라 encapsulation되어 있고, 새로 전달할 frame은 사용할 link에서의 protocol에 따라 encapsulation됨. 즉, 각 router에서는 추출한 datagram의 크기보다 더 작은 MTU를 만나면 fragmentation을 수행해야함.

이때 fragmentation이 적용된 datagram들에 대한 reassembly는 destination host에서만 한꺼번에 수행됨. 각 datagram은 독립적인 경로를 통해 전송되므로 이는 당연함.

2. Fragmentation 관련 Field
fragmentation에서 datagram을 나눌 때 header는 전체 field 값을 복사해 구성하는데, flags, fragmentation offset, totoal length는 수정함. 또한 checksum은 수정한 field들에 의해 다시 계산됨.

destination에서의 reassembly를 위해서는 identifiacation, flags, fragmentation offset field를 활용함.

1) Identification
Identification은 source host에 의해 결정되어 datagram을 식별하는 16비트 길이의 field임. 즉, datagram은 source address와 identification으로 유일하게 식별될 수 있음.

source host는 양의 정수로 초기화한 Counter를 사용해 identification 값을 지정하고, 지정한 뒤에는 counter 값을 1 증가시킴.

identification은 fragmentation 시에 orignal datagram과 동일한 값을 갖도록 복제되어, destination에서 reassembly 시에 datagram들을 식별할 수 있음.

2) Flags
Flags는 fragmentation과 관련된 flag를 저장하는 3비트 길이의 field임.

첫 번째 비트는 사용되지 않음(reserved).

두 번째 비트는 do not fragment(D) 비트임. 이 값이 1이면 fragmentation되지 않고, MTU를 초과하는 경우 해당 datagram을 discard하고 ICMP error message를 source host에 전송함. 이 값이 0이면 필요한 경우 fragmentation함.

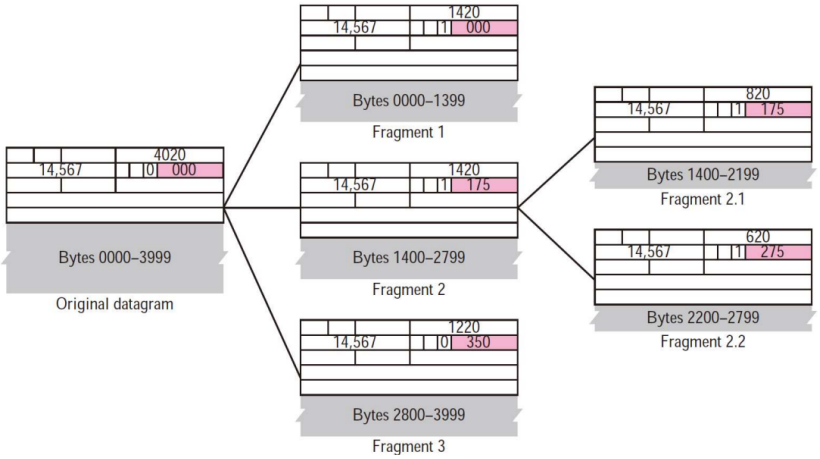
세 번째 비트는 more fragment(M) 비트임. 이 값이 1이면 해당 datagram이 마지막 fragment가 아닌 것이고, 0이면 마지막 fragment인 것임.

3) Fragmentation Offset
Fragmentation Offset은 전체 datagram에서 해당 fragment의 상대적 위치를 8바이트 단위로 나타내는 13비트 길이의 field임. 즉, 전체 datagram의 시작 지점으로부터 해당 fragment의 시작 지점이 얼마나 떨어져 있는지를 나타냄.

datagram의 total length가 16비트를 사용해 바이트 단위로 표현하므로, 13비트를 사용해 8바이트 단위로 표현하면 정확히 전부를 나타낼 수 있음. 또한 8바이트 단위로 표현하므로 각 fragment가 가지는 첫 번째 바이트의 주소는 8의 배수여야 함.

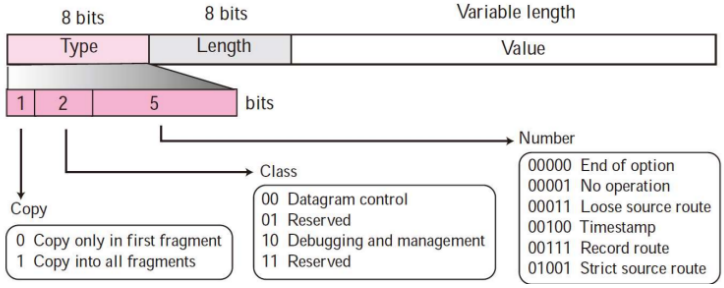
fragmentation이 여러 번 수행되더라도, offset은 전체 datagram의 시작 지점으로부터 계산됨.

예를 들어 아래와 같이 4000바이트 크기의 datagram을 총 5개의 fragment로 나누는 경우를 생각할 수 있음.



2.4.4. Option : Concept

Option은 header에서 테스트 또는 디버깅을 위해 사용되는 부분임. 이는 아래와 같이 1바이트의 type field, 1바이트의 length field, 가변 길이의 value field로 구성된 format을 가짐.



1. Type
Type field는 copy, class, number로 나뉨.

1) copy는 fragmentation에서 option을 복사할 것인지를 지정하는 1비트 크기의 subfield임. 0이면 첫 번째 fragment에만 복사되고, 1이면 전체 fragment에 복사됨.

2) class는 option의 일반적인 목적을 지정하는 2비트 크기의 subfield임. 00이면 datagram contorl에 사용한다는 것이고, 10이면 디버깅/관리에 사용한다는 것임. 01과 11은 정의되지 않음(reserved).

3) number는 option의 type을 지정하는 5비트 크기의 subfield임. 5비트이므로 총 32개의 type을 정의할 수 있지만, 6개의 type만 정의되어 있고 나머지는 정의되지 않음(reserved).

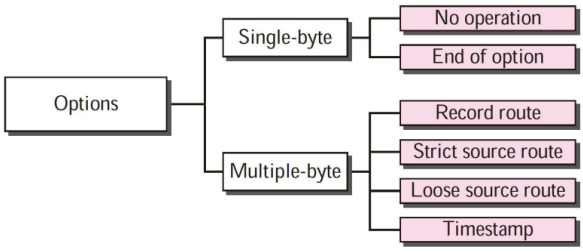
2. Length
Length field는 type/length field를 포함한 option의 전체 길이임.

3. Value
Value field는 특정 option이 필요로 하는 데이터를 저장함.

2.4.5. Option : Types

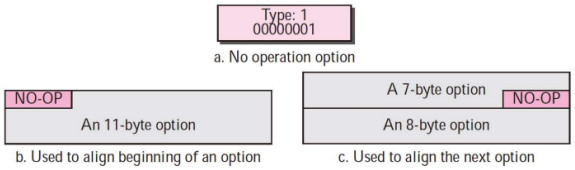
아래와 같이 6가지 option type들이 있음. 사용하는 바이트 수에 따라 single byte와 multiple byte로 나뉨. single byte는 length/value field를 필요로 하지 않고, multiple byte는 length/value field를 필요로 함.

이때 하나의 datagram에서 각 option은 40바이트 내에서 동시에 존재할 수 있음.



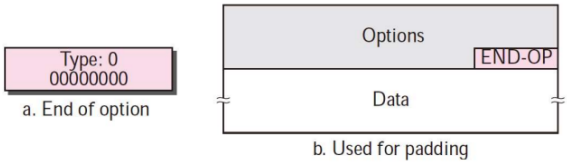
1. No-Operation Option

No-Operation Option은 option 사이에서 빈 공간을 채우거나 align하기 위해 사용되는 1바이트 크기의 option임.



2. End-of-Option Option

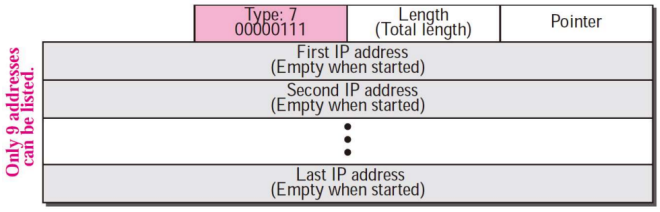
End-of-Option Option은 header에서 가장 마지막 option의 가장 마지막 부분에 padding으로 사용되어 마지막임을 나타내는 1바이트 크기의 option임. 이 option 뒤에는 payload data가 위치함.



3. Record-Route Option

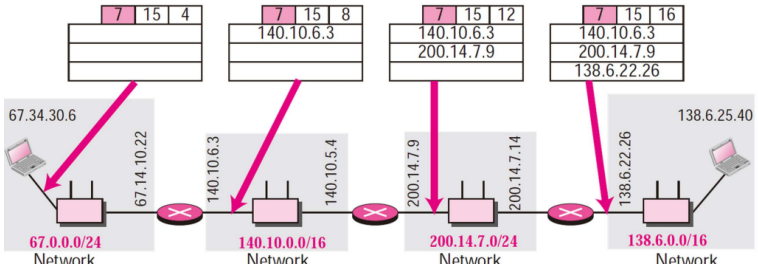
Record-Route Option은 해당 datagram이 거친 router들의 정보(IP address)를 저장하는 option임.

아래와 같이 field가 구성됨. pointer field는 router IP address가 다음으로 저장될 위치에 대한 offset을 나타내는 정수 값을 가짐.



구체적으로는, 해당 option이 존재하면 source host는 router 정보를 저장하기 위한 빈 value 공간을 확보해둠. 이때 pointer field 값은 4임(type/length/pointer가 존재하므로.). 이후 router에 도달했을 때 빈 공간이 존재하면(length와 비교하여 알 수 있음.) IP address를 pointer field가 가리키는 위치에 넣고 pointer 값을 4 증가시킴. ***이때 아래 그림과 같이, 넣는 IP address는 router의 IP address 중 새로 진입하는 network interface에 대한 IP address임.***

option은 최대 40바이트이고, type, length가 각각 1바이트, pointer가 1바이트, IP address가 4바이트이므로 하나의 datagram에서 총 9개의 router 정보를 저장할 수 있음.

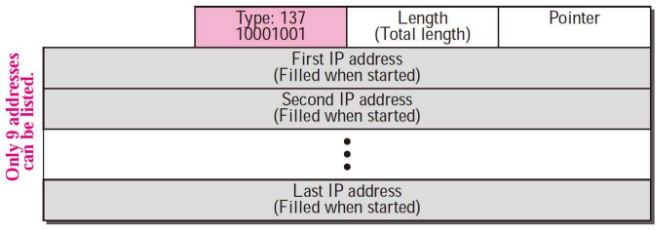


4. Strict-Source-Route Option

Strict-Source-Route Option은 datagram이 거쳐야 하는 router의 IP address를 직접 지정하는 option임. 더 신뢰성 있는 경로를 선택하거나 특별한 서비스를 제공할 때 사용함.

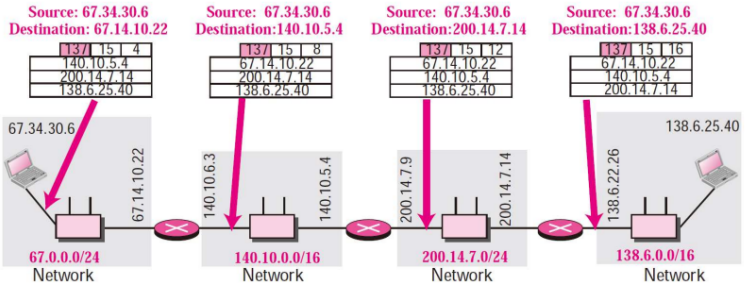
해당 datagram은 지정된 IP address의 router만 순차적으로 방문할 수 있고, 지정되지 않은 router에 방문하면 discard되고 error message가 전송됨.

아래와 같이 record-route option과 유사하게 field가 구성됨. pointer field는 해당 datagram이 다음으로 방문해야 하는 router IP address를 가리키는 offset 정수 값을 가짐.



구체적으로는, source host에서는 value에 각 router 정보를 저장하고, pointer 값을 4로 지정함. ***이후 router에 도달했을 때 pointer가 가리키는 IP address가 router의 IP address 중 진입한 interface의 IP address와 같다면 datagram을 처리하고, destination address와 pointer가 가리키는 위치의 address를 바꾼 뒤****, pointer 값을 4 증가시킴. 같지 않다면 discard함. 이에 따라 모든 router를 이미 방문해 pointer가 length를 넘어가서 더 이상 IP address가 존재하지 않는 경우에도 discard됨.*

여기에서도 마찬가지로 최대 9개의 router IP address를 지정할 수 있음.

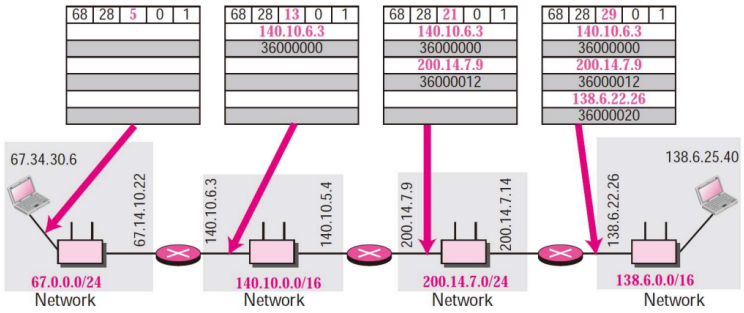


5. Loose-Source-Route Option

Loose-Source-Route Option은 Strict-Source-Route Option과 동일한데, 지정한 router들을 반드시 방문해야 하지만 중간에 다른 router도 방문할 수 있는 option임.

6. Timestamp

Timestamp는 datagram이 각 router에서 처리된 시간을 기록하는 option임. 이때 시간은 ms 단위로 UTC를 기준으로 설정됨. 시간 정보를 활용하면 router 사이의 시간 간격을 유추할 수 있음.



2.4.6. Checksum

TCP/IP에서는 error detection 기법으로 checksum을 사용함. 이를 통해 packet의 전송 도중에 발생하는 corruption을 detect함. ***이때 IP는 n을 16bit로 하고, header에 대해서만 checksum을 계산함.***

1. Checksum

Checksum은 각 section을 활용한 연산으로 계산한 checksum을 packet에 추가로 포함시켜 전송하는 error-detecting 기법임. 이는 임의의 길이를 가지는 packet에 대해 적용이 가능함. checksum은 data-link layer에서도 사용되고, network/transport layer에서 주로 사용됨.

Section은 packet을 n비트씩 묶은 것을 말함. Checksum은 각 section에 대한 연산으로 계산되는 길이 n의 비트열임.

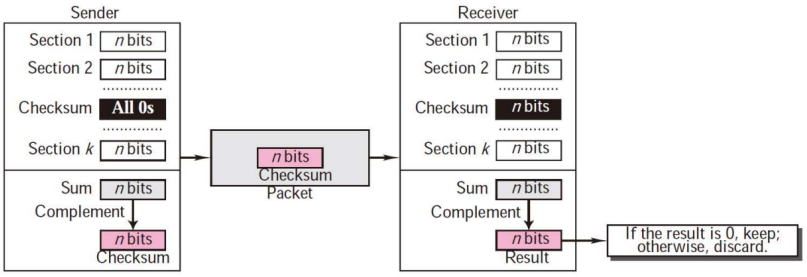
checksum은 section을 모두 더한 뒤 1의 보수(One's Complement)를 취한 것을 값으로 함. 이때 checksum은 길이가 n이므로, 덧셈의 결과값은 $0 \sim 2^n - 1$의 값만을 가질 수 있음. 덧셈 시에 길이가 m을 넘어가면 넘어간 왼쪽 bit열(carry bit)을 잘라 오른쪽 bit열에 다시 더함. 더하는 방식이 조금 독특하기는 한데, 어떤 방식으로 더하든 checksum 값은 그것의 1의 보수이므로, checksum을 다시 구할 때 error가 발생하지 않았다면 0이 도출됨.

이때 계산을 편리하게 하려면, bit열을 십진수로 바꾼 뒤 다 더하고, 다시 이진수로 바꿔서 넘어가는 값을 잘라서 더하면 됨. 이후 1의 보수를 취함.

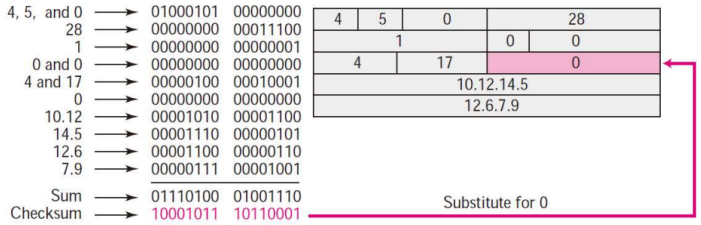
2. 동작 과정

송신자는 packet을 section으로 쪼개 checksum을 구하고, 이를 packet에 포함시켜 전송함. 구체적으로는, checksum을 0으로 하고, checksum을 포함하는 전체 section에 대해 checksum을 계산하여 그것을 checksum으로 하는 것임.

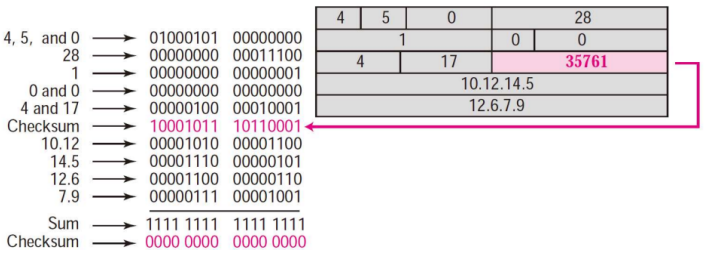
수신자는 수신받은 message(checksum 포함)를 unit 단위로 쪼개 checksum을 다시 계산함. 여기에서도 section과 checksum 모두에 대해 계산함. 해당 연산의 결과가 0이면 error가 발생하지 않은 것으로, 0이 아니면 error가 발생한 것으로 판단함.



- Example of checksum calculation at the sender



- Example of checksum calculation at the receiver



2.4.7. Security

IP에는 아래와 같은 보안 상의 문제점들이 존재함.

오늘날에는 IPSec(IP Security)라는 protocol을 사용하여 아래의 문제점들을 해결함. 아래의 정리한 각 문제점에 대한 방지 기법들을 지원함.

1. Packet Sniffing
Packet Sniffing은 침입자가 IP packet을 중간에 가로채 복사본을 만드는 것임. 이는 packet을 수정하지 않으므로 송신자와 수신자가 알아채기 어려움.

원천 봉쇄하지는 못하지만, packet에 대한 encryption(암호화)을 통해 무력화할 수 있음.

2. Packet Modification
Packet Modification은 침입자가 packet 내용을 수정하는 것임.

data integrity(무결성) mechanism을 활용해 detect할 수 있음.

3. IP Spoofing
IP Spoofing은 공격자가 자신의 것이 아닌 IP address를 source로 하는 IP packet을 생성해 전송하는 것임.

origin authentication(발신지 인증)을 적용해 방지할 수 있음.

2.5. ARP

2.5.1. Address Mapping

Address Mapping(주소 변환)은 physical address와 logical address 사이의 mapping(변환, 쌍)임.

host/router는 network level에서 logical address로 식별되지만, physical level에서는 physical address로 식별되므로 통신을 위해서는 이 둘 간의 mapping이 필요함.

mapping에는 static mapping과 dynamic mapping이 있음.

1) Static Mapping
Static Mapping은 physical address와 logical address에 대한 고정된 table을 만들어 활용하는 방식임. 이는 physical address의 수정 등이 발생할 수 있으므로 한계가 존재함.

2) Dynamic Mapping
Dynamic Mapping은 protocol을 사용하여 logical address와 physical address 사이의 mapping을 찾는 방

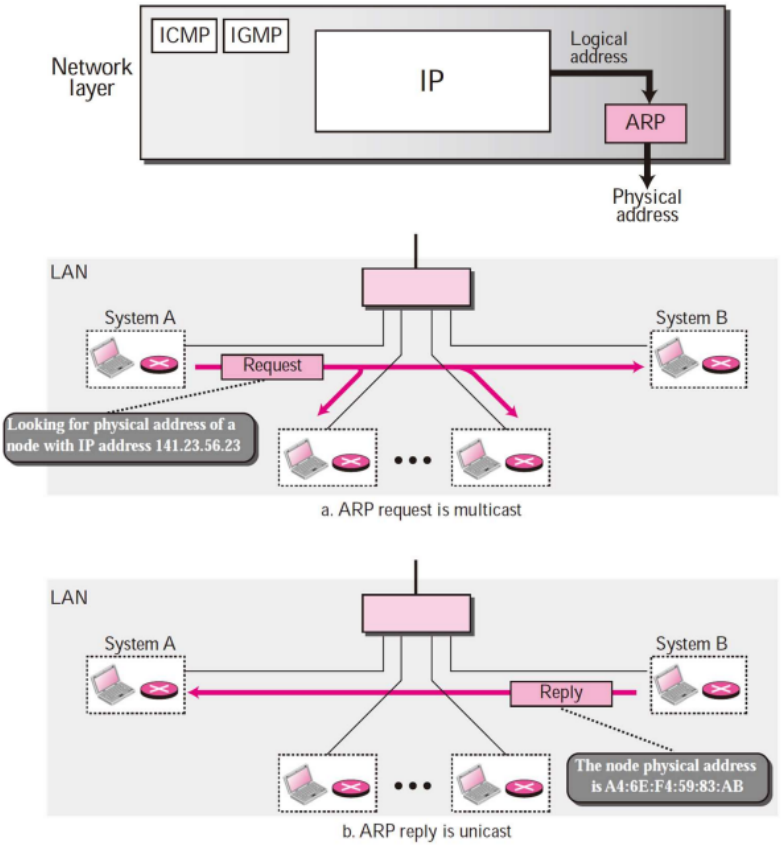
식임. dynamic mapping을 위한 protocol로는 ARP와 RARP(Reverse ARP, physical address로 logical address를 찾는 protocol.)가 있는데, RARP는 다른 protocol에 대체되어 사라졌으므로 아래에서는 ARP에 대해 정리함.

2.5.2. ARP

1. ARP

ARP(Address Resolution Protocol)는 수신자의 IP address로, 대응되는 경로의 physical address를 찾아 data-link로 전달하는 network layer와 data-link layer의 중간 protocol임. 수신자의 IP address를 알아도, 해당 packet을 encapulation해서 전송하려면 결국 대응되는 physical address를 알아야 함.

대략적인 동작은 아래와 같음. 송신자는 수신자의 IP address로 physical address를 묻는 ARP request packet을 broadcast함. 해당 ARP packet에 대한 응답자는 자신의 physical address를 포함하는 ARP reply packet을 송신자에게 unicast함.



ARP에 의해 생성된 mapping도 특정 시간마다 ARP가 자동으로 생성하는 dynamic mapping과, os나 사용자가 직접 생성하는 static mapping으로 구분할 수 있음. 즉, ARP는 기본적으로 dynamic mapping에 대한 것이지만, static mappping도 지정할 수 있음.

2. ARP Packet

ARP에서 전송하는 ARP Packet은 아래와 같은 format을 가짐. 여기에서 hardware는 data-link layer를, protocol은 network layer를 의미함. 이런 ARP packet은 그대로 frame으로 encapsulation됨.

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)		

1) Hardware/Protocol Type : 각각 data-link layer(ex. ethernet은 1), network layer에 사용된 protocol(ex. IPv4는 0x0800)을 지정함.

2) Hardware/Protocol Length : 각 layer에서의 address 길이를 바이트 단위로 지정함(ex. ethernet은 6, IPv4는 4).

3) Operation : 해당 packet이 ARP request(요청)으로 동작하는지, ARP reply(응답)로 동작하는지를 지정함.

4) Target Hardware Address : physical address를 지정함. request의 경우 address를 모르므로 0으로 지정됨.

3. ARP 활용 cases

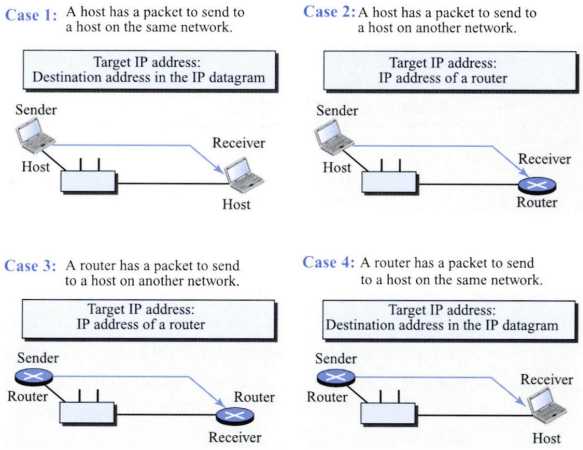
ARP가 활용되는 상황은 아래와 같이 4가지 case로 나눌 수 있음.

1) Case 1 : 송신자가 host이고 같은 network의 다른 host가 수신자인 경우(host에서 host로.). datagram header의 destination IP address가 physical address로 mapping됨.

2) Case 2 : 송신자가 host이고 다른 network의 host가 수신자인 경우(host에서 router로.). routing table에서 찾은(못 찾았으면 default.) router의 IP address가 physcial address로 mapping됨.

3) Case 3 : 송신자가 router이고 다른 network의 host가 수신자인 경우(router에서 router로.). 다음 router의 IP address가 physical address로 mapping됨.

4) Case 4 : 송신자가 router이고 같은 network의 host가 수신자인 경우(router에서 host로.). datagram header의 destination IP address가 physical address로 mapping됨.



4. ARP Cache

ARP Cache는 host/router가 ARP에 의해 mapping(logical/physical address 쌍)을 발견했을 때 반복적으로 ARP 요청을 보내지 않도록 그 정보를 저장해두는 cache임.

어떤 host A에 대해 logical address와 phyisical address 쌍을 (A, M_A)라고 하자. 동일한 network에 속하는 host A, B, C가 존재할 때 A와 B에 대한 ARP cache의 update는 아래와 같이 수행될 수 있음.

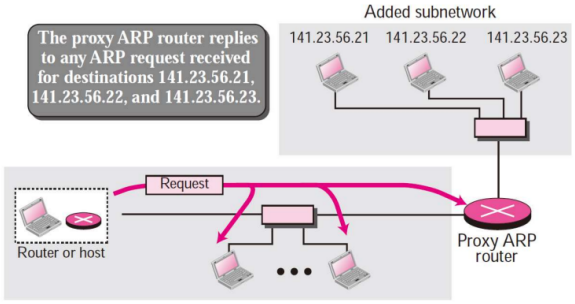
1) B가 A를 찾기 위해 broadcast한 ARP request가 A에게 전달된 경우.
A는 B에 대한 정보 (B, M_B)를 자신의 ARP cache에 저장하고, 자신의 정보 (A, M_A)로 ARP reply를 B에게 unicast함. B는 A에 대한 정보를 자신의 ARP cache에 저장함.

2) B가 C를 찾기 위해 broadcast한 ARP request가 A에게 전달된 경우.
A가 B에 대한 정보를 ARP cache에 포함하고 있었다면, 해당 ARP request에서 추출한 새로운 B에 대한 정보로 ARP cache의 내용을 update함. 즉, physical data에 변동이 있었을 수 있으므로 해당 정보를 활용해 최신화하는 것. A가 B에 대한 정보를 ARP cache에 포함하고 있지 않았다면, 해당 ARP request를 discard함.

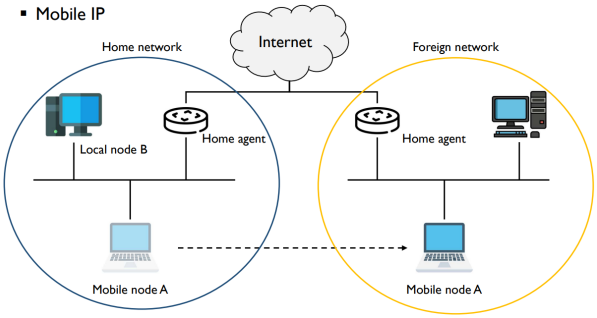
3) ARP cache의 각 entry는 lifetime이 1200s 정도인데, 해당 시간이 지나면 그 entry에 대한 interface에 ARP request를 unicast함. 만약 ARP reply를 수신받으면 lifetime을 reset함.

5. Proxy ARP
Proxy ARP는 host 집합을 대표해 수행되는 ARP임. ARP에 대한 subnetting 효과를 냄.

proxy ARP를 수행하는 router가 자신이 대표하는 host 집합의 한 host를 타겟으로 하는 ARP request를 받으면, 자신의 physical address로 구성한 ARP reply를 전송함. 이에 따라 packet을 받으면 이를 적절한 host/router로 전송함.



예를 들어, Mobile IP에서는 mobile 장치가 다른 네트워크로 이동해도 home agent가 proxy ARP를 수행하여 해당 장치로의 packet을 실제로 장치가 위치한 network로 전달함. Mobile IP는 mobile 장치가 네트워크를 이동하면서도 동일한 IP 주소를 유지하며 통신할 수 있게 해주는 protocol임.

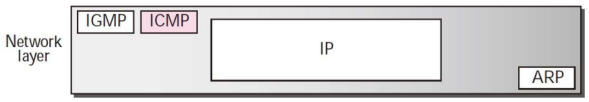


2.6. ICMPv4

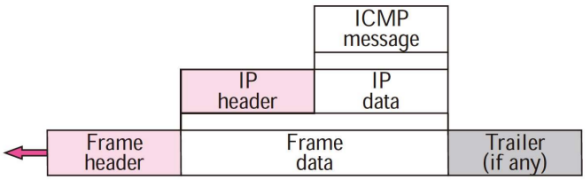
2.6.1. ICMPv4

1. ICMPv4
ICMPv4(Internet Contorol Message Protocol Version 4)는 error-reporting과 query message를 지원하는 IP에 대한 동반 protocol임.

IP는 packet이 손실 또는 discard 된 경우에 대한 error-correcting과, host 등에 대한 management를 수행하는 query 기능이 포함되어 있지 않음. ICMPv4는 이를 보완함.

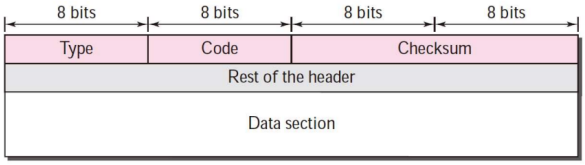


ICMP message는 직접 data-link layer로 전달되는 대신, IP의 datagram 내에 encapsulation됨. 이 경우 datagram header의 protocol field의 값이 1로 지정됨.



2. ICMP Message

ICMP Message는 ICMP에 의해 전송하는 message임. 그 format은 아래와 같음.



1) Type : ICMP message의 type 지정.
2) Code : ICMP message의 원인 지정.
3) Checksum : checksum 값. ICMP의 header와 data 모두에 대해 계산함.
4) Rest of the header : 각 message type에 대한 구체적인 정보를 지정.

ICMP message는 error-reporting message와 query message로 나뉨.

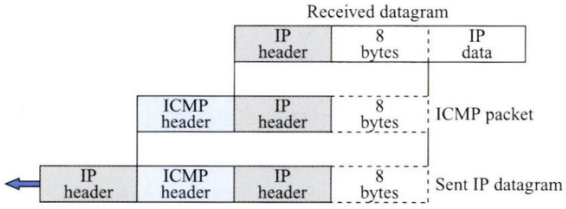
Category	*Type*	*Message*
Error-reporting messages	3	Destination unreachable
	4	Source quench
	11	Time exceeded
	12	Parameter problem
	5	Redirection
Query messages	8 or 0	Echo request or reply
	13 or 14	Timestamp request or reply

2.6.2. ICMP Message Type : Error-reporting Message

Error-reporting Message는 router나 destination host가 datagram 처리 중에 발생하는 에러를 보고하는 message임.

ICMP는 error correction은 수행하지 않고, 단순 report만을 수행함. error correction은 상위 layer의 책임임. datagram에는 source host의 IP address가 포함되어 있으므로, 이 message는 항상 source host에게 전송됨.

모든 error-reporting message는 data section에 error가 발생한 datagram의 IP header와, data 부분의 첫 8바이트(TCP/UDP 등 transport layer의 정보)를 포함함.



또한 error-reporting message의 동작과 관련해서 아래와 같은 주요 사항들이 있음.
1) error-reporting message를 전달하는 datagram에 대해서는 error-reporting message가 생성되지 않

음.
2) error-reporting message는 첫 번째 fragment에 대해서만 생성됨.
3) multicast address를 사용하는 datagram에 대해서는 error-reporting message가 생성되지 않음.
4) special address(127.0.0.0 또는 0.0.0.0 등)를 사용하는 datagram에 대해서는 error-reporting message가 생성되지 않음.

error-reporting message에는 아래와 같은 5가지 type이 존재함.

1. Destination Unreachable
Destination Unreachable은 router가 datagram을 전달할 수 없거나, host가 datagram을 받지 못해(datagram이 reach되지 못함.) 전송되는 message임. 이 경우 전송되던 datagram은 discard되고, source host에게 destination unreachable message가 전송됨.

datagram이 discard된 이유는 아래와 같이 code에 지정됨(실제로는 더 많은 code들이 존재함.). 하드웨어 고장 또는 잘못된 address에 의해 이런 상황이 발생할 수 있음.

1) Code 0 : network가 unreacheable함. 즉, network를 찾지 못함.
2) Code 1 : host가 unreacheable함. 즉, network는 찾았는데 host를 찾지 못함.
3) Code 2 : protocol이 unreachable함. 즉, destinnation과 protocol이 일치하지 않음.
4) Code 3 : port가 unreachable함. 즉, destination에 해당 port가 활성화되어 있지 않음.
5) Code 4 : routing에서 fragmentation이 수행되어야 하는데, flag에 의해 불가능함.
6) Code 5 : option에 source-routing이 지정되었는데, 이를 따르지 못함.

2. Source Quench
Source Quench는 flow control 또는 congestion(혼잡도) control을 위한 message임. router/host가 congestion에 의해 datagram을 discard하면 송신자에게 source quench message를 전송해 전송 속도를 늦추도록 함. quench는 불을 끈다는 의미임.

기본적으로 IP에서는 flow control, congestion control을 제공하지 않으므로 이를 활용할 수 있지만, 현재는 이 책임을 TCP가 수행하므로 사용되지 않는 type임.

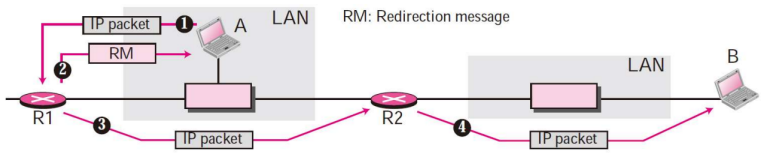
3. Time Exceeded
Time Exceeded는 아래와 같은 두 가지 시간 초과 상황에 송신자에게 전송되는 message임.

1) datagram header의 time to live 값이 0이 되어 discard된 경우.
2) destination에 어떤 message에 대한 fragment들이 time limit 안에 전부 도착하지 못한 경우.

4. Parameter Problem
Parameter Problem은 datagram header에 불명확하거나 빠진 부분이 존재하는 경우 전송되는 message임.

5. Redirection
Redirection은 host의 routing table을 redirect(update)하도록 전송되는 message임.

internet에는 router보다 더 많은 수의 host가 존재함. 이에 따라 router는 동적으로 routing table을 관리하고, host는 traffic 문제 때문에 정적으로 routing table을 관리함. 그래서 많은 경우 host는 잘못된 router(주로 default)로 packet을 전송하게 되고, 해당 router는 이를 적절한 위치로 forwarding하게 됨. 이에 packet을 받은 router는 host의 routing table을 update하기 위해 redirection message를 전송함.



2.6.3. ICMP Message Type : Query Message

Query Message는 host 나 network 관리자 등이 다른 host나 router로부터 특정 정보를 얻기 위해 사용하는 message임.

여러 query message가 존재하지만, 현재는 두 가지만 활용됨. 여기에서는 아래와 같이 ehco request and

reply만을 살펴봄.

1. Echo Request And Reply
Echo Request And Reply는 진단(diagnose)에 활용되는 message로, 두 host/router가 서로 통신할 수 있는지를 확인함.

한 host/router가 echo request(type 8)를 보내고, 대응되는 host/router는 echo reply(type 0)를 보내 reachable하고 통신이 가능함을 나타냄.

Type 8: Echo request
Type 0: Echo reply

Type: 8 or 0	Code: 0	Checksum
Identifier		Sequence number
Optional data Sent by the request message; repeated by the reply message		

2.6.4. Debugging Tool

network layer에 대한 debugging tool을 알아보자.

1. Ping
Ping은 어떤 host가 alive하고 응답이 가능한지 점검하는 debugging tool임. source host는 echo request message를 destination host에 전송하고, destination host는 echo reply message를 source host에 전송함.

ping은 message의 sending time을 message의 data section에 넣어서 전송함. 이를 활용해 RTT(Round-trip time. 왕복 시간)을 계산할 수 있음.

아래와 같이 ping 명령어로 특정 서버와의 통신 상태를 점검할 수 있음.

```
$ 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=1.91 ms
64 bytes from tiptoe.fhda.edu (153.18.8.1): icmp_seq=1 ttl=62 time=2.04 ms
64 bytes from tiptoe.fhda.edu (153.18.8.1): icmp_seq=2 ttl=62 time=1.90 ms
64 bytes from tiptoe.fhda.edu (153.18.8.1): icmp_seq=3 ttl=62 time=1.97 ms
```

2. Traceroute
Traceroute(UNIX) 또는 tracet(Windows)는 soure에서 destination으로의 경로(route)를 추적(trace)하는 debugging tool임.

traceroute는 ICMP message와 TTL(time to alive) field를 활용해 경로를 찾음. 즉, TTL이 rotuer를 거치며 1씩 줄어들다가 0이 됐을 때 전송하는 time exceeded message를 활용함.

구체적으로는, TTL을 1로 지정한 packet을 전송한 뒤 discard됐을 때의 전달받는 ICMP message로부터 router IP address와 RTT를 얻는 것에서 시작하여, destination에 도달할 때까지 TTL을 1씩 증가시켜 보내는 작업을 반복함. 특히 마지막 destination에 도달했을 때에는 UDP가 지원하지 않는 port 번호를 지정해 ICMP message를 전송받도록 함.

```
$ traceroute(tracert) xerox.com
traceroute to xerox.com (13.1.64.93), 30 hops max, 38 byte packets
1 Dcore.fhda.edu (153.18.31.254) 0.622 ms 0.891 ms 0.875 ms
2 Ddmz.fhda.edu (153.18.251.40) 2.132 ms 2.266 ms 2.094 ms
3 Cinic.fhda.edu (153.18.253.126) 2.110 ms 2.145 ms 1.763 ms
4 cenic.net (137.164.32.140) 3.069 ms 2.875 ms 2.930 ms
5 cenic.net (137.164.22.31) 4.205 ms 4.870 ms 4.197 ms
6 cenic.net (137.164.22.167) 4.250 ms 4.159 ms 4.078 ms
```

2.7. Unicast Routing Protocol

2.7.1. Unicast Routing Protocol

1. Routing Protocol
Routing Protocol은 각 경로 별 cost(비용)을 할당하여 dynamic routing table을 위해 최적의 경로를 찾고, table을 생성 및 갱신하는 역할의 protocol임. 여기에서 설명할 routing protocol은 unicast(one-to-one) 통신에서의 protocol임.

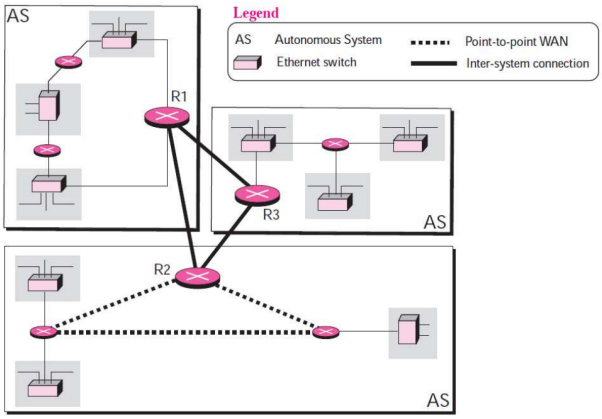
Static Routing Table은 entry가 수동으로 결정되는 table이고, Dynamic Routing Table은 변화가 발생했을 때 entry가 자동으로 update되는 table임.

이때 cost는 Metric이라고도 부름.

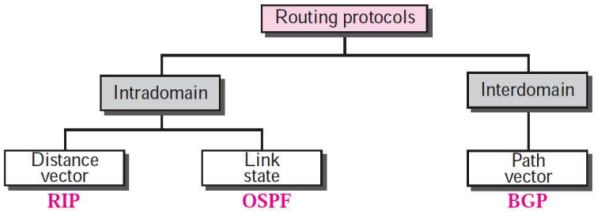
2. Intradomain vs. Interdomain

AS(Autonomous System)는 어떤 단일 기관(administration)의 권한(authority) 아래에 있는 network와 router의 집합임. 현재의 internet은 한 가지 routing protocol로 모든 부분에 대한 routing table을 관리하기에는 그 크기가 너무 크므로, internet을 여러 개의 AS로 나눠 관리함.

AS 내부에서의 routing을 Intradomain Routing, 서로 다른 AS끼리의 routing을 Interdomain Routing이라고 함.



아래에서 설명할 routing protocol은 intradomain에 대한 것 2개와, interdomain에 대한 것 1개임.



2.7.2. Distance Vector Routing

1. Distance Vector Routing

Distance Vector Routing은 AS를 노드(router)와 링크(network)로 구성된 그래프로 생각하고, bellen-ford 알고리즘으로 각 router에서 특정 network까지의 최단거리(cost)를 계산하여 table에 반영하는 방식임.

Bellman-Ford 알고리즘은 두 노드 사이의 최단거리(최소 비용)을 찾는 알고리즘임. 간단히 설명하자면, 노드 i와 j 사이의 최단거리는, 각 이웃에 대해 i의 이웃에서 j까지의 최단거리와 i에서 이웃까지의 거리를 더한 값들 중 최솟값(이웃 별로 값을 계산함.)임. 이때 링크가 존재하지 않는 두 노드 사이에는 무한대(infinity)의 거리를 가지는 링크로 처리됨. 이 경우 임의의 두 노드 사이(링크)의 거리를 안다면, 전체 그래프에 대해 최단거리를 계산해 놓을 수 있음.

bellman-ford 알고리즘을 router와 network로 구성된 그래프에 적용하는 경우, 아래와 같은 설정들의 필요함.

1) 두 대상 사이의 거리(cost)는 경로의 존재하는 network의 개수임. 이때 도착 network도 포함하여 셈. 즉, 거치게 되는 링크의 개수로 이해할 수 있음.

2) 각 router는 경로에 대해 3가지 정보 dest(destination network), cost(dest까지의 cost), next(경로에 있는 첫 번째 router)를 table에 저장해야 함.

3) 각 router는 이웃 router로부터 정보를 수신하면 자신의 table을 update하고, 또한 자신의 정보가

수정되면 이를 이웃 router에게 전송해 update하도록 함. (최단거리를 공유해야 알고리즘이 동작함.)

4) router 간에 전송하는 정보는 R이라고 하며, R은 2가지 정보 dest(해당 router로부터의 dest), cost(해당 router로부터 dest까지의 cost)를 포함함. next는 어차피 해당 router이므로 필요 없음.

2. 동작 과정

동작 과정은 초기화 단계와 update 단계로 나눌 수 있음. 앞에 정리한 것처럼, table에 변화가 발생하면 그 정보를 이웃 router에게 전송함. 그 스도코드는 아래와 같음.

```
1  Distance_Vector_Algorithm ( )
2  {
3      // At startup
4     for (i = 1 to N)            // N is number of ports
5     {
6         Table_i.dest = address of the attached network
7         Table_i.cost = 1
8         Table_i.next = —        // Means at home
9         Send a record R about each row to each neighbor
10    } // end for loop
11
12     // Updating
13    repeat (forever)
14    {
15        Wait for arrival of a record R from a neighbor
16        Update (R, T)           // Call update module
17        for (i = 1 to N)        // N is the current table size
18        {
19              Send a record R about each row to each neighbor
20        }
21     }  // end repeat
22
23 }  // end Distance_Vector
24 Update (R, T)                  // Update module
25 {
26     Search T for a destination matching the one in R
27     if (destination is found in row i)
28     {
29         if (R.cost + 1 < T_i.cost    or    R.next == T_i.next)
30         {
31               T_i.cost = R.cost + 1
32               T_i.next = Address of sending router
33         }
34      else  discard  the  record      // No change is needed
35      }
36      else
37       // Insert the new router
38      {
39          T_N+1.dest = R.dest
40          T_N+1.cost = R.cost + 1
41          T_N+1.next = Address of sending router
42          Sort the table according to destination address
43      }
44 }  // end of Update module
```

초기화 단계에는 각 router가 이웃 network와 그 최단경로(1)에 대해 정보를 저장함. 이는 router를

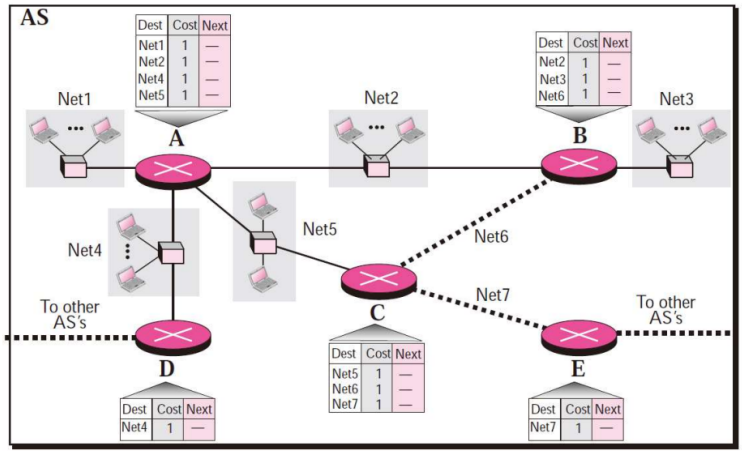
거치지 않고 연결되므로 next값은 없음.

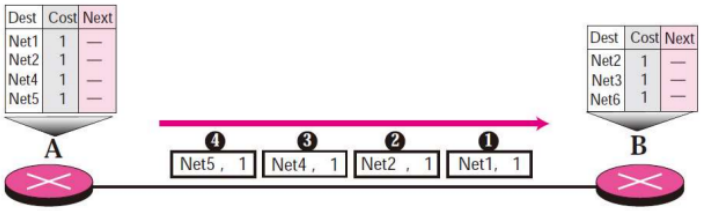
update 단계에는 R.dest를 table(T_i. table T의 특정 entry i.)의 dest와 비교하여 R이 table에 존재하는 dest에 대한 정보인지를 확인함. 이후 존재 여부에 따라 아래의 두 가지 중 하나가 수행됨.

1) dest가 존재하는 경우.
아래의 수식이 참이면 R의 정보로 table을 update함. 즉, 해당 정보를 사용하면 더 적은 비용의 경로를 얻게 되거나, 이미 사용하던 경로에 변화가 발생한 경우에 update하는 것임. 참이 아니면 해당 정보를 discard함.

$$R.cost + 1 < T_i.cost \quad or \quad R.next == T_i.next$$

2) dest가 존재하지 않는 경우.
R의 정보를 table에 추가함.





Routing Table B

Dest	Cost	Next
Net1	2	A
Net2	1	—
Net3	1	—
Net6	1	—

After receiving record 1

Dest	Cost	Next
Net1	2	A
Net2	1	—
Net3	1	—
Net6	1	—

After receiving record 2

Dest	Cost	Next
Net1	2	A
Net2	1	—
Net3	1	—
Net4	2	A
Net6	1	—

After receiving record 3

Dest	Cost	Next
Net1	2	A
Net2	1	—
Net3	1	—
Net4	2	A
Net5	2	A
Net6	1	—

After receiving record 4

A

Dest	Cost	Next
Net1	1	—
Net2	1	—
Net3	2	B
Net4	1	—
Net5	1	—
Net6	2	C
Net7	2	C

B

Dest	Cost	Next
Net1	2	A
Net2	1	—
Net3	1	—
Net4	2	A
Net5	2	A
Net6	1	—
Net7	2	C

C

Dest	Cost	Next
Net1	2	A
Net2	2	A
Net3	2	B
Net4	2	A
Net5	1	—
Net6	1	—
Net7	1	—

D

Dest	Cost	Next
Net1	2	A
Net2	2	A
Net3	3	A
Net4	1	—
Net5	1	A
Net6	3	A
Net7	3	A

E

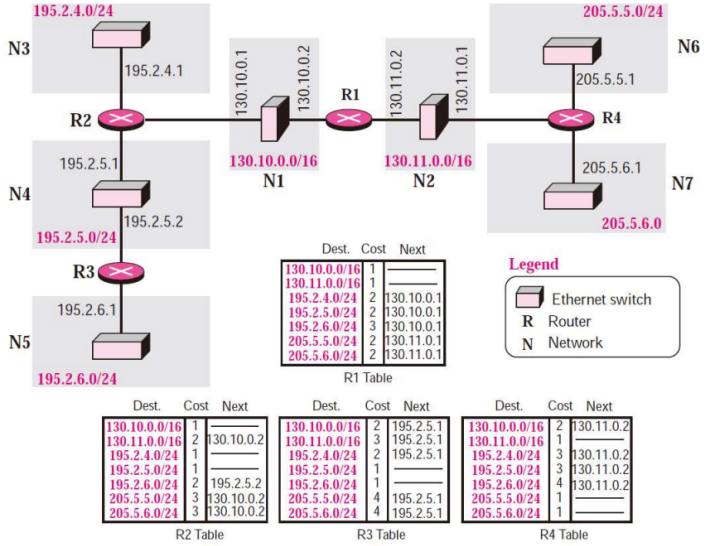
Dest	Cost	Next
Net1	3	C
Net2	3	C
Net3	3	C
Net4	3	C
Net5	2	C
Net6	2	C
Net7	1	—

3. RIP

RIP(Routing Information Protocol)는 distance vector routing을 기반으로 구현된 intradomain routing protocol임.

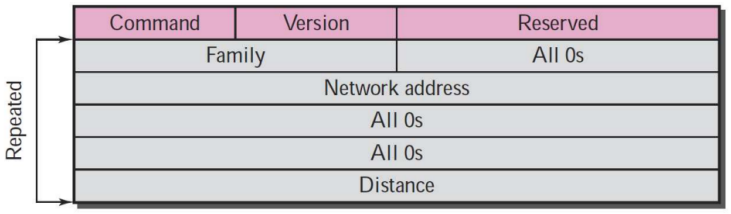
distance vector routing과 동일하게, routing table은 dest(network address), cost(network/hop의 개수), next(경로의 바로 다음 router의 IP address)로 구성됨.

이때 무한대 값은 16으로 처리됨. 즉, 각 경로가 가지는 최대 cost(metric) 값은 15임.



4. RIP Message

RIP message는 아래와 같은 field들로 구성됨. R이 포함하는 정보를 생각하면 이런 구성은 당연함.



1) Command : message의 종류(request는 1, response는 2)를 나타내는 8비트 field.
2) Version : RIP version을 나타내는 8비트 field.
3) Family : protocol 계열(TCP/IP는 2)을 나타내는 16비트 field.
4) Network Address : destination network의 address를 나타내는 field. 원래는 14바이트로 설계되었지만, IP address는 4바이트이므로 나머지는 0으로 채워져 있음.
5) Distance : 해당 message를 전송하는 router로부터 destination network까지의 cost(distance)를 나타내는 8바이트 field.

command field를 보면 알 수 있듯이 RIP message에는 request와 response가 있음.

1) Request message : 새롭게 등장했거나, 어떤 entry가 expire된 경우 정보 요청을 위해 전송하는 message임. 아래와 같이 특정 entry들을 지정해 정보를 요청할 수도 있고, 모든 entry에 대해 정보를 요청할 수도 있음.

2) Response message : response message는 정보를 제공하는 응답 message임. request에 의해 특정 entry 또는 전체 entry 정보를 담아서 전송하는 것은 Solicited Response Message라고 하고, 특정 시간

간격(timer)이나 변경이 발생한 경우에 전송하는 것은 Unsolicited Response Message(Update Message)라고 함. solicit는 요청하는 것을 의미함.

5. RIP Timer
RIP에서는 아래와 같이 3가지 종류의 Timer를 사용함.

1) Periodic Timer : unsolicited response message를 전송하는 주기에 대한 timer(ex. 30s).

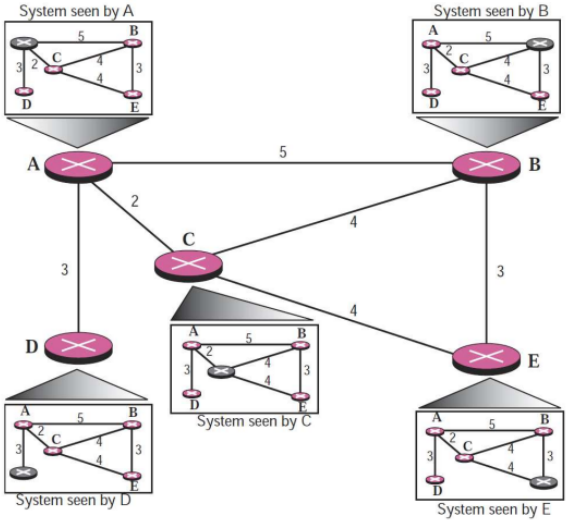
2) Expiration Timer : table에서 path에 대한 유효성을 나타내는 timer(ex. 180s). response message를 받으면 timer가 초기화되고, 해당 timer가 expire되면 해당 path는 무효화됨.

3) Garbage Collection Timer : table에서 무효화된 path에 대한 유예 기간을 나타내는 timer(ex. 180s). 무효화된 path는 바로 삭제되는 것이 아니라, 이 timer가 끝나면 그 때 삭제됨. 이 시간 동안 이웃이 해당 path가 무효화된 것을 알아차릴 수 있음.

2.7.3. Link State Routing

1. Link State Routing
Link State Routing은 AS를 노드(router)와 cost를 가지는 링크로 구성된 topology(그래프)로 생각하고, 각 router에서 topology에 대해 dijkstra 알고리즘을 사용해 최단경로를 계산하는 방식임.

link state routing에서는 우선 각 router들이 LSP(Link State Packet)을 주고받으며 topology를 파악함. 그 결과로 각 router는 동일한 전체 topology를 가지게 됨. 이때 LSP가 network 내의 모든 다른 router에게 효율적이고 신뢰할 수 있는 방식으로 전파하는 과정을 Flooding이라고 함. 이후 각 노드에서의 topology를 다르게 활용하므로 table은 서로 다르게 구성됨. 이후 dijkstra 알고리즘을 적용함.

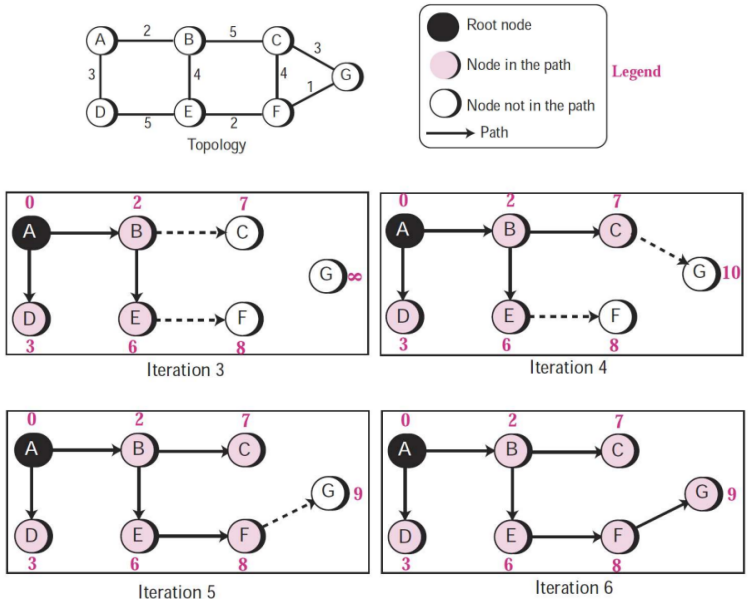


Dijkstra 알고리즘은 아래와 같이 initialization과 iteration을 거쳐 수행되며, 그 결과로 자신을 루트로 하는 shortest path tree가 구성됨. Shortest Path Tree는 루트와 임의의 한 노드 사이의 거리가 모두 최소인 tree임.

1) Initialization
현재 노드(router)를 루트로 해서 path(tree)에 추가함. 이때 path와 연결된 노드는 그 링크의 cost를 활용해 루트와의 거리가 계산되고, 연결되지 않은 노드는 루트와의 거리가 무한대로 지정됨.

2) Iteration
모든 노드가 path에 추가될 때까지 아래의 두 과정을 반복함.
-> path에 포함되지 않은 노드 중 루트로부터와의 거리가 최소인 것을 path에 포함시킴.
-> 추가된 노드를 활용하여 추가되지 않은 노드들에 대해 거리를 재계산함. 노드 i가 새로 추가되었다면, 노드 j에 대한 거리는 아래의 수식으로 계산됨. D_j는 루트에서 j까지의 거리이고, c_{ij}는 i와 j 사이의 cost임. 즉, 추가된 노드를 거치는 경우의 거리와 기존 거리 중 작은 것을 값으로 함.

$$D_j = \min(D_j, D_i + c_{ij})$$

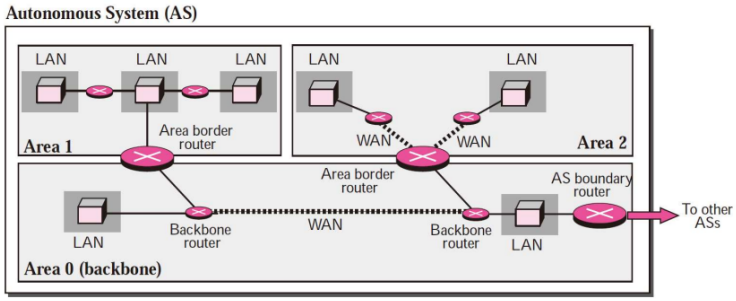


2. OSPF

OSPF(Open Shortest Path First)는 link state routing을 기반으로 구현된 intradomain routing protocol임.

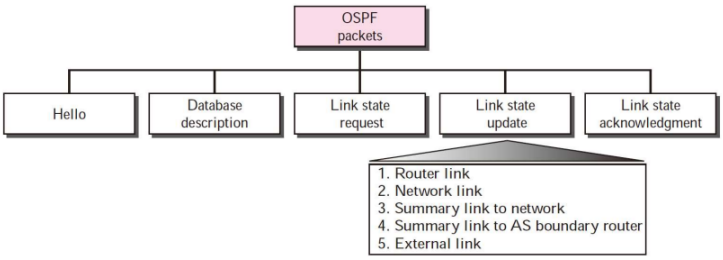
OSPF는 routing을 효율적으로 수행하기 위해 area라는 개념을 사용함. Area는 host/router/network의 집합으로, AS를 더 작게 나눈 것임. 이때 area 내의 모든 network는 연결되어 있음. 또한 Backbone은 area의 일종으로, AS 내의 다른 모든 area는 backbone에 연결되어 있어야 함. 추가로, Area Border Router는 area의 경계에 위치하며 area에 대한 정보를 다른 area로 전달하는 등의 통신을 수행함.

OSPF에서는 administrator가 각 경로에 대해 metric(cost)를 할당함. 이때 metric은 제공 서비스에 따라 delay, throughput 등일 수 있음.

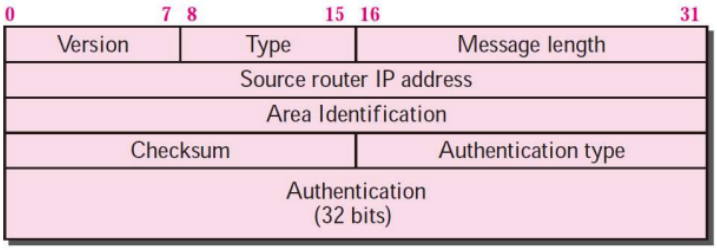


3. OSPF Packet

OSPF는 아래와 같이 hello, database description, link state request, link state update, link state acknowledgment로 총 5가지 종류의 packet을 정의함.



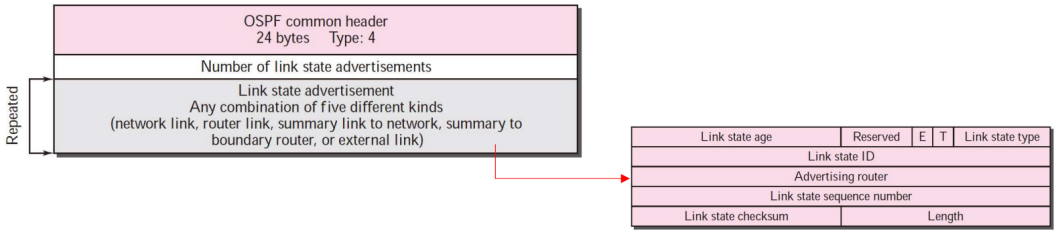
각 packet은 공통적으로 아래의 header를 가짐. type은 packet의 종류를, area identification은 routing 이 수행되는 area를 지정함.



4. OSPF Packet : Link State Update Packet

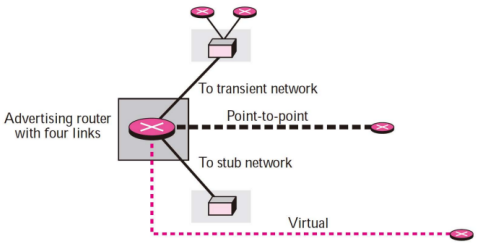
OSPF packet 중 Link State Update Packet은 router가 링크의 상태를 advertise(다른 router에게 알림.) 할 때 사용됨. 이 작업을 LSA(Link State Advertisement)라고 하는데, link state update packet은 여러 개의 LSA를 포함할 수 있음. 즉, link state update packet은 앞에서 설명한 LSP에 대응되고, 각 router 에서는 이 packet의 정보를 활용해 topology를 구성함.

이 packet은 OSPF header에서 type이 4로 지정되고, header 이후에는 아래와같이 field가 구성됨.

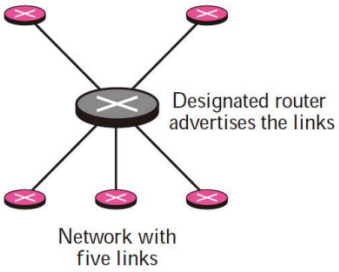


Link state age는 packet이 생성된 이후로부터 경과된 시간을 초 단위로 나타낸 것이고, Link State Type 은 아래와 같은 LSA 종류를 지정함. 이때 각 router는 경로에 대한 정보를 알고 있어야 최적의 경로로 packet을 전송할 수 있으므로 이런 종류의 정보 교환이 존재하는 것임.

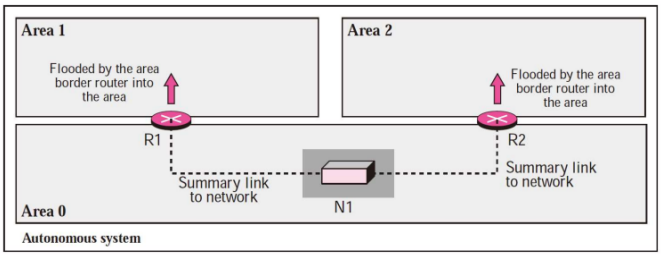
1) Router Link LSA : 실제 router가 가진 모든 링크와, 해당 링크에 연결된 router의 정보를 포함함.



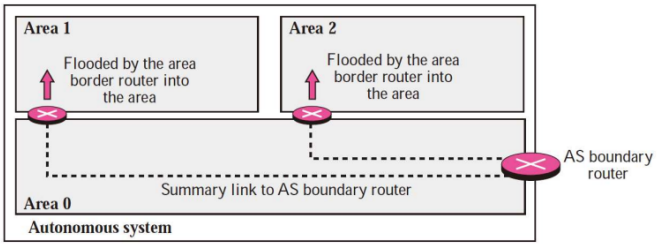
2) Network Link LSA : network가 가진 링크(router)의 정보를 포함함.



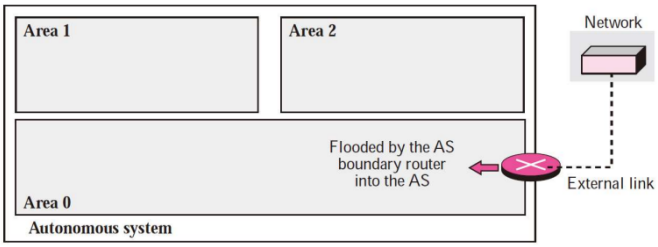
3) Summery Link to Network LSA : area 바깥의 정보를 포함하며, area broader router에 의해 제공됨.



4) Summery Link to AS Boundary Router LSA : AS boundery router까지의 경로에 대한 정보를 포함하며, area broader router에 의해 제공됨.



5) External Link LSA : AS 외부의 정보를 포함하며, AS boundery router에 의해 제공됨.



2.7.4. Path Vector Routing

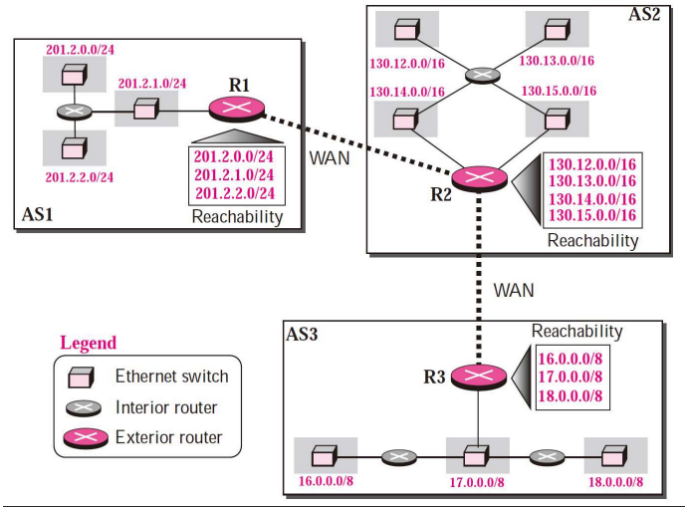
1. Path Vector Routing

Path Vector Routing은 각 AS에 대한 정보를 수집 및 저장하는 path router를 사용하고, 그 정보를 다른 AS의 path router끼리 교환하도록 하여 interdomain에 대한 routing을 수행하는 빙식임.

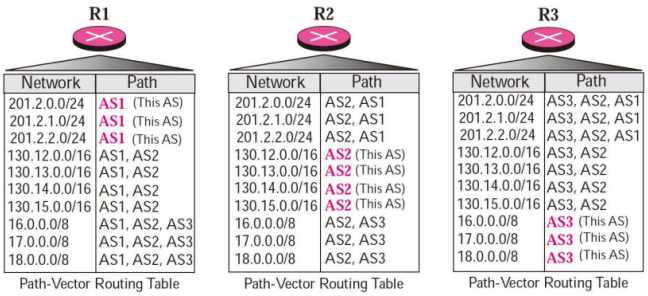
앞에서 다룬 distance vector routing과 link state routing은 intradomain에 대해서는 잘 동작하지만, interdomain에 대해서는 불안정하거나 연산량이 너무 많아 적용하기 어려움. interdomain에 대해서는 path vector routing을 사용함.

path vector routing에서 어떤 AS는 자신이 관리하는 network에 대해 Reachability(도달 가능성)가 있다고 함. interdomain의 AS는 reachability 정보를 수집해서 가지고 있어야 하고, 이를 다른 AS와 공유하여 각 AS가 통신 가능한 network 목록을 구성하도록 함.

이를 수행하는 router는 Path Router(exterior router, speaker node)라고 함. 즉, path router는 특정 AS를 대표하는 router로, 자신의 AS에 대한 reachability 정보를 포함하는 path vector routing table을 생성하고, 다른 path router와 그 정보를 공유함. path vector routing table에는 특정 network와, 그 network에 reachability를 가지는 path(AS)가 저장됨.



아래는 각 path router가 reachability 정보를 공유한 결과임. 이 정보는 aggregation될 수 있음.



2. BGP

BGP(Border Gateway Protocol)는 path vector routing을 기반으로 구현된 interdomain routing protocol임.

앞에서는 path router에서 network와 path를 저장하다고 설명했지만, 실제로 BGP가 저장하는 값은 path(AS)가 아닌 path에 대한 정보인 Attribute를 저장함. 여러 가지 종류의 attribute들이 있는데, 이는 아래와 같이 두 가지 종류로 나뉨.

1) Well-known Attribute : 모든 BGP router(path router에 대응되는 router.)가 인식할 수 있는 attribute.
ORIGIN : path 정보를 제공하는 발신지.
AS_PATH : destination에 도달하기 위해 거쳐야 하는 AS 목록.
NEXT-HOP : destination에 도달하기 위해 보내져야 할 다음 router.

2) Optional Attribute : 모든 BGP router가 인식해야 하는 것은 아닌 attribute.

3. BGP Message

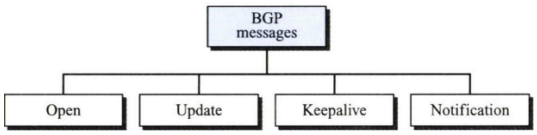
BGP message는 아래와 같이 4가지로 나뉨.

1) Open : BGP router가 이웃한 BGP router와 이웃 관계 설정을 위해 전송하는 message. keepalive messgae로 응답받으면 이웃 관계가 설정됨.

2) Update : BGP router가 destination에 대한 정보를 update하기 위해 전송하는 message. BGP의 핵심 message임. 이때 새로운 destination을 알리는 경우 하나의 update message는 하나의 destination 정보만을 포함할 수 있음.

3) Keepalive : open message를 전달받은 BGP router가 이웃 관계를 수락하거나, 이미 존재하는 이웃 관계가 expire되지 않도록 특정 시간 간격마다 서로에게 전달하는 message.

4) Notification : error가 발생하거나 연결을 끊으려 할 때 전송하는 message.



3. Transport Layer

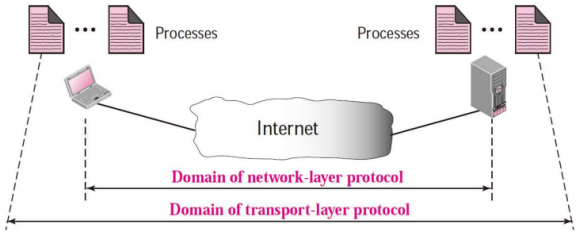
3.1. Transport Layer Service

transport layer에서 제공하는 serivce들을 알아보자. 이는 기본적으로 network layer의 service를 활용해 application layer에 지원하는 service임. 이때 당연하게도 protocol에 따라 여기에서의 service들 중 일부만을 제공하기도 함.

3.1.1. Process-to-process Communication

1. Procses-to-process Commmunication

network layer에서는 computer 수준의 통신이 host-to-host communication이 수행된 반면, transport layer에서는 process-to-process communication이 수행됨. 즉, 적절한 process로의 packet 전달을 처리함. 참고로 여기에서의 process는 transport layer의 service를 활용하는 application layer의 running entity(program)임.



이때 각 process는 port number로 구분됨. 즉, transport layer는 특정 port number에 해당하는 process에 packet을 전달함. port number를 나타낼 때는 16비트를 사용하고, 이에 따라 port number는 0 ~ 65535 사이의 값을 가질 수 있음.

또한 transport layer에서는 하나의 computer에 대해 여러 process를 처리해야 하므로 multiplexing(many to one)과 demultiplexing(one to many)이 활용됨.

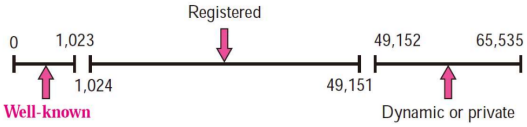
2. ICANN range

ICANN(Internet Corporation for Assigned Names and Numbers)은 DNS, IP address, port number 등을 관리하고 조정하는 비영리 국제 기구임. 특히 ICANN에서는 port number를 아래와 같이 3가지 range로 구분함.

1) well-known : 0 ~ 1023. 특정 process가 사용하도록 ICANN에 의해 미리 정해져 있는 부분. 즉, 사용이 예약된 부분으로, 임의로 수정할 수 없음.

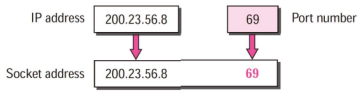
2) registered : 1024 ~ 49151. 특정 기관이 사용하기 위해 ICANN에 등록하여 확보해놓은 부분.

3) dynamic(private) : 나머지. private하게 활용할 수 있는 부분. 즉, 각 os가 임의로 설정해 사용할 수 있음.



3. Socket Address

Soket Programming에 사용하는 Socket Address는 IP address와 port number를 합친 것임.



socket은 application과 transport layer 사이에 위치하며 process-to-process communication을 지원하는 interface임. 리눅스에서 socket은 file처럼 다뤄지고, file에 대한 read write으로 통신을 단순화함. 이에 따라 socket programming에서는 transport layer를 포함하는 아랫쪽 layer를 고려할 필요가 없고, tranport layer부터 data link layer 부분은 os가 처리함.

3.1.2. Flow/Error/Congestion Control

transport layer에서 제공하는 flow/error/congestion control을 알아보자.

1. Flow Control

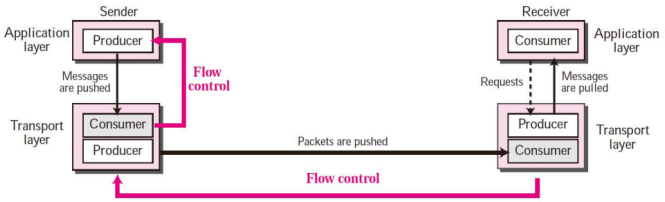
Flow Control은 수신자에서 송신자로부터 데이터를 전달받을 때, 자신이 처리할 수 있는 만큼의 데이터를 전달받도록 전송 속도를 조정하는 것임. 즉, 해당 시스템이 가진 buffer가 overflow되지 않도록 하는 것으로 이해할 수 있음.

여기에서 송신자로부터 수신자로의 데이터 전달 방식은 아래와 같이 두 가지로 나눌 수 있음.

1) Pushing : 수신자로부터의 요청이 없어도, 전송할 데이터가 생기면 즉시 전송하는 방식.
2) Pulling : 수신자로부터의 요청이 들어오는 경우에 전송할 데이터를 전송하는 방식.

transport layer에서의 flow control은 아래의 그림과 같이 두 가지로 나눌 수 있음. 송신자의 application layer와 transport layer, 그리고 송신자의 transport layer와 수신자의 transport layer에서는 push가 수행되므로 flow control이 필요함. 반면 수신자의 application layer와 transport layer에서는 pull이 수행되므로 별도의 flow control이 필요하지 않음.

1) 송신자 측의 transport layer에서 송신자 측의 application layer로의 flow control.
2) 수신자 측의 transport layer에서 송신자 측의 transport layer로의 flow control.

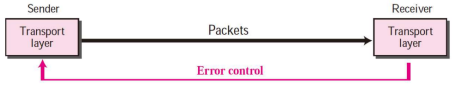


2. Error Control

transport layer에서 수행하는 error control은 아래와 같은 책임을 가짐.

1) 전달된 packet이 discard 또는 corrupt되었는지 확인하고, 제대로 전달되지 않은 packet을 track하고 재전송함.
3) 전달된 packet이 중복된(duplicated) 것인지 검사하고 discard함.
3) packet이 전부 도착할 때까지(timer 사용) out-of-order인 packet들을 buffer에 저장함.

error control에서는 flow control과 달리 application layer와 transport 사이에서의 error는 고려하지 않고, transport layer 사이의 통신만 고려함.



error control에서는 sequence number와 ack 등을 활용함. sequence number는 packet의 순서를 나타내는 수이고, ack(acknowledgment)는 데이터가 전달되었거나 연결을 허용함을 알리는 packet임. ack로는 정상적 상황을 알리는 positive ack와 비정상적 상황을 알리는 negative ack(nack)가 사용될 수 있는데, TCP/UDP는 모두 positive ack을 사용함.

3. Congestion Control

Congestion Control은 network의 capacity를 넘어서는 과부하를 방지하기 위해 전송 속도를 조정하는 것임. flow control이 해당 시스템이 가진 buffer가 overflow되지 않도록 하는 것이라면 congestion control은 network 관점에서의 control로 이해할 수 있음.

transport layer에서의 congestion control에는 open-loop과 closed-loop이 있음.

1) open-loop : 데이터를 전송하기 전부터 policy(ex. window size 등)를 미리 설정하여 congestion을 static하게 방지하는 방식.

2) closed-loop : congestion이 실제로 발생하면 이를 감지하여 dynamic하게 조정함. 예를 들어, lost packet의 발생을 확인하여 congestion 발생 여부를 확인함.

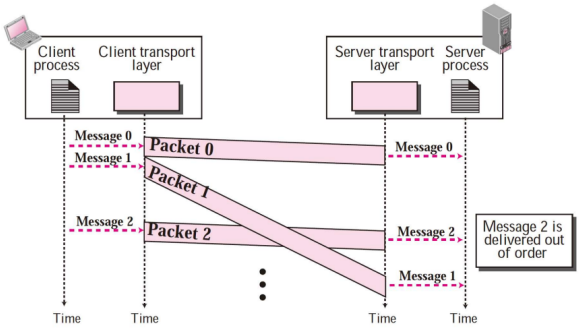
3.1.3. Connectionless/Connection-oriented Service

transport의 service는 connectionless service와 connection-oriented service로 나뉨. 두 방식 중 뭐가 항상 더 낫다기보다는 application의 요구사항에 따라 선택됨.

1. Connectionless Service

Connectionless Service는 각 packet이 independence하게 전송되는 방식임. 즉, 각 packet은 각자도생하며 전달되고, out-of-order인 packet들을 transport layer가 재조립하는 대신 application layer에서 처리하도록 단순히 넘김.

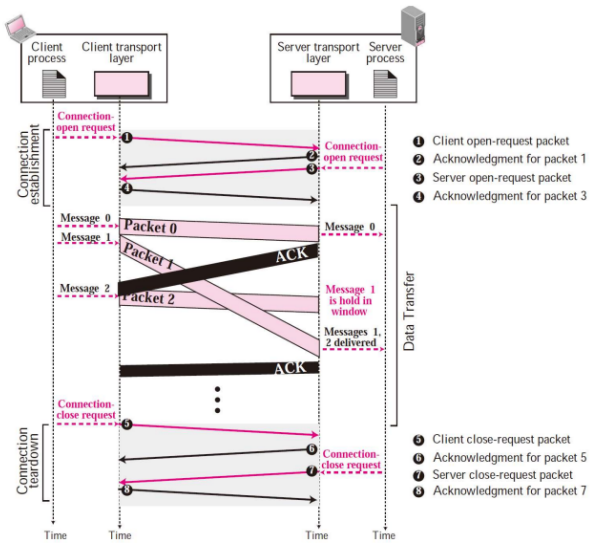
예를 들어, UDP가 있음.



2. Connection-oriented Service

Connection-oriented Service는 client와 server가 packet 전송 전에 request, ack를 주고받으며 connection을 생성(establish)함. 이후 out-of-order로 전달받더라도 입력받은 packet은 모두 buffer에 저장하는 식으로 순서를 맞춰 application layer에 넘김. packet 송수신 이후 connection을 제거함.

예를 들어, TCP가 있음.



network layer에서의 connectionless 및 connection-oriented service와 잘 구분하자.

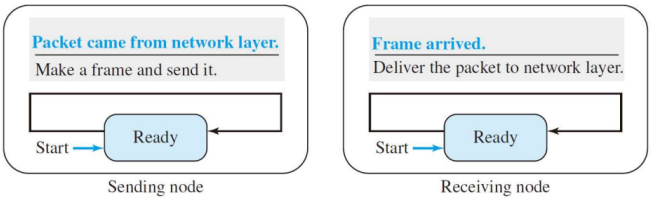
3.2. Transport Layer Protocols

Connection-oriented Service의 프로토콜들을 알아보자. 이는 flow/error control을 수행하는 protocol로, data-link layer의 DLC가 사용하는 protocol과 동일한 방식임.

3.2.1. Simple

Simple은 flow/error control을 수행하지 않는 protocol로, 별다른 이름이 존재하지 않음.

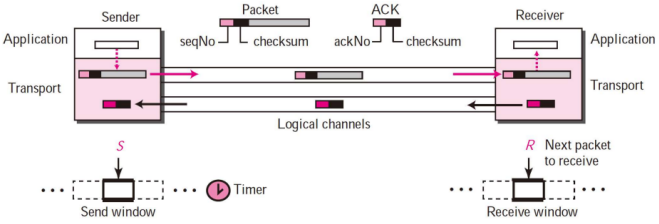
송신자는 상위 layer의 packet으로 packet을 구성해서 계속해서 보내기만 하고, 수신자는 해당 packet을 받아서 상위 layer로 전달하기만 함.



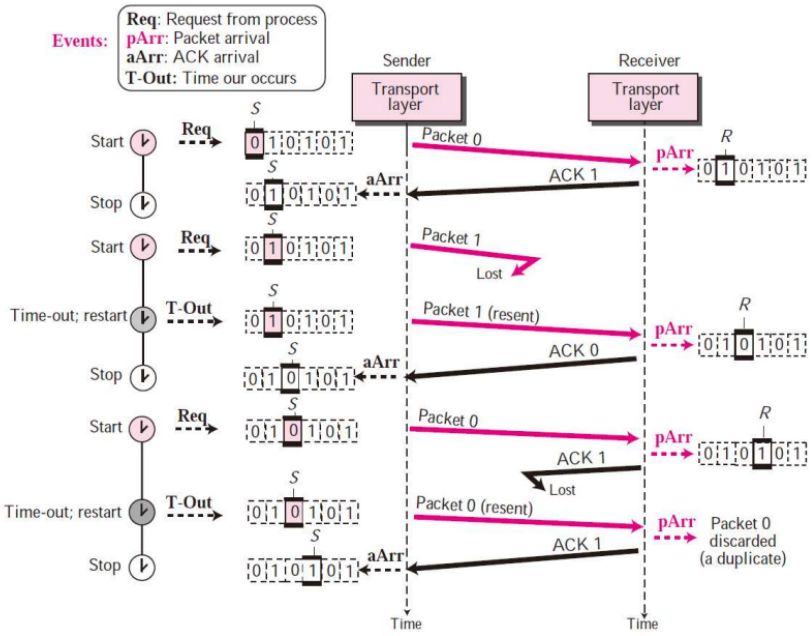
3.2.2. Stop-and-Wait

1. Stop-and-Wait
Stop-and-Wait는 송신자가 하나의 packet을 전송한 뒤 수신자로부터 ack를 전달받을 때까지 대기하는 protocol임.

stop-and-wait에서 송신자와 수신자는 각각 크기 1의 window를 사용함. 여기에서 Window는 packet의 sequence number를 저장하여 송신자/수신자 각각에서 packet 송수신에 활용하도록 하는 개념임.



그 동작 과정은 다음과 같음. 송신자는 한 번에 하나의 packet을 전송하고, 즉시 timer를 시작하여 시간을 측정함. timer가 expire되기 전에 수신자로부터 ack가 도착하면, 다음 packet을 전송하고 timer를 초기화함. timer가 expire되면, 송신자는 packet이 소실되었거나 error가 발생한 것으로 판단하여 해당 packet을 한번 더 전송함.



2. Sequence/Ack Number

packet과 ack 모두에 소실 또는 error가 발생할 수 있음. 즉, packet 전송에 대한 경우는 아래와 같이 3가지가 있음.

1. packet과 ack가 올바르게 전송된 경우.
2. packet이 올바르게 전송되지 않은 경우. packet이 재전송됨.
3. packet은 올바르게 전송되었는데 ack가 올바르게 전송되지 않은 경우.

문제는 ask가 제대로 전송되지 않은 경우임. 송신자는 ack를 받지 못했으므로 이전에 전송된 packet이 다시 전송되게 되는데, 수신자 측에서는 이 packet이 재전송된 것인지를 알 방법이 없음. 그래서 sequence/ack number를 사용하여 각 packet/ack의 순서 정보를 나타냄.

Sequence Number는 packet의 순서에 대한 번호이고, Ack Number는 ack의 순서에 대한 번호임. 3번 경우에 대해 packet이 재전송된 것인지만 파악하면 되므로, sequence nubmer는 0과 1만을 값으로 가지면 됨. ack number는 수신자 입장에서 다음에 받아야 하는 packet의 sequence number를 나타냄. 예를 들어, 현재 전달받은 packet의 sequence number가 1이면, ack number는 0이 됨. 당연하게도 0과 1만 사용하므로 1비트로 number를 표기할 수 있음.

3. 성능

stop-and-wait은 channel이 두껍고 길수록 비효율적임.

channel을 pipe처럼 생각할 수 있음. channel이 두껍다(thick)는 것은 channel이 큰 bandwidth(높은 data rate)를 가졌다는 것이고, 길다(long)는 것은 수신자까지의 왕복 지연(round-tip delay. 시간.)이 길다는 것임.

Bandwidth-Delay Product는 channel이 ack를 기다리는 동안 전송할 수 있는 data(bit)의 개수임. data rate과 delay time(여기서는 왕복 시간)을 곱해 bandwidth-delay product를 계산할 수 있음.

당연하게도 channel이 두껍고 커서 bandwidth-delay product가 클수록, ack의 수신을 기다리며 발생하는 낭비가 심함.

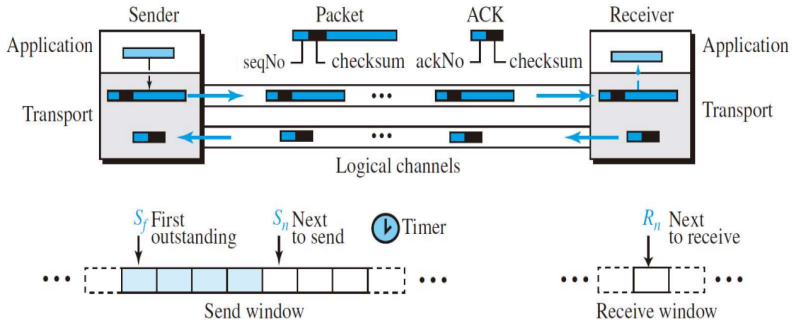
stop-and-wait은 window size가 1인 GBN임.

3.2.3. Go-Back-N

1. Go-Back-N

Go-Back-N(GBN)은 송신자가 여러 개의 packet을 전송하다가 수신자로부터 ack를 받지 못하면 ack를 받지 못한 지점부터 다시 전송하는 protocol임.

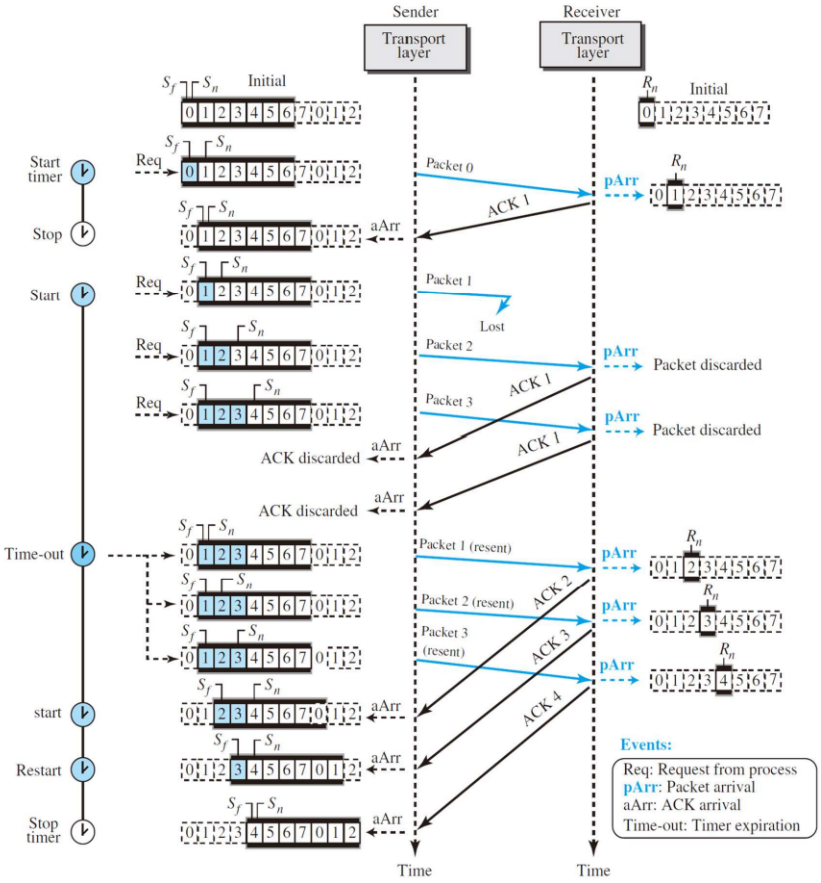
GBN에서 송신자는 특정 길이의 window를 사용하고, 수신자는 길이 1의 window를 사용함. 이에 따라 여러 개의 packet과 ack가 channel에 동시에 존재할 수 있음.



그 동작 과정은 다음과 같음. 송신자는 send window를 사용해 전송할 packet을 관리하며, ack의 도착에 상관없이 window의 packet을 계속 전송함. 이때 timer를 사용함. 유효한 ack(이미 과거에 받은 ack는 무시함.)를 받으면 timer를 초기화하고, n에 대한 ack를 받았다면 n-1까지 정상적으로 전송된 것이므로 해당 부분까지 window를 slide함. timer가 expire되면 table의 처음부터 다시 전송함. 이때 주의할 점은 timer는 유효한 ack를 받을 때 초기화된다는 것임. 즉, window를 slide할 때 timer가 초기화됨.

수신자는 receive window를 사용해 packet을 전달받는 packet을 관리함. R_n에 해당하는 packet이 아니면 전부 무시하고, R_n에 해당하는 packet이면 해당 packet 바로 다음 packet의 sequence number를 ack로 보냄.

이때 ack는 누적(cumulative)으로 처리됨.



2. Sequence/Ack Number

sequence/ack number로 m bit를 사용한다면, sequence/ack number는 mod-2^m에서 정의됨. 즉, $0 \sim 2^m - 1$의 값을 가지고, 해당 값을 넘어가면 다시 0부터 시작함. GBN에서는 여러 개의 packet을 구분해야 하므로 여러 개의 bit를 사용함.

stop-and-wait에서와 마찬가지로 sequence number는 packet 번호를 나타내고, ack number는 수신자 입장에서 다음에 받아야 하는 packet의 sequence number를 나타냄.

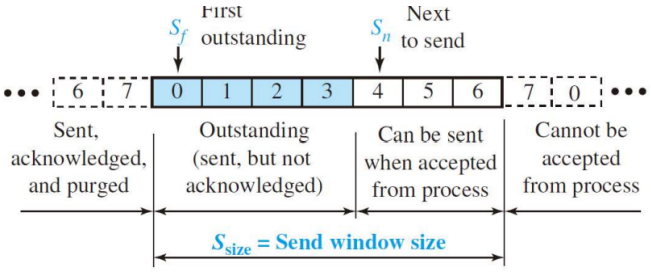
3. Send Window

Send Window는 송신자가 사용하는 window로, 아래와 같이 총 4가지 영역으로 나뉨.

1) window 왼쪽 : 전송되었고 ack를 받은 packet들.
2) window에서 색칠된 부분 : 전송되었고 ack를 받지 못한 packet들. 재전송을 위해 해당 packet의 복사본이 저장됨.
3) window에서 색칠되지 않은 부분 : 전송이 준비되지 않아 아직 전송하지 않은 packet들.
4) window 오른쪽 : window가 오지 않아 고려 대상이 아닌 packet들.

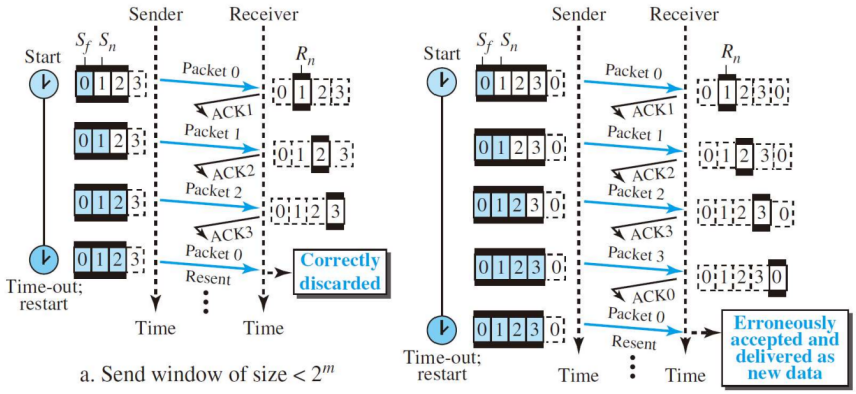
send window의 사용을 위해서는 3가지 변수가 필요함. S_f는 window의 시작 위치를 가리키고, S_n은 다음에 전송할 packet을 가리키고, S_{size}는 window의 size를 나타냄.

ack를 받으면 send window는 오른쪽으로 해당 ack만큼 slide됨.



send window의 최대 크기는 $2^m - 1$(전체 sequence number 개수보다 1 작음.)임. 만약 window의 크기가 2^m 이상이라면, send window 안에 전체 sequence number가 모두 들어오는 상황이 발생할 수 있음. 앞에서 설명한 것처럼 window 내의 packet을 전부 정상적으로 전송했는데 ack가 전달되지 못해 timer가 expire되면 packet은 window의 처음에 해당하는 것부터 다시 전송됨. 이때 수신자 입장에서는 바로 수신받은 다음 number의 packet이 전송되는 것이므로, 처음부터 다시 전송되는 것인지, 그 다음 packet이 전송되는 것인지를 구분할 수 없음. 즉, window의 크기가 전체 sequence number의 개수보다는 작아야 이런 상황을 방지할 수 있음. 예를 들어, sequence number가 3까지 있는데 window size가 3이라면, 송신자 측에서 0부터 3까지의 packet을 전부 전송했는데 ack가 오지 않아 처음부터 전송하는 경우, 수신자 입장에서는 전송된 0번째 packet이 그 다음 packet인지, 다시 보내는 packet인지 알 수 없음.

여기에서도 문제가 발생하는 지점은 ack가 제대로 전달되지 않는 경우임.

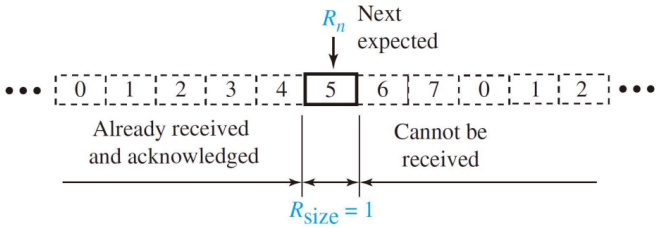


4. Receive Window

Receive Window는 수신자가 사용하는 window로, 항상 크기가 1임. 아래와 같이 구성되는데, 왼쪽은 packet을 수신받아 ack를 전송한 부분, 오른쪽은 packet을 수신받지 못한 부분임.

receive window의 사용을 위해서는 현재 수신받아야 하는 packet을 가리키는 변수 R_n 하나만을 사용하면 됨.

수신자가 packet을 받으면, receive window에서는 R_n에 해당하는 packet일 때에만 R_n을 한 칸 slide하고, 그렇지 않은 경우에는 해당 packet을 무시함.



5. 성능

stop-and-wait에서의 ack의 수신을 기다리며 발생하는 낭비는 해결하지만, timer가 expire될 때 window

의 모든 packet을 전부 다시 전송하므로 noise가 많은 channel에서는 비효율적임.

3.2.4. Selective Repeat

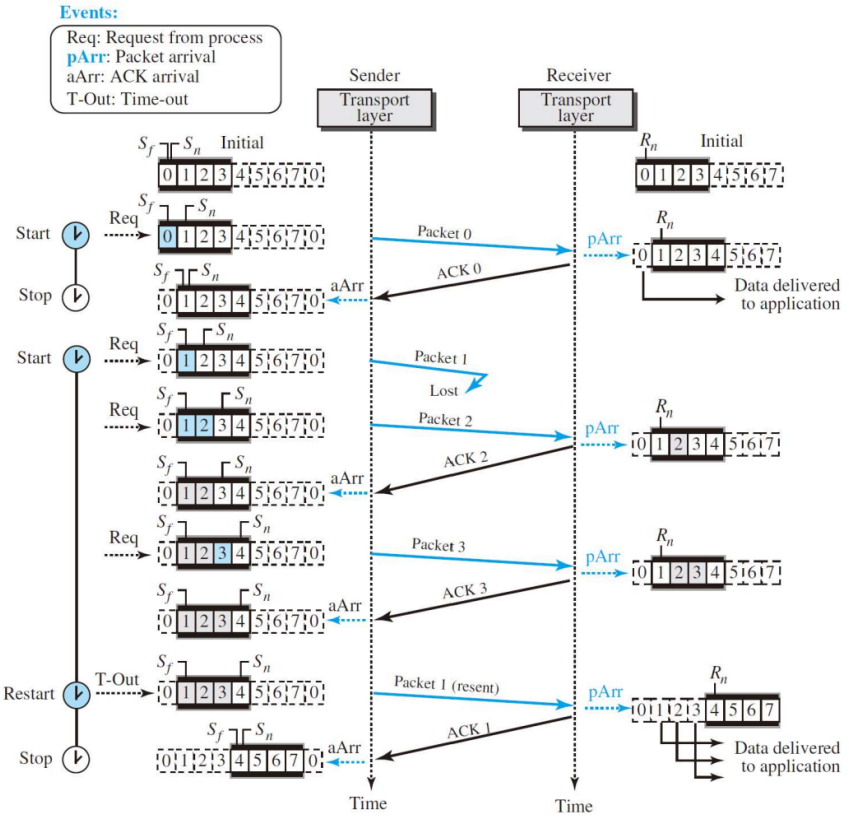
1. Selective Repeat

Selective Repeat(SR)는 GBN과 유사하지만, window의 모든 packet을 전부 다시 전송하는 대신 손상된 packet만을 선택적으로(selective하게) 재전송하는 protocol임.

GBN과 유사하므로, 모든 내용 대신 GBN과의 차이점을 중심으로 정리함.

동작 과정은 다음과 같음. 송신자는 timer를 시작하면서 window 내의 packet을 하나씩 전송함. 수신자로부터 ack를 받으면 해당 number를 ack 처리함. window의 시작점부터 연속적으로 ack를 받았다면 window를 slide하고 timer를 초기화함. timer가 expire되면 window의 앞쪽부터 ack되지 않은 것들을 다시 전송함. 이때 주의할 점은, ack를 받을 때 timer를 초기화하는 것이 아니라, window를 slide할 때 timer를 초기화한다는 것임.

수신자는 window에서 전달받은 packet을 받은 것으로 표기하고, 해당 sequence number를 ack로 보냄. 이때 과거에 이미 전달받은 packet이라면 discard하고, ack를 재전송함. window의 시작점부터 연속적으로 packet이 모두 모이면 해당 부분을 상위 layer로 전달하고 window를 slide함.



이론적으로 timer는 packet 별로 존재해서 timer가 expire된 packet을 다시 전송하도록 함. 하지만 SR을 실제로 구현하는 대부분의 protocol들에서는 하나의 timer만 사용함.

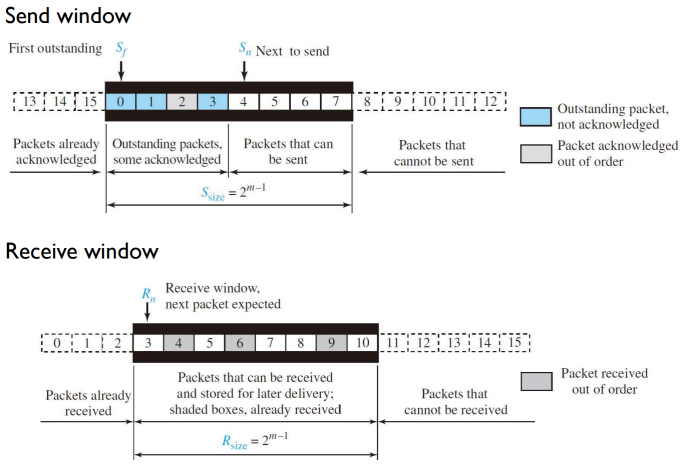
2. Window

send window와 receive window를 사용함. 여기서 send/receive window는 크기가 같음.

send window는 최대 크기가 달라진 것 외에는 GBN과 형태는 동일함. 다만, ack를 받으면 해당 packet만을 ack된 것으로 처리함. window의 시작점부터 연속적으로 ack를 받았다면 window를 slide함.

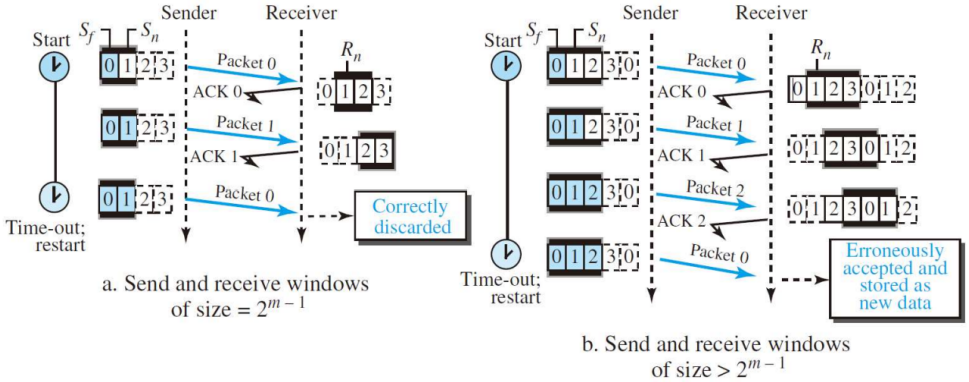
receive window는 최대 크기가 send window와 맞춰짐. 수신자는 순서에 상관없이 packet을 전달받으면 일단 window에 넣음. reveive window의 크기가 send window와 같기 때문에 packet의 순서가 뒤바뀌어 오더라도 전부 모일 때까지 적절히 저장해 둘 수 있음. window의 시작점부터 연속적으로 packet이 모두

모이면 해당 부분을 상위 layer로 전달하고 window를 slide함.



sequence/ack number로 m bit를 사용한다면, window의 최대 크기는 2^{m-1}(전체 sequence number 개수의 절반)임. 만약 window의 최대 크기가 이 값 이상이라면, 송신자 측에서 timer의 expire에 의해 packet을 맨 앞부터 재전송하는 경우에 receiver window가 이미 slide되어 있어 해당 packet이 처음부터 다시 전송되는 것인지, 그 다음 packet이 전송되는 것인지를 구분할 수 없음. 즉, window의 크기가 전체 sequence number 개수의 절반보다 같거나 작아야 이런 상황을 방지할 수 있음.

여기에서도 문제가 발생하는 지점은 ack가 제대로 전달되지 않는 경우임.



3. Acknowledgment

앞에서 다룬 protocol들에서와 달리, SR에서 ack number는 정상적으로 전달된 packet의 sequence number임.

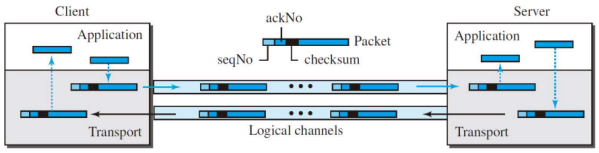
GBN에서는 ack가 cumulative하게 처리되었지만, SR에서는 ack가 cumulative하게 처리되지 않고 packet마다 모두 전송됨.

3.2.5. Piggybacking

Piggybacking은 packet에 ack를 담아서 전송하는 기법임.

실세계에서는 주로 양방향(Bidirectional) 통신이 빈번하므로 상호 간에 packet과 ack를 주고받아야 하고, 이때 piggybacking을 사용할 수 있음.

GBN은 piggybacking을 사용해 bidirectional하게 구현된다고 함.



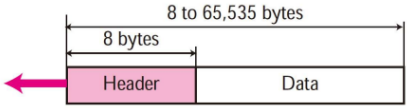
3.3. UDP

3.3.1. UDP

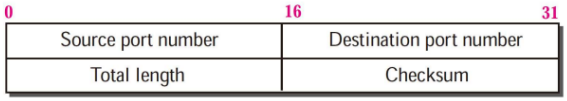
1. UDP

UDP(User Datagram Protocol)는 데이터를 빠르고 간단하게 전송하는 데에 주로 사용되는 transport layer의 protocol임.

UDP packet은 User Datagram이라고 함. datagram은 아래와 같이 8바이트 크기의 고정 길이 header와 data로 구성됨. 여기에서 length는 header와 data를 모두 포함하는 user datagram의 전체 길이를 바이트 단위로 나타내고, 해당 field가 16비트이므로 user datagram은 8 ~ 65535 바이트의 크기를 가질 수 있음.



a. UDP user datagram



b. Header format

UDP는 packet을 빠르고 간단하게 전송하므로, delay에 민감한 real-time application이나, multicasting 등에 주로 사용됨.

2. UDP Well-known Port

UDP에서 사용하는 well known port들로는 아래와 같은 것들이 있음.

Port	*Protocol*	*Description*
7	Echo	Echoes a received datagram back to the sender
9	Discard	Discards any datagram that is received
11	Users	Active users
13	Daytime	Returns the date and the time
17	Quote	Returns a quote of the day
19	Chargen	Returns a string of characters
53	Domain	Domain Name Service (DNS)
67	Bootps	Server port to download bootstrap information
68	Bootpc	Client port to download bootstrap information
69	TFTP	Trivial File Transfer Protocol
111	RPC	Remote Procedure Call
123	NTP	Network Time Protocol
161	SNMP	Simple Network Management Protocol
162	SNMP	Simple Network Management Protocol (trap)

3. UDP Service

UDP는 connectionless service를 제공함. 즉, user datagram들은 서로 independent함. 즉, user datagram은 전송 시에 numbering하지 않으므로 application layer에서 알아서 순서를 관리해야 함.

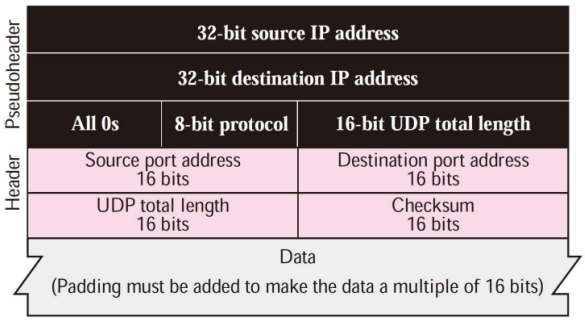
checksum을 제외한 flow/error control을 수행하지 않음. UDP를 활용한 application layer에서 이를 처리해야 함.

UDP에서는 application layer와 데이터를 주고받을 때 packet을 queuing함.

UDP에서 queue가 overflow되었다거나 하는 문제 상황에는 ICMP를 활용하여 그 정보를 전달함.

4. Checksum

IP에서의 checksum과 달리, user datagram의 checksum은 아래 그림과 같이 pseudoheader, UDP header, data로 총 3가지 부분 전체에 대해 16비트 단위로 계산함. 16비트에 들어맞지 않으면 padding(전부 0)을 추가해 계산함.



pseudoheader는 해당 user datagram의 앞에 붙을 IP header의 일부를 포함하는 임시 header임. 이는 checksum 계산 시에만 활용되고 실제로 packet에 붙이지는 않음.

UDP에서의 checksum은 optional함. checksum 필드를 전부 0으로 채워 전송하여 checksum을 계산하지 않는 것으로 지정할 수 있음. 이때 checksum을 실제로 계산했는데 그 결과가 모두 0이 되는 경우에는 그 값을 모두 1로 채워 전송함.

checksum과 관련된 부분은 이전에 정리한 내용을 참고하자. 또한 checksum을 계산할 때는 10진수로 계산하는 것이 간단함. 특히 필드의 값이 153.15.8.105 등으로 존재한다면 16비트씩 잘 나누어 더하고 이후 이진수로 바꿀 수 있음.

remote 통신이 기본이지만, 앞에서 언급한 것처럼 machine 안에서의 통신인 loopback도 존재하는데 이건 UDP, TCP 모두로 가능함.

3.4. TCP

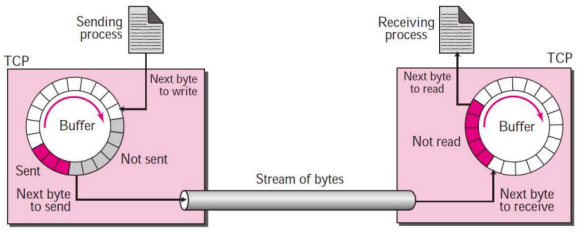
3.4.1. TCP

1. TCP

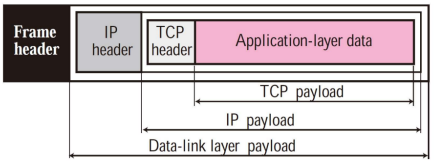
TCP(Transmission Control Protocol)는 데이터를 신뢰성 있게 전송하는 데에 주로 사용되는 transport layer의 protocol임.

TCP에서의 packet은 segment라고 한다.

TCP는 두 process 사이에 connection을 생성하여 데이터를 바이트의 stream처럼 주고받을 수 있도록 하는 stream delivery service를 지원함. 또한 이때의 두 송신자와 수신자를 통틀어 Party라고 함.



TCP에 의해 application layer의 데이터는 encapsulation됨.



2. TCP Well-known Port

TCP에서 사용하는 well known port들로는 아래와 같은 것들이 있음.

Port	Protocol	Description
7	Echo	Echoes a received datagram back to the sender
9	Discard	Discards any datagram that is received
11	Users	Active users
13	Daytime	Returns the date and the time
17	Quote	Returns a quote of the day
19	Chargen	Returns a string of characters
20 and 21	FTP	File Transfer Protocol (Data and Control)
23	TELNET	Terminal Network
25	SMTP	Simple Mail Transfer Protocol
53	DNS	Domain Name Server
67	BOOTP	Bootstrap Protocol
79	Finger	Finger
80	HTTP	Hypertext Transfer Protocol

3.4.2. Segment Format

1. TCP Numbering

TCP에서는 sequence number와 ack(acknowledgment) number를 사용하는데, 이는 segment 번호가 아니라 바이트 번호임. TCP에서는 number generator를 사용해 $0 \sim 2^{32-1}$ *중 수 하나를 임의로 골라 시작 번호로 하고, 각 데이터 바이트에 대해 번호를 부여함. 예를 들어, 시작 번호가 1057로 설정되었고 전체 데이터의 길이가 6000이라면 데이터의 각 바이트는* $1057 \sim 7056$*로 numbering됨.*

segment는 자신이 가지고 있는 첫 번째 바이트 번호를 sequence number로 하는데, 이를 ISN(Initial Sequence Number)라고 함.

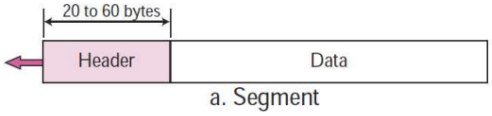
ack number는 다음으로 수신하기를 기대하는 다음 바이트 번호를 cumulative하게 나타냄. 즉, 마지막으로 수신한 바이트 번호 + 1을 값으로 하는데, 이 값이 다음 sequence number임. 또한 ack는 data와 piggyback될 수 있음.

양방향 통신에서 번호는 각 송수신 방향에서 독립적으로 생성됨.

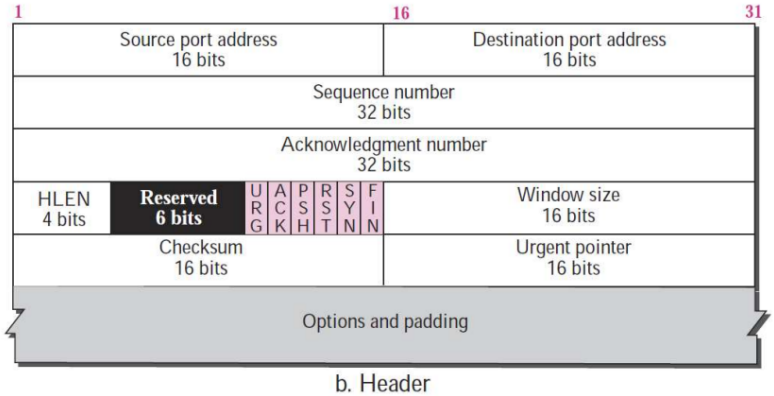
Segment 1	→	Sequence Number:	10,001	Range:	10,001	to	11,000
Segment 2	→	Sequence Number:	11,001	Range:	11,001	to	12,000
Segment 3	→	Sequence Number:	12,001	Range:	12,001	to	13,000
Segment 4	→	Sequence Number:	13,001	Range:	13,001	to	14,000
Segment 5	→	Sequence Number:	14,001	Range:	14,001	to	15,000

2. Segment Format

segment format은 아래와 같이 $20 \sim 60$ *바이트 크기의 header와 data로 구성됨. header는 기본적으로 20바이트 크기이고, option에 따라 최대 60바이트까지의 크기를 가질 수 있음.*



a. Segment



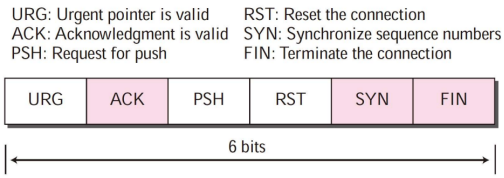
b. Header

Header Length는 header의 길이를 4바이트 단위로 나타냄. header의 크기는 option 유무에 따라 $20 \sim 60$ *바이트이므로 이 field는 5부터 15까지의 값을 가질 수 있음.*

Control field는 아래와 같이 하나 당 1비트씩 6개의 flag(control bit)로 구성됨. 각 flag의 값이 1이면

enable인 것임. flag에 의해 해당 segment의 역할이나 각 client/server의 동작이 결정되므로 잘 보자.

참고로, TCP는 efficiency를 위해 짧은 시간동안 값이 모이길 기다렸다가 한꺼번에 처리하는데, 여기에서 PSH가 enable되어 있으면 기다리는 대신 즉시 전송함. 특정 application에서는 즉각적인 처리가 필요할 수 있는데, 이 경우 PSH flag를 활용하여 push operation을 요청할 수 있음.



Window Size에는 window size를 바이트 단위로 지정함. field가 총 16비트 크기이므로 window의 최대 크기는 65535 바이트임. 여기에서의 window는 주로 receiving window(rwnd)를 나타내고, 이는 수신자에 의해 결정되므로 송신자는 이를 따라야 함.

segment의 checksum은 user datagram에서와 동일하게, pseudoheader, UDP header, data로 총 3가지 부분 전체에 대해 16비트 단위로 계산함. 16비트에 들어맞지 않으면 padding(전부 0)을 추가해 계산함. TCP에서 checksum의 사용은 의무임.

urgent pointer는 특수한 상황에 쓰이는 포인터임. 해당 flag가 enable되어 있어야 유효함.

3.4.3. TCP Connection

TCP에서는 virtual connection을 생성한 뒤 full-duplex(각 장치에서 동시에 양방향 통신 가능)로 데이터 교환을 수행함. 물론 데이터 교환 이전에 다른 party에서의 통신이 고려되어야 함. 여기에서 client와 server는 transport layer protocol로 TCP를 사용하는 application program임.

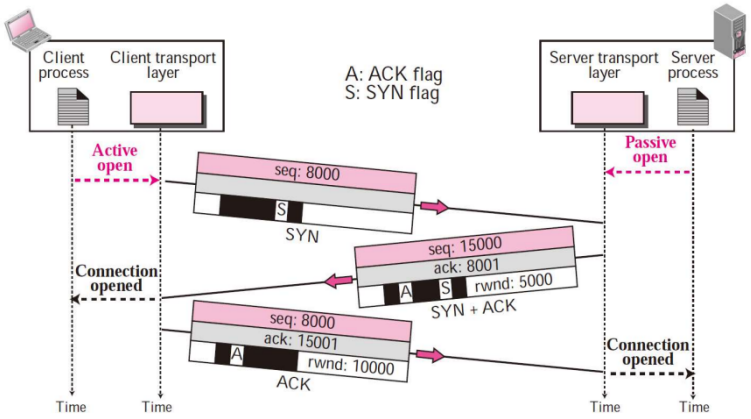
1. Connection Establishment
TCP에서는 Three-way Handshaking을 통해 connection을 생성함.

three-way handshaking 이전에 server는 자신의 TCP에게 connection 생성이 준비되었음을 알림. 이런 요청을 Passive Open이라고 함. server는 client로부터의 connection을 수락할 수 있지만, 자신이 먼저 connection을 요청할 수는 없음. 반면 client에서는 직접 connection을 요청하는데, 이런 것을 Active Open이라고 함.

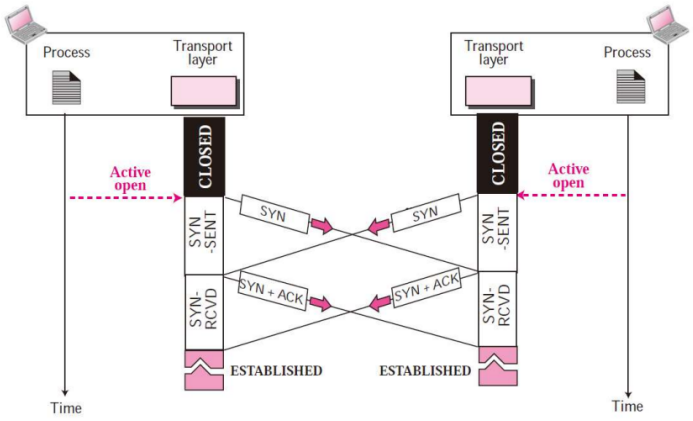
connection 생성을 위한 three-way handshaking은 syn(synchronization) segment와 ack segment를 활용해 아래와 같은 과정으로 수행됨. 이때 각 segment에는 flag가 설정됨. connection establishment가 완료되면 양쪽에 soket이 생성됨.

1) client는 ISN을 무작위로 하나 정해, 해당 값을 sequence number로 server에게 syn을 보냄.
2) server는 syn+ack를 보냄. 이때 해당 segment에는 rwnd(receive window size) 정보가 포함되고 이는 client에 의해 활용됨.
3) client는 ack를 보냄.

이때 syn과 syn+ack는 데이터를 전달하지는 않지만 sequence number를 하나 소비하고, ack는 데이터를 전달하지 않는 경우 sequence number를 소비하지 않음. 즉, syn은 하나의 가상 바이트를 포함한다고 생각할 수 있고, ack는 추가적인 바이트를 포함하지 않는다고 생각할 수 있음. 참고로 아래의 그림에서 syn, ack는 segment format에서 중간 부분부터를 나타낸 것으로, 물론 실제로는 다른 부분도 함께 전송됨.



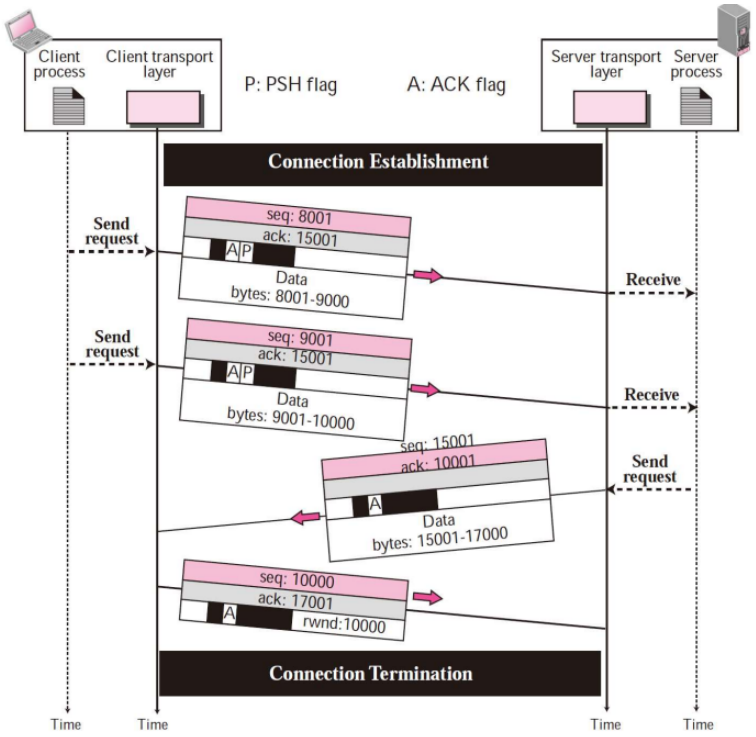
참고로, 두 process의 TCP가 모두 active open을 시도하는 경우 4 way handshake가 수행됨. 이 경우 아래 그림과 같이 서로 syn과 syn+ack를 주고받음. 이는 client-server 구조가 아니라 P2P 구조의 application에서 존재할 수 있는 드문 상황임.



2. Data Transfer

connection이 생성된 이후에는 client와 server가 bidirectional하게 데이터를 전송할 수 있고, 이때 ack는 piggyback됨.

참고로, 아래의 예시에서 client가 전송하는 segment에는 PSH flag가 enable되어 있어, 해당 데이터가 대기하는 대신 application layer에 바로 전송되도록 처리됨.



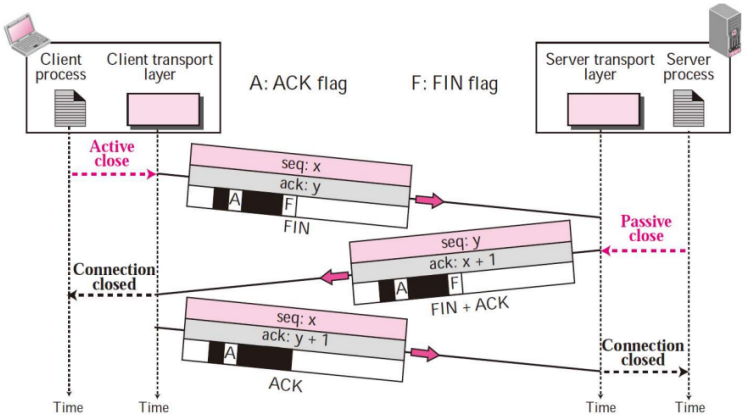
3. Connection Termination

TCP에서는 connection 제거에서도 3 Way Handshake를 사용함.

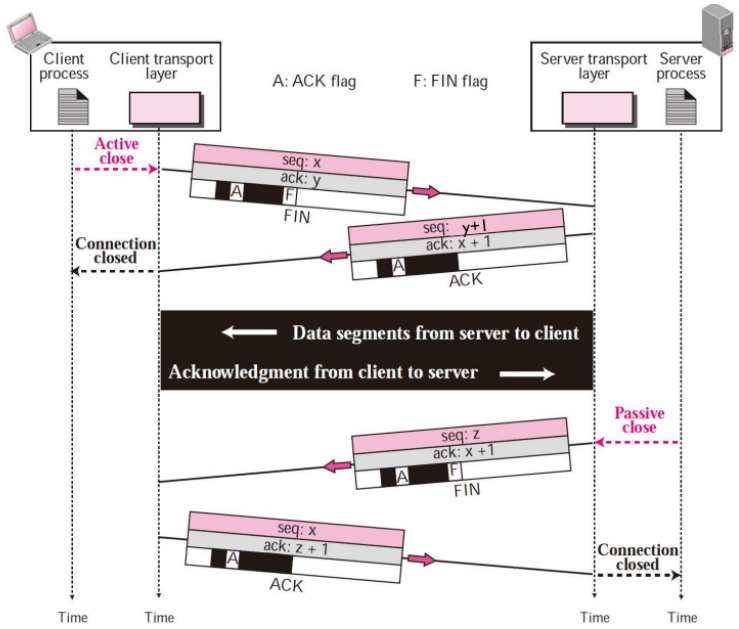
connection 제거을 위한 3 way handshake에서는 syn 대신 fin(Finish Segment)을 활용해 아래의 과정으로 수행됨. syn과 마찬가지로 fin에서도 해당 flag가 지정되고, 하나의 가상 바이트를 포함하는 것처럼 sequence number를 1 소비함.

1) client에서 마지막 데이터 chunk와 함께 fin을 전송함.
2) server에서 fin+ack를 전송함.
3) client에서 ack를 전송함.

물론 connection 제거에 timer를 사용하기도 하는데, 이는 나중에 다룸.

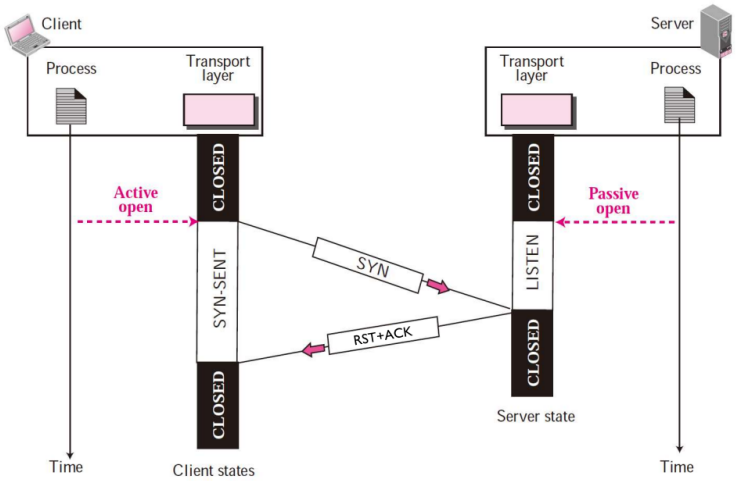


추가로, 데이터 송신은 하지 않으면서 수신만을 수행하도록 하는 Half-Close도 존재함. 즉, 데이터를 전달받고 ack를 전송하는 작업만 수행함. 이는 server와 client 모두에서 사용될 수 있음. 현재는 잘 쓰이지 않는 기법이라고 함.



4. Connection Denial

client의 connection 요청(syn)을 보냈는데 server가 passive open에 의한 LISTEN 상태가 아니면, server는 rst+ack segment를 전송함. rst segment는 connection을 reset(deny)함.



물론 실제로 full-duplex로 동작하려면 link에서도 full-duplix가 지원되어야 함.

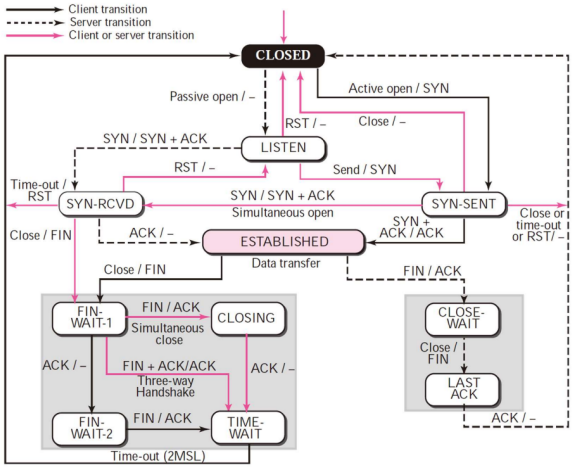
3 way handshake와 관련하여 SYN Flooding Attack이라는 공격이 존재함. 이는 하나 이상의 악의적인 공격자가 server에게 굉장히 많은 양의 syn을 보내서 리소스를 소진시키는 기법임. 각 syn은 source IP address를 조작해서 각각이 다른 client에서 온 것처럼 처리됨. 이는 Dos(Denial of Service)라고도 하고, 다수의 분산 시스템을 활용하는 Dos를 DDos(Distributed Dos)라고 함. 이에 대한 대책으로는, 특정 시간 간격 동안의 connection 요청 개수를 제한하거나, 쿠키를 사용해여 해당 connection 요청이 유효한지를 판단한 뒤 자원을 할당하도록 하는 방법이 있음.

3.4.4. State Transition Diagram

1. State Transition Diagram

connection 생성/제거, 데이터 전송 시에 발생하는 여러 event를 관리하기 위해 TCP 소프트웨어는 FSM(Finite State Machine)을 활용하여 구현되고, 이를 아래와 같이 State Transition Diagram으로 나타낼 수 있음.

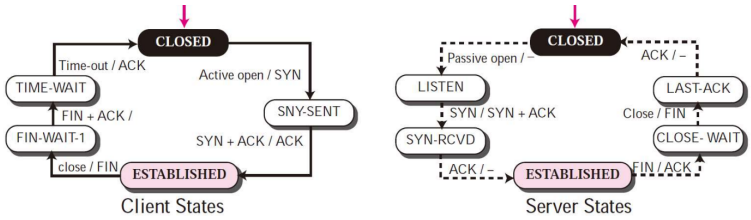
아래의 그림에서 /의 왼쪽은 TCP가 수신받는 입력이고, 오른쪽은 TCP가 송신하는 출력임. 이때 입력은 다른 TCP로부터의 입력일 수도 있지만, 해당 기기의 process(application)에서의 입력일 수도 있음. 참고로, 여기에서 ESTABLISHED는 client와 server 각각에 대해서 존재하는 서로 다른 상태를 하나로 나타낸 것임.



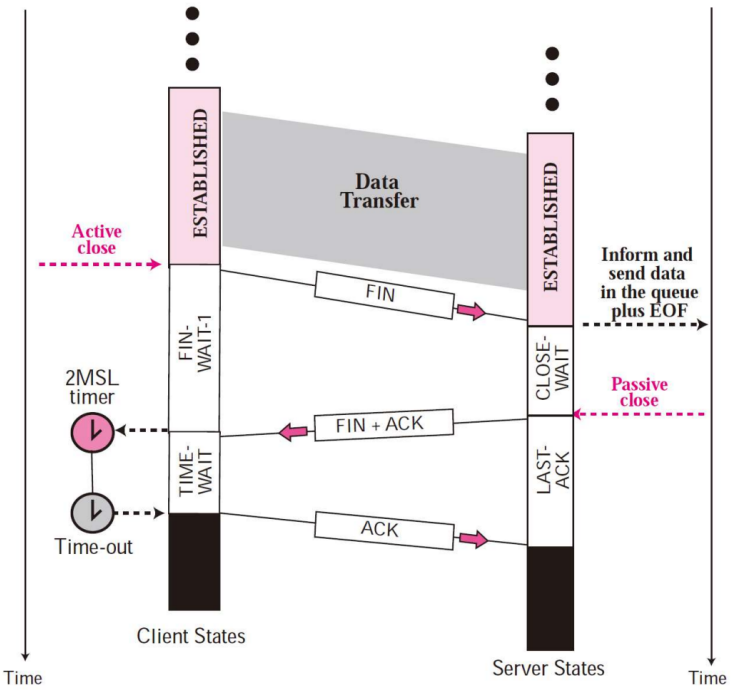
State	*Description*
CLOSED	No connection exists
LISTEN	Passive open received; waiting for SYN
SYN-SENT	SYN sent; waiting for ACK
SYN-RCVD	SYN+ACK sent; waiting for ACK
ESTABLISHED	Connection established; data transfer in progress
FIN-WAIT-1	First FIN sent; waiting for ACK
FIN-WAIT-2	ACK to first FIN received; waiting for second FIN
CLOSE-WAIT	First FIN received, ACK sent; waiting for application to close
TIME-WAIT	Second FIN received, ACK sent; waiting for 2MSL time-out
LAST-ACK	Second FIN sent; waiting for ACK
CLOSING	Both sides decided to close simultaneously

2. Connection Establishment/Termination Scenario

client와 server에서 connection을 생성하고 제거하는 작업에 대한 state transition diagram은 아래와 같음. 여기에서 passive open, active open, close 등은 process로부터의 입력임.



이를 timeline으로 나타내면 아래와 같음.



3.5. TCP : Flow/Error/Congestion Control

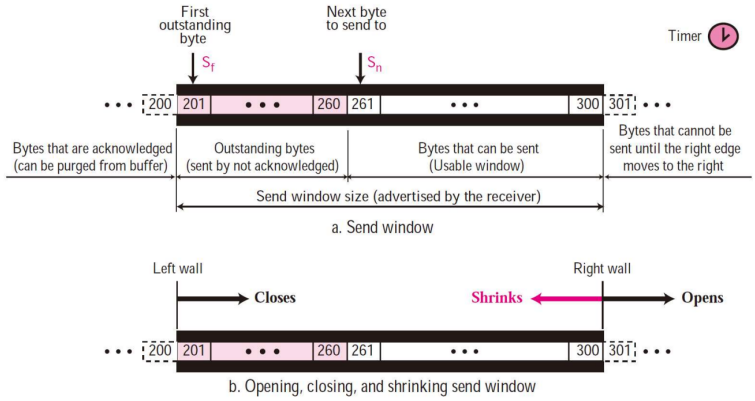
3.5.1. Window

TCP에서의 window는 아래와 같이 구성됨. 이는 SR에서의 window와 유사하지만 큰 차이점이 일부 존재함.

1. Send Window

송신자 측에서의 send window는 아래와 같은 구조로 되어 있음. 이때 send window size는 receiver와(flow control), network의 congestion(congestion control)에 의해 결정됨.

TCP의 send window는 segment 단위가 아니라 바이트 단위로 처리함. 또한 바이트 별로 timer를 사용하는 대신, 하나의 timer만을 사용함.

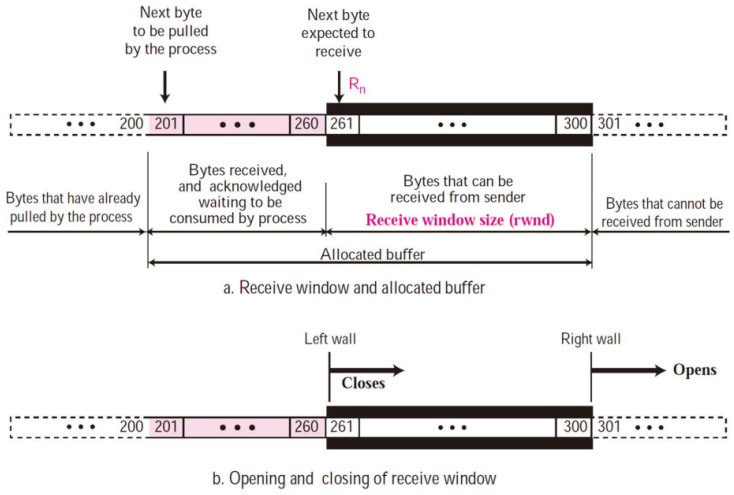


2. Receive Window

수신자 측에서의 receive window는 아래와 같은 구조로 되어 있음. receive window size는 시스템이 과부하되지 않을 정도의 크기로 지정됨.

이때 buffer를 사용하여 process가 읽어갈 수 있도록 바이트를 저장해 놓음. receive window size는 전체 buffer size에서 process로부터 읽히기를 기다리는 바이트의 개수로 계산할 수 있음. 또한 당연하게도 window size는 buffer보다 커질 수 없음.

TCP에서의 receive window는 ack를 cumulative하게 처리함. 다만 최신 TCP에서는 cumulative와 selective 모두를 활용하기도 한다고 함.



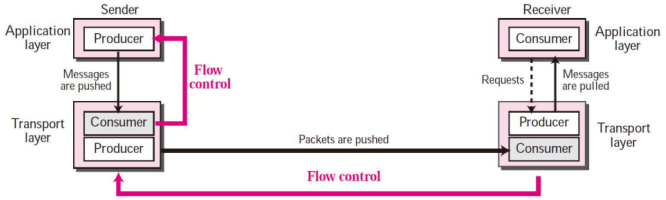
3.5.2. Flow Control

1. Flow Control

앞에서 transport layer에 대해 정리한 것처럼, TCP에서는 producer에서 데이터를 생성하는 속도와

consumer에서 데이터를 소비하는 속도를 조절하는 flow control을 수행함.

아래의 그림과 같이 대부분의 구현에서 수신자의 process가 수신자의 TCP로부터 데이터를 pull하도록 하여 별도의 flow control을 수행하지 않음. flow control은 송신 TCP가 송신 process에게, 그리고 수신 TCP가 송신 TCP에게 수행함.

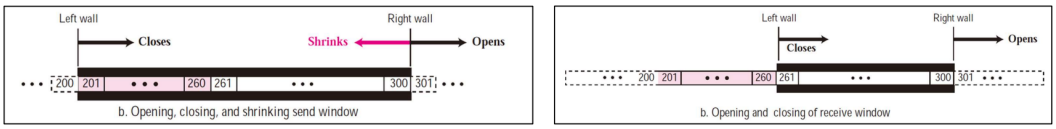


2. Window Opening/Closing

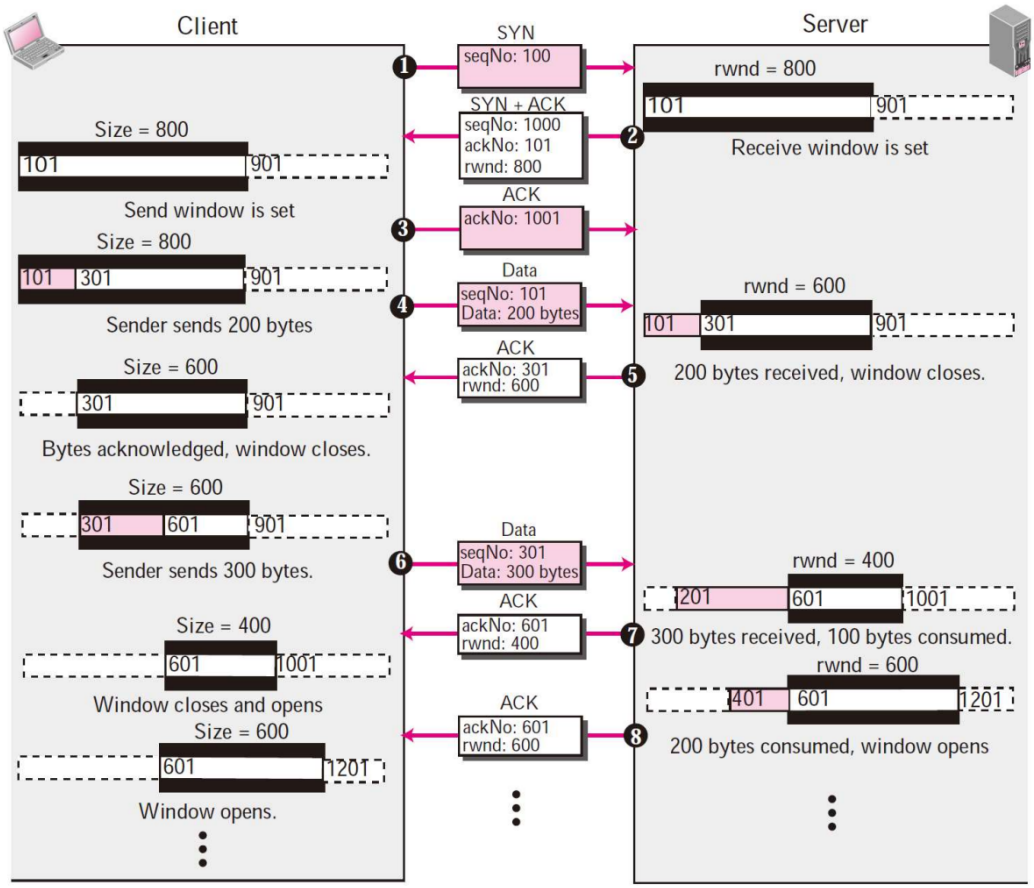
TCP에서는 flow control을 위해 송신자와 수신자 측에서 각각 자신의 window size를 조정하도록 함. 이때 open은 window의 왼쪽 벽이 오른족으로 이동하는 것이고, close는 오른쪽 벽이 오른쪽으로 이동하는 것임. send window에서는 오른쪽 벽이 왼쪽으로 이동하는 shrink도 존재하지만, 여기에서는 자세히 다루지 않음.

send window에 대한 open/close/shrink는 송신자에 의해 제어됨. 이는 ack를 받으면 close되고, 수신자에 의해 rwnd(receive window size)를 전달받으면 해당 범위까지 적절히 open됨. 이때 당연하게도 receive window size보다 send window size가 더 커질 수 없음.

receive window는 송신자로부터 바이트를 전달받으면 close되고, process에 의해 바이트가 pull되면 open됨. 또한 window는 shrink되지 않음.



즉, TCP에서 flow control은 아래와 같이 rwnd 등을 활용하여 window size를 조정하며 수행됨.



3. Silly Window Syndrome

Silly Window Syndrome은 송신자에서 데이터를 너무 느리게 생성하거나, 수신자에서 데이터를 너무 느리게 처리하는 경우, 매우 적은 크기의 데이터로 segment를 구성해 전송하게 되는 현상임. 이 경우 전송하는 데이터가 적으므로 segment 구성 및 전송에 따른 efficiency가 굉장히 떨어지게 됨.

이 문제는 송신자에 의한 경우와 수신자에 의한 경우로 나누어 고려할 수 있음.

1) 송신자에 의해 발생하는 경우

송신자의 application이 데이터를 너무 느리게 생성하는 경우 silly window syndrome이 발생할 수 있음. 예를 들어, 초당 1바이트 크기의 데이터를 생성한다면 1바이트 데이터에 대해 segment를 구성하고 전송하게 됨.

이는 Nagle's Algorithm으로 해결할 수 있음. Nagel's algorithm에서는 송신 TCP에서 application으로부터 데이터를 전달받으면 일단 segment를 구성해 전송함. 이후에는 output buffer를 사용해 application으로부터 전달받은 데이터들로 해당 buffer가 꽉 차거나, 수신자로부터 ack를 받은 경우에만 전송하도록 함.

물론 이는 RTT(Roundtrip time. 왕복 시간)이 짧은 경우나, real-time application에 대해서는 적절하지 않음.

2) 수신자에 의해 발생하는 경우

수신자의 application이 데이터를 너무 느리게 소비하는 경우 silly window syndrome이 발생할 수 있음. 예를 들어, receive window size가 4KB이고 수신받은 데이터가 4KB인데, application이 초당 1바이트의 크기를 소비한다면 송신자에게 전달되는 rwnd가 굉장히 작아짐. 이후 송신자는 아주 작은 데이터를 포함하는 segment를 전송하게 됨.

이는 Clark's Solution으로 해결할 수 있음. 이는 데이터가 도착하는 경우 ack를 바로 전송하지만, buffer에 충분한 공간(전체 또는 적어도 절반 가량)이 생길 때까지 rwnd를 0으로 전송하는 기법임.

또는 Delayed Acknowledgment로 해결할 수 있음. 이는 데이터가 도착하는 경우 ack를 바로 전송하는 대신, buffer에 충분하 공간이 생길 때까지 기다렸다가 전송하는 기법임.

3.5.3. Error Control

1. Error Contorl
앞에서 transport layer에 대해 정리한 것처럼, TCP에서는 error control에 대해 아래의 책임을 가지고, TCP로부터 TCP로의 reliable한 데이터 송수신을 지원함.

1) 전달된 packet이 discard 또는 corrupt되었는지 확인하고, 제대로 전달되지 않은 packet을 track하고 재전송함.
3) 전달된 packet이 중복된(duplicated) 것인지 검사하고 discard함.
3) packet이 전부 도착할 때까지(timer 사용) out-of-order인 packet들을 buffer에 저장함.

이를 위해 TCP에서는 checksum, ack, time-out을 활용함.

각 segment는 checksum field를 의무적으로 채워 넣어야 하고, 이를 활용해 해당 segment가 corrupt 되었는지를 검사할 수 있음.

현재의 TCP에서는 out-of-order segment를 그대로 process에게 전달하는 대신, 원래의 order로 구성하여 전달함.

2. Ack
앞에서 다룬 것처럼 ack(acknowledgment)는 데이터가 수신됨을 알리는 segment임.

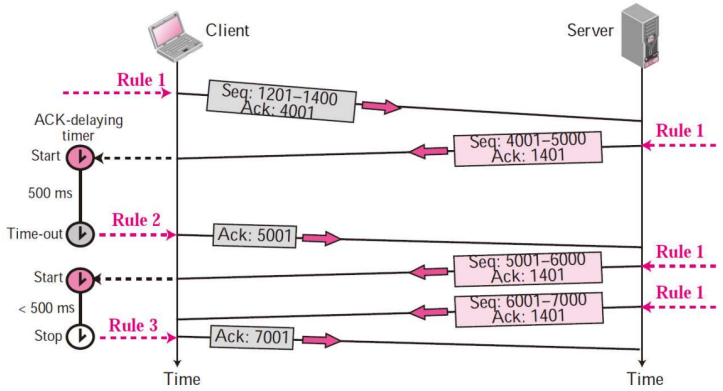
TCP의 ack는 ack(cumulative ack)와 sack(selective ack)로 나눌 수 있음. sack은 ack를 대체하는 것이 아니라, TCP header의 option으로 구현되어 out-of-order, duplicated인 데이터를 report하는 역할을 함.

3. ACK Generating Rules
수신자가 ack를 생성하여 전송하는 일반적인 rule은 아래와 같음. 이때 당연하게도 전송할 데이터가 존재한다면 ack는 piggyback하여 전송함.

rule1) segment를 정상적으로 수신받은 경우.
-> 다음으로 수신받을 것으로 기대되는 sequence number를 ack를 전송함.

rule2) 수신자가 전송할 데이터가 없고, 수신받은 segment가 in-order이고, 이전에 수신받은 데이터에 대해서 ack를 이미 모두 보낸 경우.
-> 다음 sequence가 전달되거나 time-out(주로 500ms)되기 전까지 ack 전송을 보류함. 즉, 단순히 하나의 segment만 수신받은 경우 ack 전송을 보류하여 traffic을 줄임.

이때 In-order는 이전 ack에서 전송한 ack number부터 시작하여 segment를 순서대로 입력받은 상황이고, Out-of-order는 그렇지 않은 상황임.



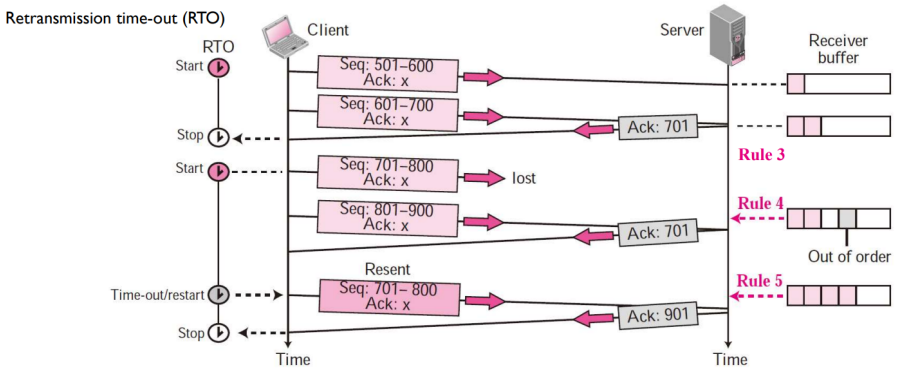
rule3) 수신받은 segment가 in-order이고, 이전에 수신받은 데이터에 대해서 아직 ack를 보내지 않은 것이 존재하는 경우.

-> 즉시 ack를 전송함. 즉, 2개보다 많은 in-order segment들이 존재하지 않도록 함.

rule4) 수신받은 segment가 out-of-order(기대된 number보다 큼)인 경우.
-> 전달받은 segment를 window에 저장하고, 즉시 ack를 전송하여 수신하기를 기대하고 있는 sequence number를 알림.

rule5) 누락되었던 segment를 수신받은 경우.
-> 즉시 다음으로 수신받을 것으로 기대되는 sequence number로 ack를 전송함. 즉, 누락되었던 segment를 수신받았다는 것을 알림.

rule6) 중복된 segment를 수신받은 경우.
-> 해당 segment를 discard하고, 다음으로 수신받을 것으로 기대되는 sequence number로 ack를 전송함.

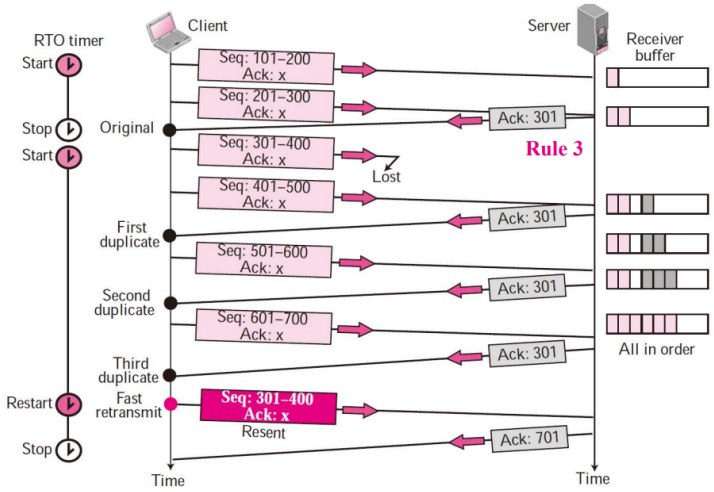


4. Fast Retransmisstion

현재 대부분의 TCP 구현은 time-out이 되기 전에 송신자가 segment를 retransmit하는 것을 허용하는데, 이를 Fast Retransmission이라고 함.

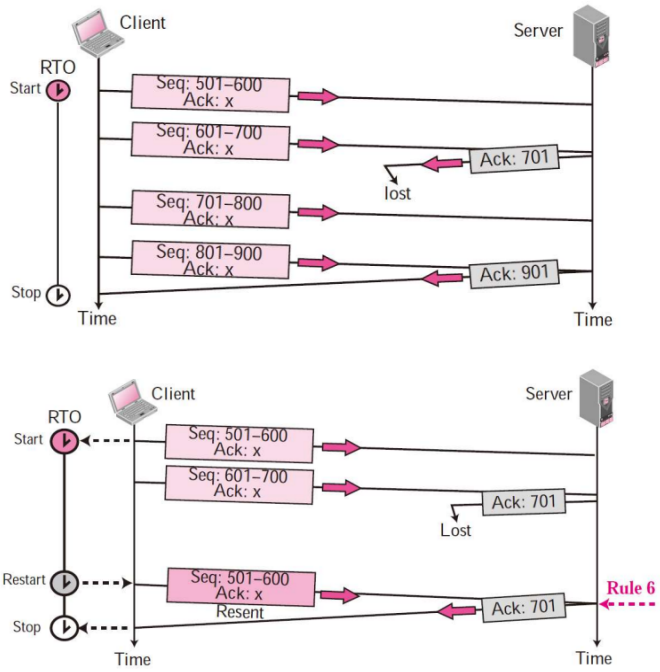
구체적으로 TCP에서는 Three Duplicated ACKS Rule을 사용함. 이는 수신자가 rule4에 의해 손실된 segment에 대해 ack를 반복적으로 보내는데, 송신자는 하나의 segment에 대해 총 3개의 동일한 ack를 전달받으면 time-out이 될 때까지 기다리는 대신 segment를 즉시 재전송하도록 하는 기법임.

즉, rule4에 의해 매번 특정 segment에 대해서 ack를 전송하는데, 3번 연속으로 out-of-order인 segment가 도착하면 해당 segment를 retransmit하게 하는 것임.



5. Lost Ack

ack의 수신에 대해서는 ack가 전송되지 않고, 이에 따라 ack는 소실되어도 재전송되지 않음. 하지만 ack가 소실되는 경우에도 rule에 의해 정상적으로 동작함.



3.5.4. Congestion Control

TCP에서는 congestion window와 congestion policy를 활용해 congestion control을 수해함. 이는 open-loop(policy를 설정하여 방지)와 closed-loop(발생 시의 대처) 모두를 활용함.

여기에서 congestion policy는 slow start, congestion avoidance, congestion detection의 총 3가지 단계(phase)를 기반으로 함. 이에 따라 송신자가 처음에는 데이터를 느린 속도로 전달하다가, 그 속도를 threshold까지 급격히 증가시키며 전달하고, threshold에 도달하면 증가 속도를 감소시킴. 이후 congestion이 감지되면 그 정도에 따라 slow start나 congestion avoidance로 되돌아감.

1. Congestion Window

Congestion Window는 송신자가 network에 한 번에 전송할 수 있는 최대 데이터량을 결정하는 window임. 이때 cwnd(congestion window size)는 그 값이 congestion policy에 의해 동적으로 결정됨.

send window size는 receive window size에 의해서도 결정되지만, network가 전송 가능한 데이터량보다 송신자가 더 많은 데이터를 전송하지 않아야 하므로 network의 congestion 또한 고려하여 결정됨. 이에 따라 send window size는 아래와 같이 rwnd와 cwnd 중 더 작은 값으로 지정됨.

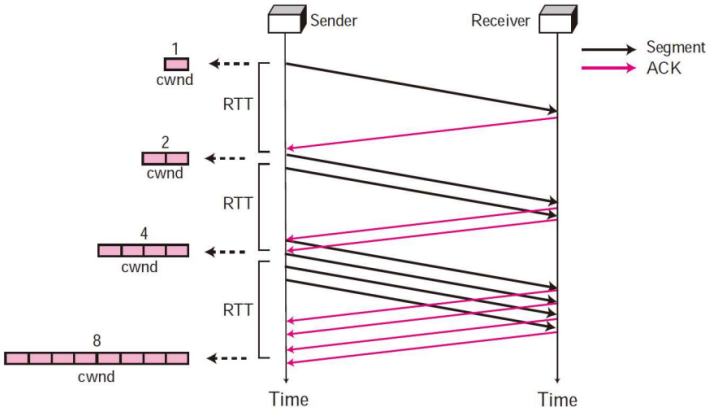
$$Sender\ Window\ Size = \min(rwnd, cwnd)$$

cwnd는 send window에 대한 개념인 것을 유의하자.

2. Slow Start

Slow Start는 처음 cwnd를 1 MSS(Maximum Segment Size)로 하고, 이후 수신자로부터 ack를 하나 받을 때마다 1 MSS만큼 cwnd를 증가시키는 기법임. 즉, 아래 그림과 같이 cwnd가 MSS로부터 시작해 지수적으로 증가함(ack를 지연 전송하는 경우를 배제했을 때.).

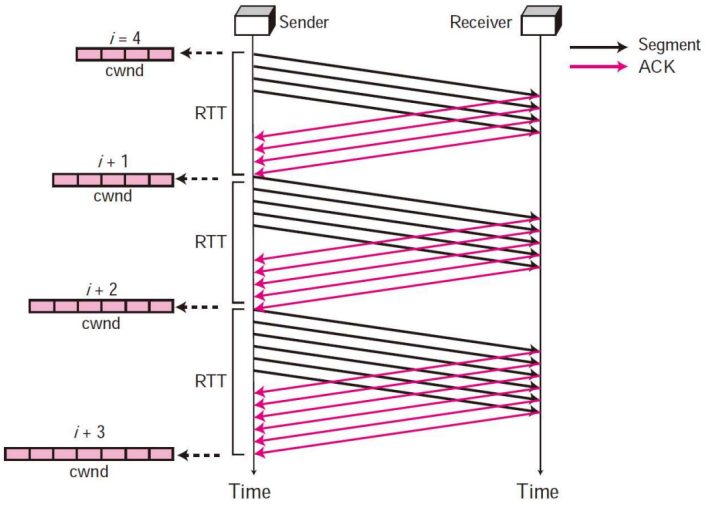
slow start에 의한 cwnd의 증가는 threshold까지만 수행되고, 이후 additive increase가 적용됨.



3. Congestion Avoidance : Additive Increase

Congestion Avoidance로는 additive increase를 사용함.

Additive Increase는 전체 send window의 segment에 대해 ack를 받을 때마다 1 MSS만큼 cwnd를 증가시키는 기법임. 즉, 1 RTT(Roundtrip Time. 왕복 시간)마다 cwnd가 1 MSS만큼 증가하므로, slow start에 의해 지수적으로 증가하던 cwnd를 가산적으로 증가시켜 congestion을 피할 수 있음.



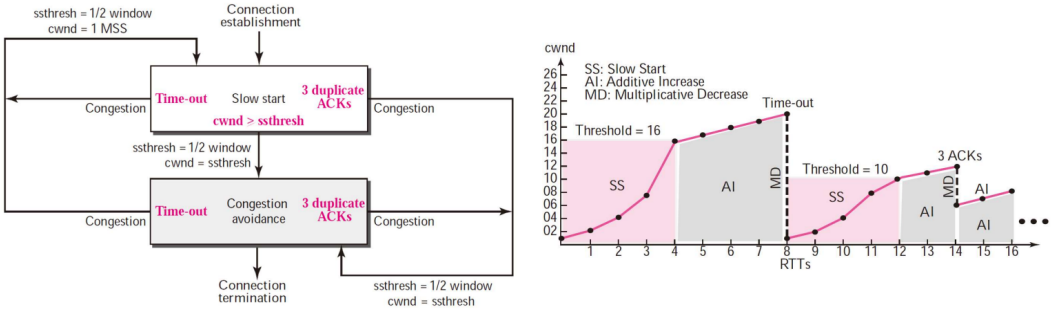
4. Congestion Detection : Multiplicative Decrease

Congestion Detection으로는 multiplicative decrease를 사용함.

Multiplicative Decrease는 congestion의 발생한 것으로 추측되면 threshold를 절반으로 줄인 뒤 cwnd를 다시 정하는 기법임. 이때 TCP에서 송신자가 congestion의 발생을 추측하려면 retransmission이 수행돼야 하고, 그런 경우로는 아래와 같이 2가지가 존재함. multiplicative decrease에는 아래와 같이 각각에 대한 동작이 정의되어 있음.

1) RTO timer가 time-out되는 경우.
congestion의 발생이 강하게 추측됨. 이 경우 threshold를 절반으로 줄이고, cwnd 값을 1 MSS로 지정한 뒤 slow start 단계부터 다시 수행하도록 함.

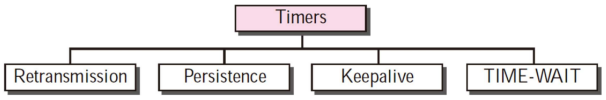
2) 3개의 동일한 ack가 수신된 경우.
congestion의 발생이 약하게 추측됨. 이 경우 threshold를 절반으로 줄이고, cwnd 값을threshold로 지정한 뒤 congestion avoidance 단계부터 다시 수행하도록 함.



3.6. TCP : 기타

3.6.1. TCP Timer : Retransmission Timer

TCP에서 사용하는 timer로는 아래와 같은 것들이 존재함.



1. Retransmission Timer

Retransmission timer는 RTO를 기다리는 timer임. RTO(Retransmission Time-out)는 한 segment에 대한 ack를 기다리는 시간임. 이에 대해서는 아래의 규칙이 존재함.

1) TCP가 송신 buffer에서 맨 앞의 segment를 전송한 경우 timer가 시작됨.
2) timer가 expire되면 송신 buffer의 맨 앞 segment부터 다시 전송하며 timer가 초기화됨.
3) ack가 적절히 수신되었으면, RTO가 초기화되고 segment들은 cumulative하게 송신 buffer에서 삭제됨.
4) 송신 buffer가 비었으면 timer를 정지함.

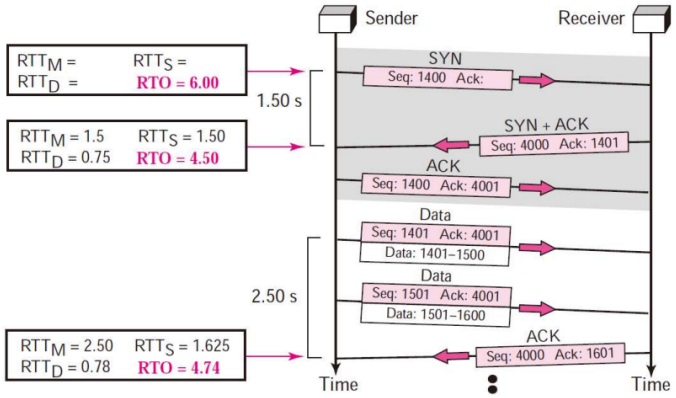
이때 RTO는 RTT(Roundtrip time. 왕복 시간)를 사용해 결정됨. 이때 계산에 사용되는 RTT에는 아래와 같이 RTT_M, RTT_S, RTT_D가 있음.

1) Measured RTT(RTT_M) : 특정 시점에 측정된 RTT. segment를 보내고 ack가 올 때까지의 시간임. 이때 여러 segment에 대해 cumulative하게 ack가 전달된 경우에도 처음 segment의 송신과 ack 수신 사이의 시간이 활용되고, 한 번에 하나의 RTT만이 측정됨.
2) Smoothed RTT(RTT_S) : 매 측정마다 RTT_M 값은 변동이 있으므로 그 가중평균으로 계산한 RTT.
3) RTT 편차(RTT_D 또는 RTT) : 특정 공식으로 계산한 RTT 편차.

$$\textit{(Init)}\ \ RTT_S = RTT_M \tag{1}$$

$$\textit{(Update)}\ \ RTT_S = (1-\alpha)\cdot RTT_S + \alpha \cdot RTT_M \tag{2}$$

$$RTO = RTT_S + 4 \times RTT \tag{3}$$



2. Retransmission과 RTO

retransmission이 실제로 발생했을 때 RTO의 계산에는 아래와 같은 사항들이 고려됨.

1) Karn's Algorithm

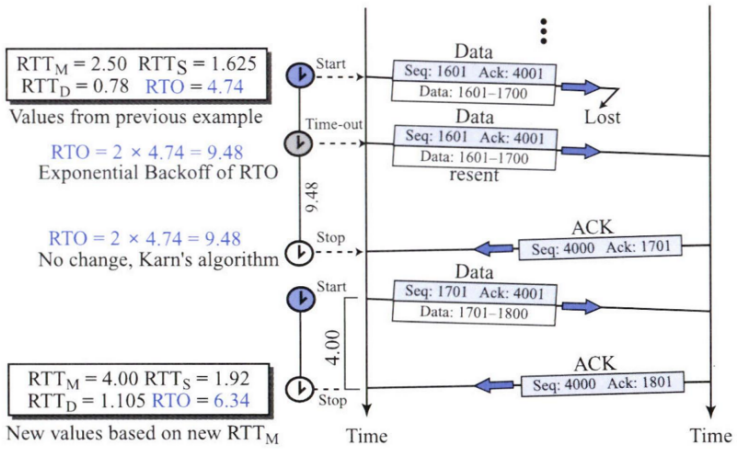
Karn's Algorithm은 RTO 계산에서 retransmit된 segment의 RTT는 고려하지 않는 기법임.

retransmit된 segment에 대한 RTT를 계산하는 경우, RTT는 retransmit 시점과 해당 retransmit된 segment의 ack가 수신된 시점으로 계산해야 함. 하지만 새로 수신받은 ack가 처음 보낸 segment의 ack인지, retransmit된 segment의 ack인지 구분할 수 없음. 이에 따라 karn's algorithm에서는 retransmit된 segment는 그냥 고려하지 않는 것으로 처리함.

2) Exponential Backoff

Exponential Backoff는 retransmission이 한 번 발생할 때마다 기존의 RTO 값을 두 배로 늘리는 기법임. 예를 들어, retransmission이 2번 발생하면 RTO 값은 4배가 되어 ack를 더 긴 시간동안 기다리게 됨.

대부분의 TCP 구현에서는 이를 사용한다고 함.



3.6.2. TCP Timer : Others

1. Persistence Timer

Persistence Timer는 송신자에게 rwnd가 0으로 전달된 경우에 대한 처리를 위한 timer임.

rwnd가 0이라는 segment를 받으면 송신자는 rwnd가 0이 아니라는 segment를 전달받을 때까지 데이터를 전송하지 못함. 하지만 rwnd가 0이 아니라는 정보를 포함하는 ack가 전달 도중 소실되는 경우, ack는 재전송되지 않으므로 양쪽 TCP가 서로의 전송을 기다리는 Deadlock이 걸림. 이에 따라 송신자는 rwnd가 0이라는 segment를 받으면 persistence timer를 시작하고, ack를 받기 전에 이 timer가 expire되면 probe라는 segment를 전송함.

Probe는 1바이트 크기의 새로운 데이터를 포함하는 특수 segment임. probve는 특정 sequence number

를 가지지만 sequence number 계산에서 제외되고, 이에 대해 ack도 전송되지 않음.

2. Keepalive Timer
Keepalive Timer는 두 TCP 사이에서의 idle connection을 방지하기 위한 timer임.

데이터 전송이 이뤄지지 않는 두 TCP에 대해, connection이 계속 존재하는 것(idle)는 낭비임. 이에 따라 대부분의 server에서는 keepalive timer를 사용하여 이 timer가 expire된 경우 connection을 종료함. 이 timer는 client로부터 segment를 받을 때마다 초기화됨.

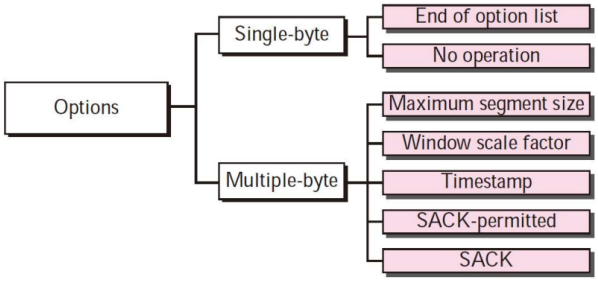
이 timer는 대체로 2시간 정도라고 함.

3. TIME-WAIT Timer
TIME-WAIT Timer는 connection 종료 시에 적절한 처리를 위해 사용되는 timer임.

TIME-WAIT Timer는 2 MSL(Maximum Segment Lifetime)로, 대체로 60초라고 함.

3.6.3. Options

앞에서 정리한 것처럼 TCP header는 최대 40바이트 크기의 option을 가질 수 있고, 그 종류는 아래와 같음.



이때 End-of-Option Option은 header에서 마지막 option의 가장 마지막 부분에 padding으로 사용되어 마지막 option임을 나타내는 1바이트 크기의 option임. 이 option 뒤에는 payload data가 위치함. 또한 No-Operation Option은 option 사이에서 빈 공간을 채우거나 4바이트로 align하기 위해 사용되는 1 바이트 크기의 option임.

1. MSS
MSS는 MSS 값을 지정하는 option임. MSS(Maximum Segment Size)는 receiver가 전달받을 수 있는 data의 최대 크기임. 이는 segment 전체의 크기가 아니라, data 영역의 크기임.

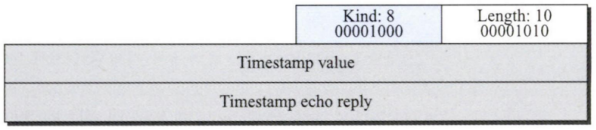
MSS는 connection 생성 시에 지정되고, 지정되지 않았다면 536바이트가 default로 지정됨.

2. Window Scale Factor
Window Scale Factor는 더 큰 window size를 지정하는 데 사용하는 option임. 기본적으로 TCP header에서는 rwnd를 16비트로 지정하므로, 지정할 수 있는 최대 size는 65535 바이트임. window scale factor를 지정하면 아래와 같이 계산되어 더 큰 window size를 활용할 수 있음.

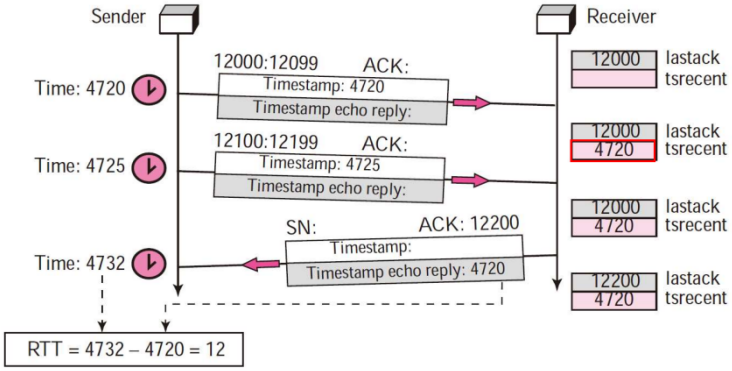
$$Window\ Size = Window\ Size\ Defined\ in\ Header \times 2^{Window\ Scale\ Factor}$$

3. Timestamp
Timestamp는 RTT 측정 및 sequence number 중복을 방지하는 데에 사용되는 option임. 아래와 같이 10바이트 크기의 format을 가짐.

만약 active open 시에 syn에 timestamp를 사용하고, 응답으로 받는 syn+ack에 timestamp가 존재하면 timestamp 사용이 허락된 것이고, timestamp가 존재하지 않으면 거절된 것으로 더 이상 timestamp를 사용할 수 없음.



아래와 같이 timestamp를 활용해 RTT를 계산할 수 있음. 여기에서 마지막으로 전송된 ack number를 저장하는 lastack 변수와, echo되지 않은 timestamp 값을 저자하는 tsrecent변수가 사용됨. 여기에서 echo는 timestamp 값을 timestamp echo reply field에 입력하는 것을 의미함. 수신자가 lastack에 해당하는 segment를 수신받으면 해당 segment의 timestamp 값을 tsrecent에 저장하고, 이후 ack를 전송할 때 tsrecent의 값을 echo함. 즉, 마지막 ack 이후 처음으로 수신받은 segment에 대해 timestamp 값을 echo해 전송함. 송신자는 수신받은 ack의 timestamp echo reply 값과 현재 clock을 비교하여 RTT를 계산함. 이때 모든 계산은 송신자의 clock을 기준으로 수행되므로 송신자와 수신자 간의 clock 동기화가 필요하지 않음.



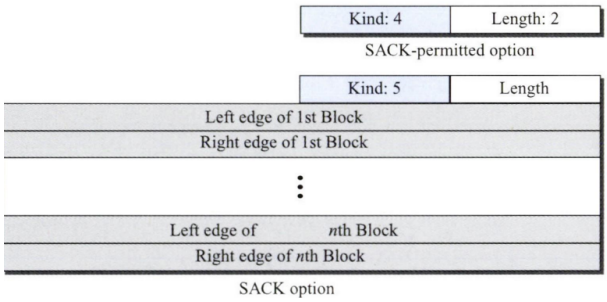
4. SACK, Sack-permitted

SACK는 sack(selective acknowledgment)를 전송할 수 있도록 하는 option임. 즉, 어떤 특정 segment가 lost, out-of-order, 또는 duplicated라는 정보를 포함함.

SACK option을 사용하려면 우선 SACK 사용에 대한 허가를 주고받아야 함. Sack-permitted는 SACK 사용에 대한 허가를 나타내는 2바이트 길이의 option임. 이는 데이터 전송 시에는 사용하지 않고, connection 생성 시에 각 end가 각각 송신하는 segment에 이 option을 사용하여 서로 SACK 사용에 대한 허가를 나타냄. 참고로 SACK option은 데이터 전송 시에만 사용할 수 있음.

SACK에서는 out-of-order segment들의 순차적인 연결인 Block의 범위를 저장함. 이 정보를 수신받은 송신자는 해당 segment들이 out-of-order로 수신자에게 도착했고, 나머지는 lost된 것으로 추측할 수 있음.

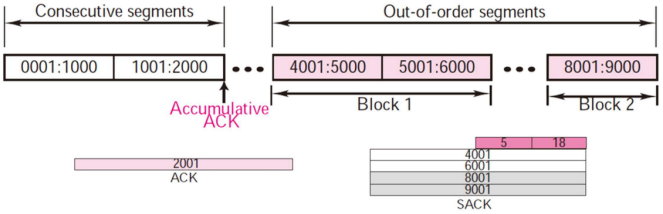
구체적으로 SACK은 아래와 같이 각각 4바이트 크기의 field로 block의 시작 number와 끝 number를 저장함. option은 최대 40바이트이므로 SACK만 사용하는 경우 총 4개 block의 정보를 저장할 수 있음. 이때 해당 구현이 되어 있는 시스템에서는 첫 번째로 작성하는 block에 duplicated segment를 지정하기도 함.



예를 들어, 아래와 같이 SACK을 구성할 수 있음.

– This means that bytes 2001 to 4000 and bytes 6001 to 8000 are lost or discarded

✓ The sender can resend only these bytes



4. Application Layer

4.1. 서론

4.1.1. 서론

1. Application Layer Protocols

application layer protocol들로는 아래와 같은 것들이 있음.

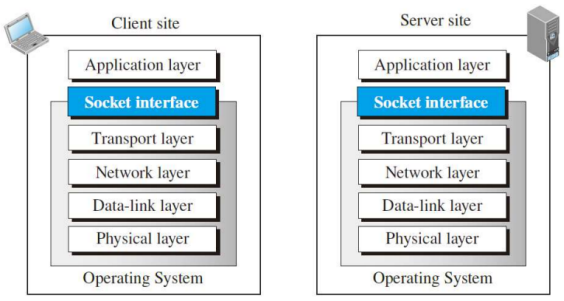
1) Standard Protocols : internet authority에 의해 표준화된 protocol들. 예를 들어, HTTP, DNS, FTP(file transfer), SSH(remote access), SIP(VoIP), SMTP/POP3(email) 등이 있음.

2) Nonstandard Protocols : private company에서 만들어 활용하는 protocol들. Zoom, Telegram, OpenVPN, Tor 등은 각자 정의한 protocol을 활용함.

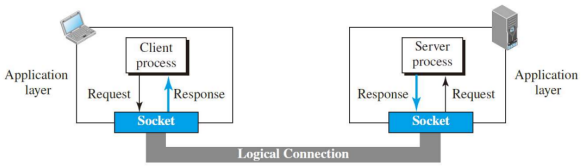
각 protocol은 그 목적에 따라 적절한 transport layer protocol을 활용함.

2. Socket Interface

Socket Interface는 1980년대 UC Berkeley에서 UNIX 환경의 일부로 개발한, process간 통신 등에 사용되는 interface임. 아래와 같이 os와 application 사이에서 동작하며, 해당 interface에 정의된 명령들을 사용하여 os에 의해 제공되는 TCP/IP 서비스들을 이용할 수 있음.



각 end의 process는 socket을 생성하고, 해당 socket에 단순히 읽고 쓰는 작업을 통해 통신할 수 있음.



interface는 두 entity 사이의 상호작용을 위해 설계된 명령들의 집합으로, 각 entity는 해당 명령들만을 고려하면 상호작용이 가능함.

4.1.2. Web

1. Web
Web 또는 World Wide Web은 전세계에 분산되어 있는 repository로, web page라고 하는 상호 연결(link) 가능한 있는 문서들을 저장함.

web page는 hypertext 또는 hypermedia임. Hypertext 문서는 그 일부가 다른 문서로의 link로 정의될 수 있는 문서이고, Hypermedia는 다른 텍스트, 이미지, 오디오, 비디오 등의 정보를 포함하는 문서로의 link를 포함하는 문서임. 즉, web page는 다른 web page로의 link를 가질 수 있음.

2. Web Client/Server
web client(browser)는 아래와 같이 3가지 부분으로 구성됨.

1) Controller : 사용자로부터의 입력, client program 등을 활용하여 특정 문서로 접근하는 작업을 수행함. 특히, interpreter를 사용하여 문서를 화면에 display함.

2) Client Protocol : HTTP, FTP 등의 application layer protocol.

3) Interpreter : HTML, java, javascript 등 문서의 종류에 따른 interpreter.

web server는 client에게 제공할 데이터를 저장하거나, 제공할 서비스와 관련된 작업을 수행함.

3. URL
URL(Uniform Resource Locator)는 각 web page 또는 web resource를 구분하는 데에 사용되는 식별자 또는 주소임. 이는 아래와 같이 4가지 부분으로 구성됨.

1) Protocol : HTTP, FTP 등.
2) Host : ip address 또는 domain name.
3) Port : client/server application이 predefine한 port number. (HTTP는 80)
4) Path : 접근하려는 시스템에서의 경로.

4.2. HTTP

4.2.1. HTTP

1. HTTP
HTTP(Hypertext Transfer Protocol)는 web에서 client와 server가 request와 response를 통해 데이터를 주고 받는 방법에 대한 protocol임. 즉, web을 구현하는 protocol으로 이해할 수 있음. HTTP는 TCP의 service를 활용함.

HTTP는 기본적으로 HTTP client가 request를 보내고, HTTP server는 그에 대한 response를 보내는 식으로 동작함. 이때 HTTP client는 일시적인 port number를 사용하는데에 반해, HTTP server는 80을 port number로 사용함.

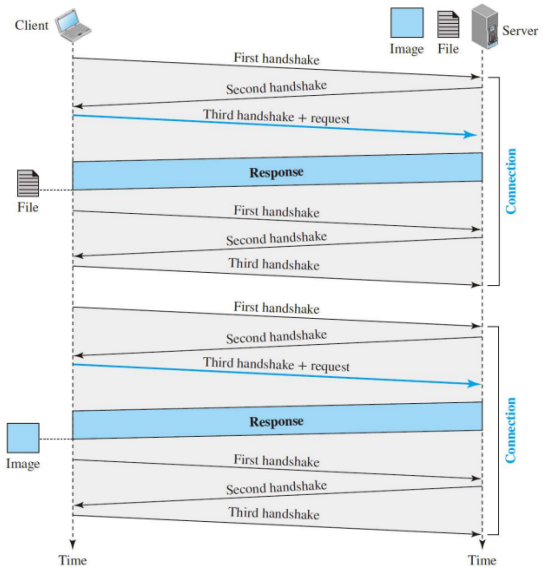
2. HTTPS
HTTPS(HTTP Secure)는 HTTP와, application layer와 transport layer의 사이에서 동작하는 security protocol인 TLS(Transport Layer Security)의 결합으로 구성됨. HTTP는 security를 고려하지 않은 채로 만들어졌고, 이후 security 위험이 증가하고 web의 규모가 커짐에 따라 security를 고려한 HTTPS가 주로 활용되고 있음.

HTTP와 비교했을 때 HTTPS는 아래와 같은 security service를 추가로 제공함. 이런 security service는 security 관점에서의 공격을 감지 및 처리하는 것으로, error control과는 차이가 있음.

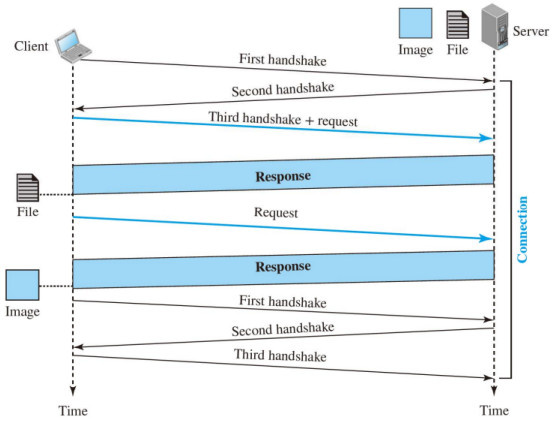
1) Authentication : 정당한 사용자인지 확인하는 것.
2) Data integrity : 데이터가 통신 중에 조작되었는지 확인하는 것.
3) Confidentiality : 데이터를 encrypt하는 것.

3. Nonpersistent Connection vs. Persistent Connection
HTTP에서의 connection은 nonpersistent와 persistent로 구분됨. HTTP/1.0에서는 nonpersistent connection을, HTTP/2.0과 HTTPS에서는 persistent connection을 사용하는 것이 default임.

1) Nonpersistent Connection
Nonpersistent Connection은 매 request와 response마다 connection을 생성하는 방식임. 즉, n개의 request/response 쌍이 있다면 connection은 n번 열리고 닫힘.



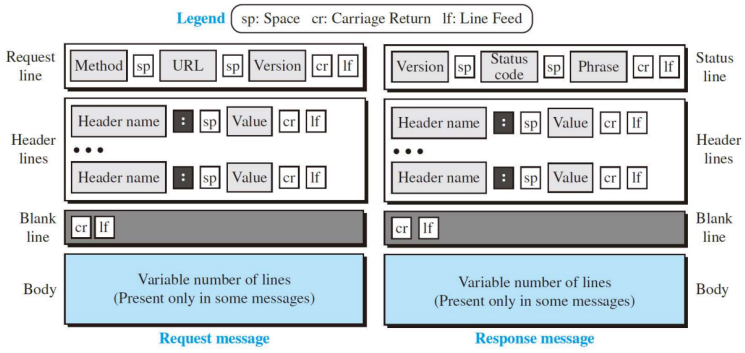
2) Persistent Connection
Persistent Connection은 request와 response 이후에도 connection을 유지하는 방식임. 시간과 resource가 절약된다고 함.



참고로, 원래 SSL(Secure Socket Layer)이 사용되었는데, 이를 기반으로 TLS가 등장했음.

4.2.2. HTTP Message

HTTP에서의 packet은 Message라고 함. request message와 response message는 각각 아래와 같은 format을 가짐. request message는 request line으로 시작하고, response message는 status line으로 시작하는 것을 제외하면, 뒤에 나오는 format은 동일함(물론 그 내용은 다름.). 이때 각 부분은 crlf(줄바꿈)에 의해 구분됨.



browser는 이런 HTTP message를 적절히 생성 및 처리함.

1. Request Message

request line은 method, URL, version(HTTP version)을 field로 구성되고, 각각은 공백 문자로 구분됨.

주요한 method들은 아래와 같음.
1) GET : client가 server로부터 resource를 받으려는(조회) 경우. 가장 많이 사용되는 method임.
2) POST : client가 server에게 데이터를 보내려는 경우.
3) PUT : resource를 업데이트하거나, 새로운 resource를 생성(client가 디테일 지정)하는 경우.
4) DELETE : resource를 제거하는 경우.

Method	*Action*
GET	Requests a document from the server
HEAD	Requests information about a document but not the document itself
PUT	Sends a document from the client to the server
POST	Sends some information from the client to the server
TRACE	Echoes the incoming request
DELETE	Removes the web page
CONNECT	Reserved
OPTIONS	Inquires about available options

header에는 0개 이상의 header line을 작성할 수 있음. header의 종류는 아래와 같고, 주로 browser의 상태 정보를 포함함. 각 header line은 header name, colon, 공백 문자, header value로 구성됨. header 중 Host로는 domain 정보, Connection으로는 persistent/nonpersistent 지정, Accept-language에는 선호 언어 지정 등을 함.

Header	*Description*
User-agent	Identifies the client program
Accept	Shows the media format the client can accept
Accept-charset	Shows the character set the client can handle
Accept-encoding	Shows the encoding scheme the client can handle
Accept-language	Shows the language the client can accept
Authorization	Shows what permissions the client has
Host	Shows the host and port number of the client
Date	Shows the current date
Upgrade	Specifies the preferred communication protocol
Cookie	Returns the cookie to the server (explained later in this section)
If-Modified-Since	Specifies if the file has been modified since a specific date

2. Response Message

status line은 version(http version), status code, phrase로 구성되고, 각각은 공백 문자로 구분됨.

status code는 3개의 숫자로 구성되며, request에 대한 처리 상태를 나타냄.
1) 100번대 : request 처리 상태를 알림.
2) 200번대 : request가 정상적으로 처리됨.

3) 300번대 : request가 다른 URL로 redirect됨.
4) 400번대 : client site에서 error가 발생했음.
5) 500번대 : server site에서 error가 발생했음.

phrase는 status code에 대한 간단한 텍스트 설명임.

동일한 방식으로 아래와 같은 header들을 작성할 수 있음. header 중 Server로는 해당 message를 생성한 server 정보를, Content-type으로는 body 영역 데이터가 가지는 type을 지정함.

Header	*Description*
Date	Shows the current date
Upgrade	Specifies the preferred communication protocol
Server	Gives information about the server
Set-Cookie	The server asks the client to save a cookie
Content-Encoding	Specifies the encoding scheme
Content-Language	Specifies the language
Content-Length	Shows the length of the document
Content-Type	Specifies the media type
Location	To ask the client to send the request to another site
Accept-Ranges	The server will accept the requested byte-ranges
Last-modified	Gives the date and time of the last change

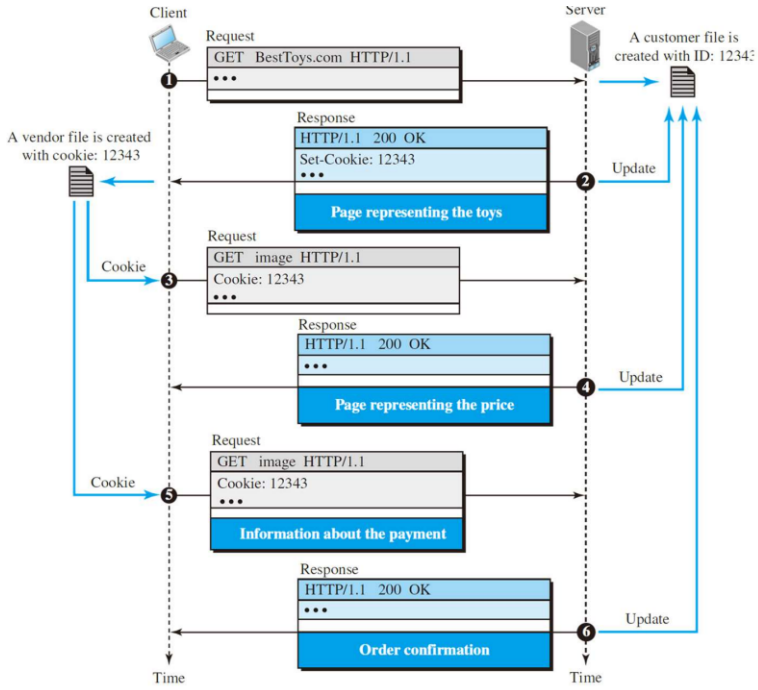
4.2.3. Efficiency in HTTP

request/response를 주고받는 통신에서의 efficiency를 위해 cookie와 proxy server 등을 활용할 수 있음.

1. Cookie
Cookie는 server나 client에서 저장해 둔 client에 대한 정보임. cookie로는 client의 domain name, 접속 정보, timestamp 등이 저장될 수 있음.

server는 response에 cookie 정보를 포함해 전달할 수 있고, browser는 이를 cookie directory에 저장해 둠. 이후 client에서 request를 보낼 때 관련 cookie가 존재하는지 확인하고, 존재하면 이를 활용하는데 이에 따라 server는 이 client가 이전에 접속했던 client라는 것을 알 수 있음.

물론 cookie도 나름의 유효 기간을 가짐.

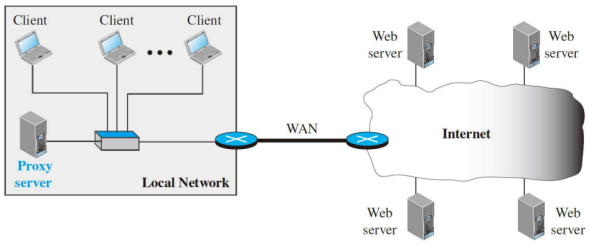


2. Proxy Server

Proxy Server는 최근 request들에 대한 response를 caching하고 있는 server임.

HTTP client가 proxy server에게 request를 보내면, proxy server는 자신의 cache를 확인하여 대응되는 response가 존재하면 전송함. response가 존재하지 않으면 original server에 전송함. 이런 caching을 통해 traffic과 latency를 줄임.

당연하게도 proxy server는 client site에 물리적으로 위치함.



4.3. DNS

4.3.1. DNS

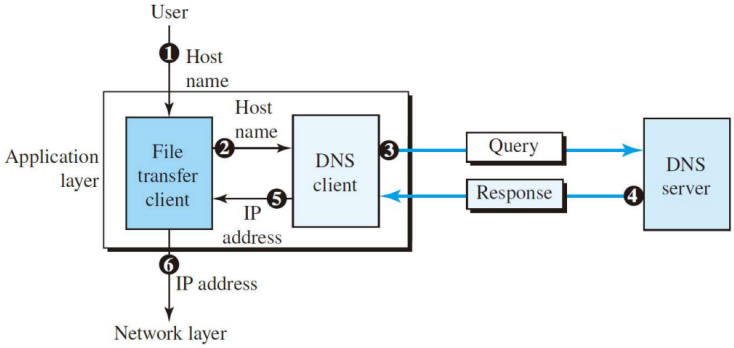
1. DNS

DNS(Domain Name System)는 domain name과 ip address를 mapping하는 system 또는 protocol임.

TCP/IP에서는 각 host를 ip address로 구분하지만, 사용자 입장에서는 이런 숫자의 집합을 다루는 것은 번거롭고 실수하기 쉬움. 이에 따라 DNS에서는 ip address에 대응되는 이름인 Domain Name을 활용하여 host를 구분할 수 있도록 함. 예를 들어, 구글 홈페이지의 doamin name은 www.google.com임.

DNS에서는 아래 그림과 같이 동작함. 사용자가 domain name을 사용하면 application layer에서 DNS client를 통해 DNS sever로 query를 보내고, 이후 응답받은 ip address로 domain name을 대치하여 하위 layer로 packet을 전달함.

각 host는 ip address를 할당받을 때 DNS server의 address도 함께 전달받음.

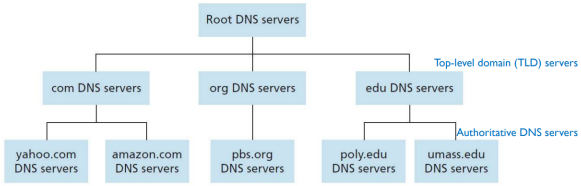


이때 ip address와 domain name 사이의 mapping은 하나의 central server에서 모두 저장하기에는 그 크기가 너무 크므로, 이를 분산 저장하고 host가 적절한 server에 접근하여 mapping 정보를 얻도록 함.

2. Domain Namespace

Namespace는 ip address에 mapping되어 각 장치에 할당 가능한 이름들의 집합임. Domain Namespace는 계층적으로 구성된 역트리 형태의 namespace임. domain namespace의 각 노드는 최대 63개의 문자로 구성된 string을 Label로 가짐. 이때 root는 빈 string(null)을 label로 가짐.

domain namespace의 subtree를 Domain 또는 Subdomain이라고 함. 아래와 같이 DNS에는 이런 각 domain에 대해서 mapping 정보를 가지고 있는 DNS Server가 존재함. 물론 각 domain 별로 여러 개의 분산 server를 가지는 것이 일반적임. root server 자체도 다양하게 존재함.



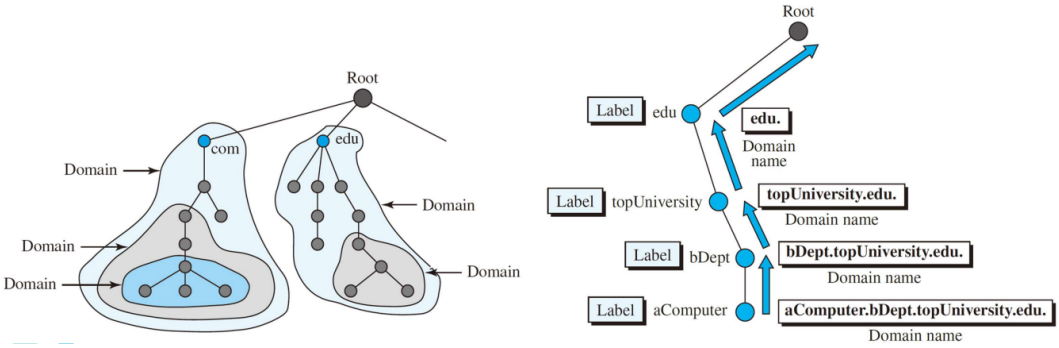
domain으로는 아래와 같은 것들이 존재할 수 있음.

1) Generic Domain : generic behavior에 따라 존재하는 domain. com(commercial), edu(education), org(nonproif orgainzation) 등이 있을 수 있음.

2) Country Domain : 나라 별로 존재하는 domain. fr, us, ko 등이 있을 수 있음.

domain namespace에서 domain name은 특정 노드에 대해서 구할 수 있는데, 이는 해당 노드부터 시작해 root까지의 label들을 모아 .으로 구분한 문자열임. 즉, domain name에서 뒤쪽에 나오는 label일수록 상위 domain임. domain namespace는 최대 128개(0 ~ 128)의 level을 가질 수 있고, 이에 따라 domain name은 최대 128개의 label로 구성될 수 있음.

root 바로 아래의 domain을 TLD(Top Level Domain)이라고 함.



3. Local DNS Server

Local DNS Server는 ISP가 관리하며 local에 존재하는 DNS server임. client는 local DNS server를 통해

DNS service를 이용함.

ip address 할당 시에는 ip address, 기본 게이트웨이, local DNS server의 ip address가 모두 제공된다고 함.

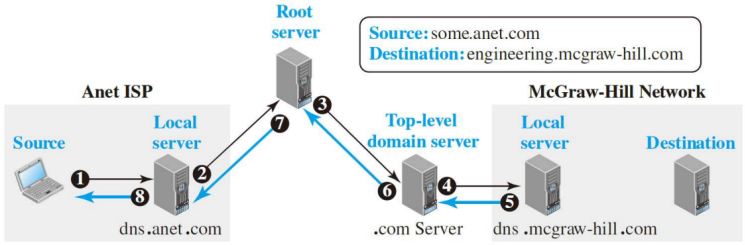
4.3.2. DNS Resolution

1. DNS Resolution

DNS Reoslution은 query로 전달받은 DNS name을 ip address로 변환하는 과정임. 이는 recurisve 또는 iterative하게 수행할 수 있음.

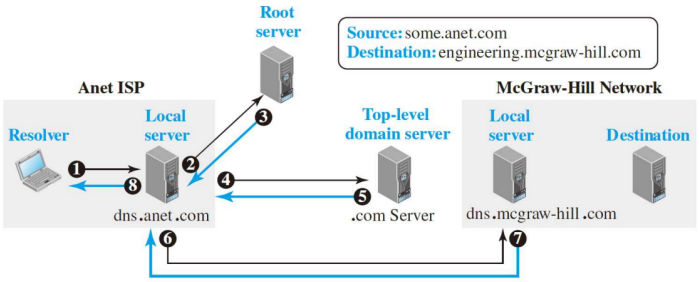
1) Recursive Resolution

Recursive Resolution은 client가 query를 local server로 전송하면, local server는 이를 root server로 전송하고 이후 server를 순서대로 순회하며 ip address를 찾는 방식임.



2) Iterative Resolution

Iterative Resolution은 client가 query를 local server로 전송하면, local server가 각 server와 각각 통신하며 ip address를 찾는 방식임.



2. Caching

물론 이런 resolution 과정이 매번 일어나는 것은 아니고, DNS server에서는 mapping 정보를 caching 함. 이후 특정 DNS 정보가 caching이 되어 있다면 그냥 그걸 씀. 이에 따라 대부분의 traffic은 local server에서 발생함.

client에서도 os 또는 browser level에서 caching이 가능함. 이는 resolvectl(linux)나 ipconfig(window) 등을 사용하여 확인할 수 있음.

3. Resource Record

각 DNS server의 DB의 각 record(행)는 특정 domain name에 대해 아래와 같은 5개의 정보를 가지는 tuple 구조를 가짐.

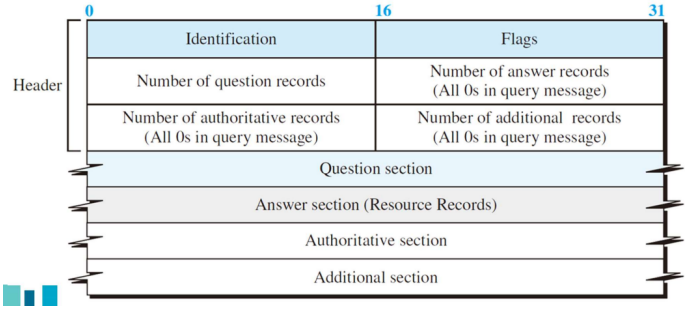
1) Domain Name
2) Value : domain name에 대한 정보.
3) TTL(Time To Live) : 해당 정보가 유효한 기간을 초 단위로 나타낸 것.
4) Class : network type.
5) Type : value가 어떻게 interprete되어야 하는지를 지정.

이때 type으로는 아래와 같은 것들이 있고, 주로 A(IPv4)와 AAAA(IPv6)를 사용함.

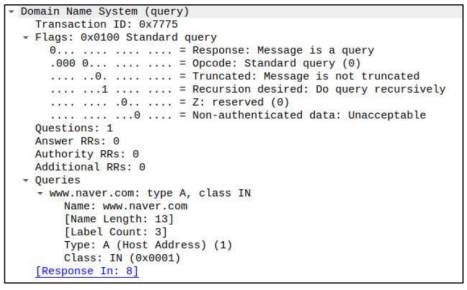
Type	*Interpretation of value*
A	A 32-bit IPv4 address
NS	Identifies the authoritative servers for a zone
CNAME	Defines an alias for the official name of a host
SOA	Marks the beginning of a zone
MX	Redirects mail to a mail server
AAAA	An IPv6 address

4. DNS Message
DNS Message의 format은 아래와 같음.

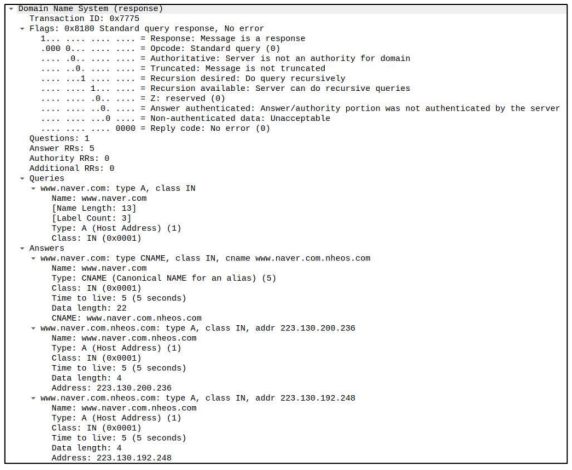
1) Identificaiton : query와 reply를 matching하기 위한 transaction id.
2) Flag : 해당 message가 query인지, reply인지 정의함.
3) Question Section : query 정보를 포함함.
4) Answer Section : 하나 또는 그 이상의 resource record를 포함함.
5) Authoritatve Section : domain name에 대한 추가적인 정보를 포함함.



이에 따라 주고받는 query(request)와 reply(response)의 예시는 아래와 같음.



Query



Reply

5. Wireless Network

현재 사용되는 대부분의 network는 wireless network 형태이므로, 이에 대해 알아보자.

5.1. Wireless Network

5.1.1. Wireless Network

1. Wireless Network
Wireless Network는 케이블 없이 전자기파를 활용해 데이터를 주고받는 network임. 즉, ethernet 대신

wifi, bluetooth, LTE/5G 등을 이용하는 network임.

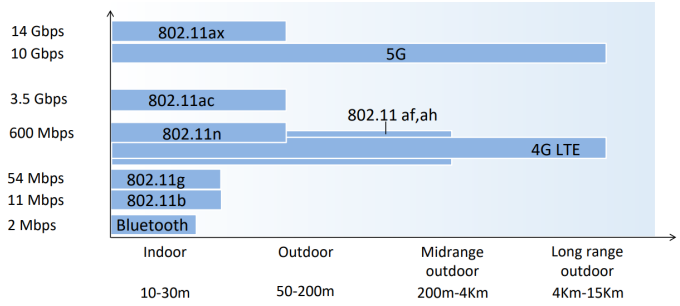
wireless network에서는 아래와 같은 주요 고려 사항이 존재함.
1) Wireless : 전자기파로 통신함. 중간에 장애물 등이 존재하면 통신이 어려움.
2) Mobility : 사용 중 이동이 존재함. 이동 시에 왜곡이 발생할 수 있음.

2. Wireless Network 구조
wireless network는 아래와 같은 구성 요소를 가짐.

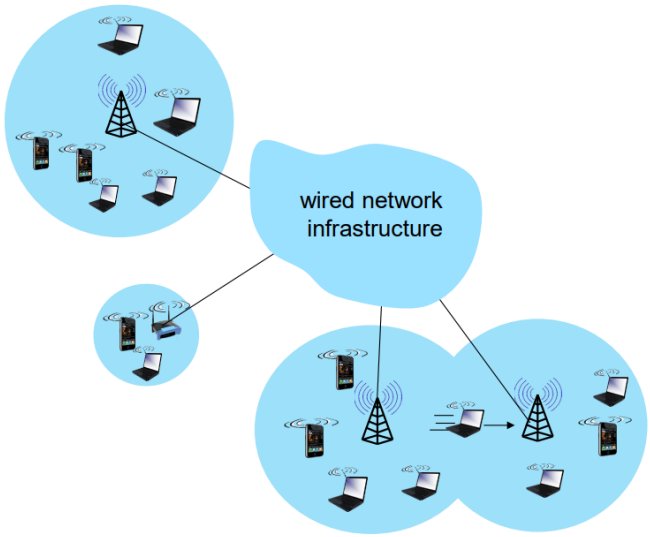
1) Wireless Host : wireless network에서의 host. 이는 stationary할 수도 있고, mobile일 수도 있음.

2) Base Station : infrastructure와 유선(etherent)으로 연결되어 있고, 해당 area의 wireless host와 packet을 주고받는 장치. 기지국 또는 AP가 존재함.

3) Wireless Link : wireless host와 base station 사이의 link. 이는 아래와 같이 사용되는 protocol에 의해 다양한 rate, distance, frequency를 가질 수 있음.



이런 구성요소에 따라 wireless network의 구조를 나타내면 아래와 같음. 각 wireless host는 base station과 wireless link로 연결되어 있음. 이때 base station은 infrastructure와 대체로 유선(etherent)으로 연결되어 있으므로, wireless network에서는 wirelss host와 base station 사이에 존재하는 통신에 대해서 다룸.

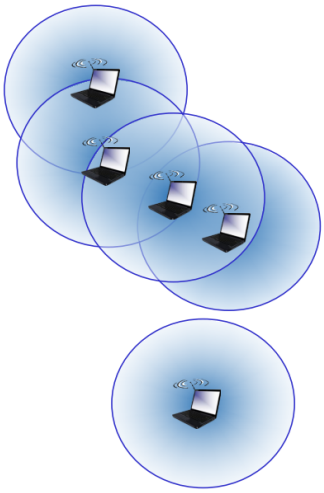


3. Wireless Network 분류
wireless network는 아래와 같이 두 가지로 분류할 수 있음. 대부분의 통신은 infrastructure mode라고 함.

1) Infrasturcture Mode
Infrasturcture Mode는 infrastructure가 존재하는 방식으로, host가 base station과 연결됨. 또한 각 host가 이동함에 따라 다른 base station과 연결됨.

2) Ad hoc Mode

Ad hoc Mode 또는 No Infrasturcture Mode는 infrastructure가 존재하지 않는 방식으로, host끼리 연결되어 통신함.



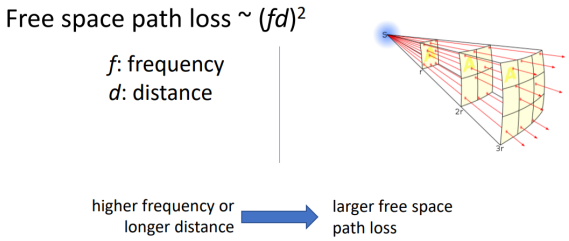
또한 각 mode는 single hop(직접 연결. wifi, cellular, bluetooth 등)이냐, multiple hop(여러 hop을 거쳐 연결. mesh net 등)이냐에 따라 더 분류될 수 있음.

5.1.2. Wireless Network의 특징

wireless network가 가지는 특징으로는 아래와 같은 것들이 있음.

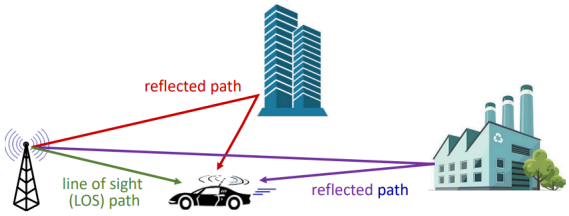
1. Fading

Fading 또는 Attenuation(감쇠)은 signal이 전달되면서 점점 작아지는 현상을 말함. wireless system에서는 전자기파로 정보를 주고받으므로 거리의 거듭제곱에 비례해 signal의 세기가 줄어듦. 아래와 같이 원할한 wireless 통신을 위해서는 frequency가 충분히 크고, 거리가 충분히 가까워야 함.



2. Multipath Propagation

Multipath Propagation은 동일한 여러 signal들이 장애물 등에 부딪힌 뒤 약간의 시간 차를 가지고 전달되는 것을 말함. 이 경우 두 신호가 중첩에 의해 bit flip 등이 발생하여 error rate가 증가할 수 있음.



3. Noise

Noise는 의도되지 않은 간섭이나 잡음임.

SNR(Signal-to-Noise Ratio)은 signal의 세기와 해당 signal에 포함된 noise의 세기의 비율임. 즉, 아래

와 같이 계산됨. signal 수신 지점에서 계산한 SNR이 높을수록 noise가 적고 signal이 명확하게 전달되는 것임.

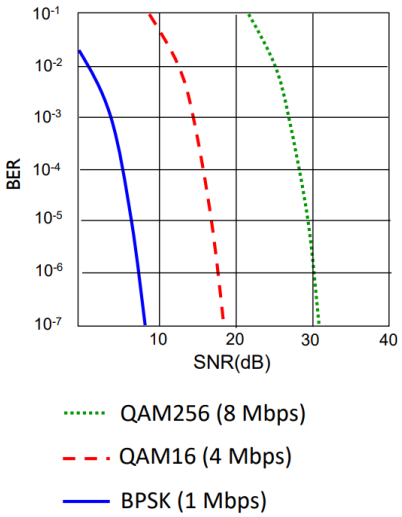
$$SNR = \frac{average\ signal\ power}{average\ noise\ power}$$

BER(Bit Error Rate)은 전송된 전체 bit에 대한 error가 발생한 bit의 개수임.

$$BER = \frac{error\ bits}{entire\ bits}$$

noise가 많을수록 error가 자주 발생하므로, 당연하게도 SNR과 BER은 tradoff 관계임. 아래의 그래프에서 이를 확인할 수 있음. 이때 QAM256, QAM16, BPSK는 각각 하나의 symbol로 8비트, 4비트, 1비트를 전송함. 한 번에 많은 비트를 전송할수록 각 signal을 정확히 구분하는 것이 어려우므로, noise가 적은 환경에서 사용하거나 더 많은 전력을 사용해서 SER을 줄여야 함(noise를 줄이거나 signal을 키워야 함.).

즉, noise의 존재를 고려해 적절한 bit rate를 가지는 기법을 선택해야 함.



5.1.3. CDMA

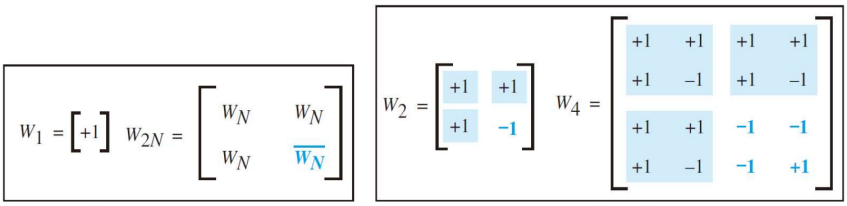
wireless network에서는 CDMA를 활용해 multiple access를 구현할 수 있음.

1. CDMA
CDMA(Code-Division Multiple Access)는 각 station이 chip이라는 code(비트열)를 사용하여 하나의 channel로 동시에 전송하는 기법임.

2. Chip
Chip 또는 Chipping Sequence는 각 station에 할당되는 서로 다른 code임. 이때 chip들은 아래의 조건을 만족함.

1) station의 개수가 N이면, chip의 길이도 N임.
2) chip은 순서쌍(벡터)으로 취급되며(스칼라 곱, 벡터 합 연산이 가능함.), 당연하게도 내적(inner product)이 가능함. 여기서 내적은 dot product임.
3) chip들은 서로 orthogonal(직교)함.

CDMA에서 chip의 sequence는 station의 개수만큼의 행/열을 가지는 Walsh table을 생성하여 얻음. 생성한 walsh table의 각 행 또는 열이 각 station의 chip으로 사용됨. walsh table은 아래와 같이 생성할 수 있음.



a. Two basic rules b. Generation of W_2 and W_4

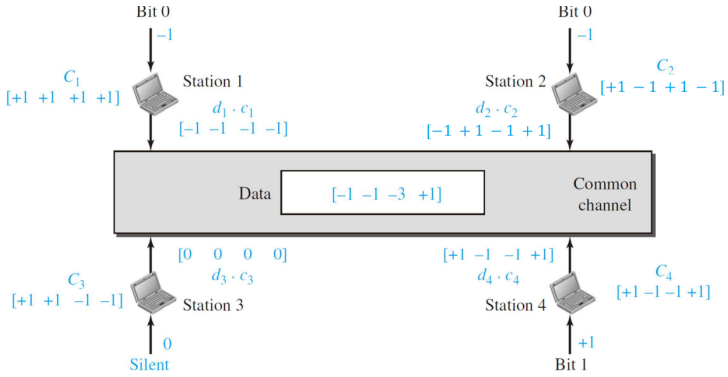
3. 동작 과정

각 station에서는 1을 전송한다면 +1을 값으로 하고, 0을 전송한다면 -1을 값으로 하고, 전송하지 않는다면 0을 값으로 하여 전송함.

각 station은 자신의 chip과 전송할 값을 곱해서(encoding) channel에 전송함. channel에는 각 station에서 전송한 chip×값이 모두 합쳐져 있게 됨.

각 station에서 특정 station에서 전송한 값을 확인하려면, channel에 존재하는 전체 값의 합과 송신자의 chip에 대해서 내적을 계산하고 station의 개수로 나누면 됨(decoding).

예를 들어, 4개의 station이 존재하고 각 chip을 $C_1 \sim C_4$, 전송하려는 값을 $d_1 \sim d_4$라고 하자. 각 station이 전송한 chip×값의 합이 $d_1C_1 + d_2C_2 + d_3C_3 + d_4C_4$이고, 송신자의 chip 값이 C_1이면, 내적의 결과는 $d_1 < C_1, C_1 >$임. 이때 walsh table에는 1 또는 -1만 존재하므로, $d_1 < C_1, C_1 >= 4d_1$임. 즉, station의 개수로 나눈 d_1을 확인할 수 있음.



이와 같은 encoding/decoding 과정이 각 time slot마다 수행됨.

5.1.4. IEEE 802.11 : Wifi

1. IEEE 802.11 : wifi

IEEE 802.11 또는 wifi는 wireless 통신 표준임.

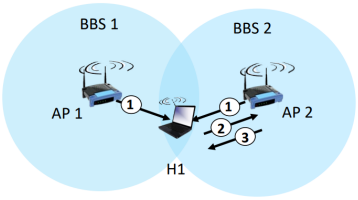
wifi는 2.4GHz, 5GHZ 등의 frequency 대역을 사용함. 참고로 wifi 이름에 붙는 2.4G, 5G는 이동통신 generation이 아니라, 해당 기기가 사용하는 대역을 나타냄. 물론 5G가 사용자가 적어 더 쾌적할 수는 있지만, 사용하는 frequency 대역이 커지면 loss 또한 커질 수 있음. 거리가 멀면 5G 대역, 가까우면 2.4G 대역이 이 유리할 수 있다고 함.

2. Passive vs. Active Scanning

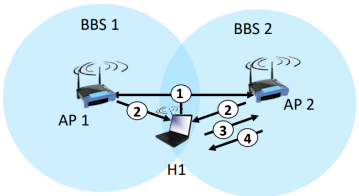
wifi에서는 AP와 host 간의 연결에 passive scanning 또는 active scanning을 활용함.

1) Passive Scanning : host가 AP로부터의 beacon frame을 받고, AP를 선택해 request를 보내면 해당 AP가 response를 전송해 연결하는 방식.

2) Active Scanning : host가 probe request frame을 broadcast하면 AP가 probe response frame을 해당 host에 보내고, 이후 host가 AP를 선택해 request를 보내면 해당 AP가 response를 전송해 연결하는 방식.



passive scanning:
(1) beacon frames sent from APs
(2) association Request frame sent: H1 to selected AP
(3) association Response frame sent from selected AP to H1



active scanning:
(1) Probe Request frame broadcast from H1
(2) Probe Response frames sent from APs
(3) Association Request frame sent: H1 to selected AP
(4) Association Response frame sent from selected AP to H1

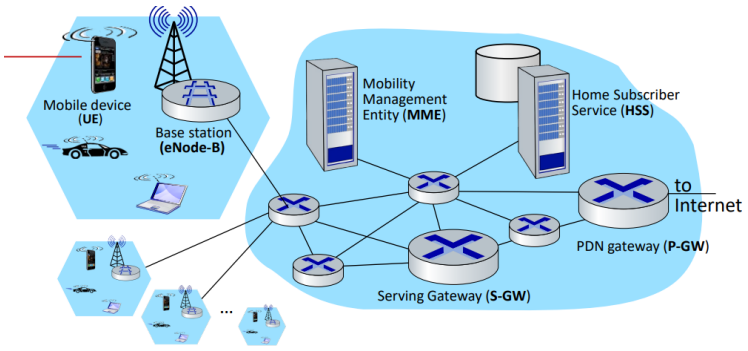
3. Rate Adaptaion
앞에서 다룬 것처럼, noise의 정도에 따라 QAM256, QAM16, BPSK 등 서로 다른 전송 rate를 가지는 기법을 적절히 선정해야 함. wifi에서는 적응형으로 각 기법을 상황에 맞에 적용함.

4. Power Management
AP에서 어떤 host에게 보낼 데이터가 없으면 해당 host는 sleep mode가 되어 power 관점에서의 efficiency를 확보함.

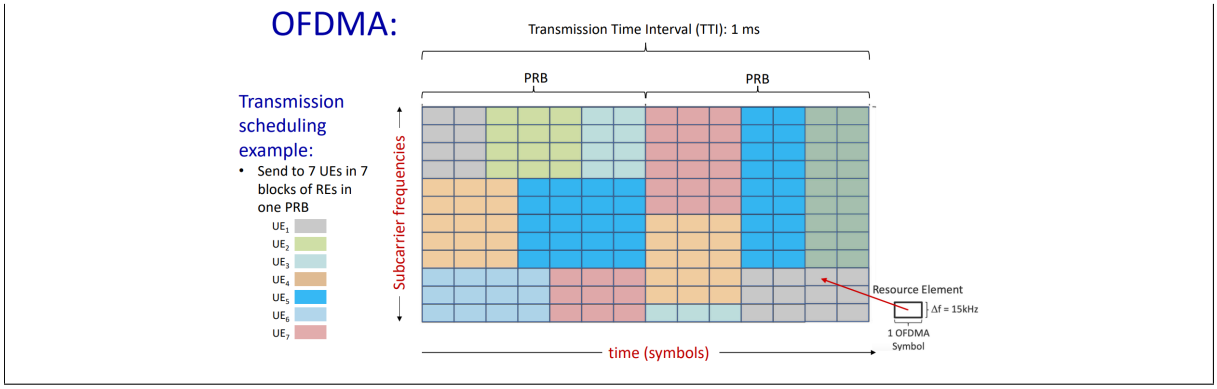
5.1.5. 4G LTE

1. 4G LTE 구조
4G LTE의 구조는 아래와 같음. 오른쪽 파란색 묶음은 EPC(Evolved Packet Core)로, LTE의 core network임. 이는 이건 이동통신사에서 관리하는 부분임.

1) UE : mobile device.
2) eNode-B : base station.
3) MME : mobile device에 대한 관리를 수행하는 부분.
4) HSS : mobile device들의 정보를 저장하는 부분.
5) S-GW : 데이터 경로 노드.
6) P-GW : EPC와 외부 network를 연결하는 노드. public internet으로 나갈 때는 항상 이를 거치게 됨.



2. OFDMA
OFDMA(Orthogonal Frequency Division Multiple Access)는 아래 그림과 같이 시간에 따라 frequency를 나눈 뒤 각 이용자에게 할당하여, 각자가 특정 시점에 특정 주파수만 사용할 수 있도록 하는 기법임. LTE에서는 OFDMA를 활용함.



참고로, Gateway는 두 개 이상의 서로 다른 network 간의 데이터 전달 및 변환을 중계하는 router 등의 장치임. Default Gateway(기본 게이트웨이)는 destination이 내부 network에 없을 때, 외부 네트워크로 나가는 관문 역할의 router 등의 장치임.

해외로의 통신은, 국내 이동통신사와 해외 이동통신사가 데이터를 주고받는 것으로 구현됨.