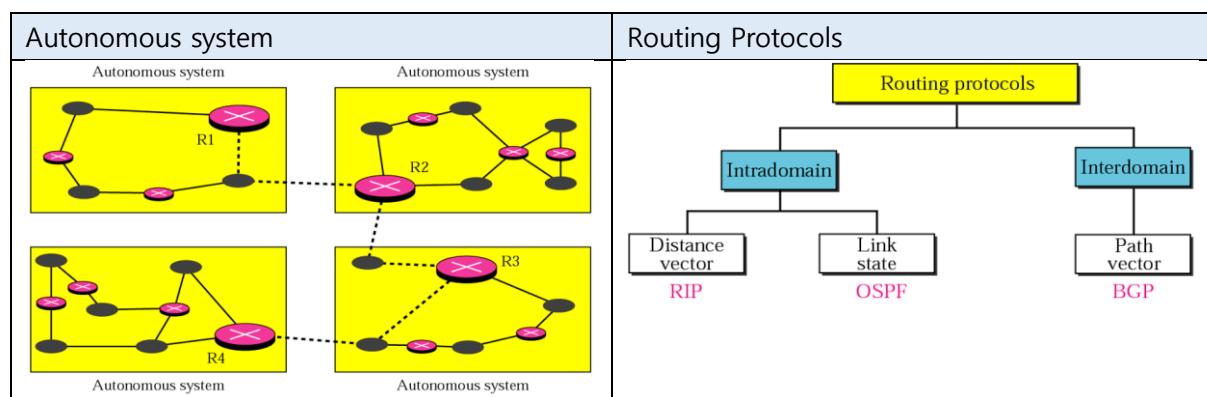


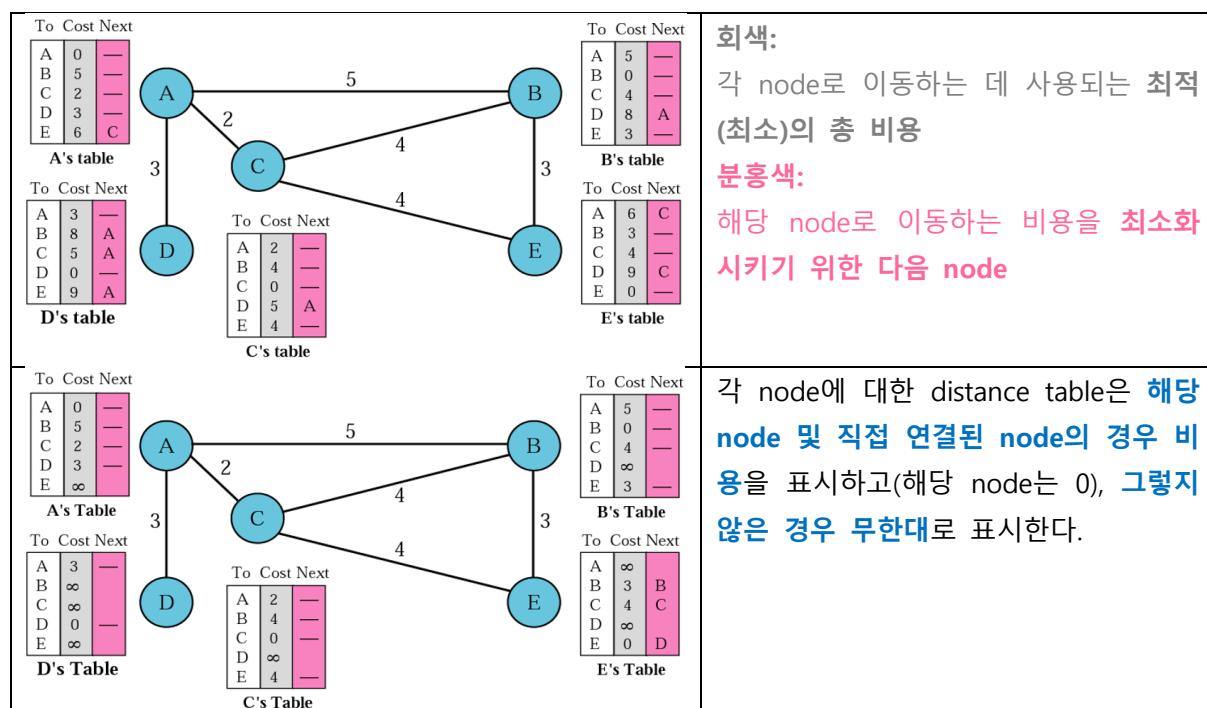
01-01: INTRA- AND INTERDOMAIN ROUTING

Intradomain routing	어떤 autonomous system 내부에서의 라우팅
Interdomain routing	Autonomous system 간의 라우팅



01-02: DISTANCE VECTOR ROUTING

<Distance Vector Routing Table>



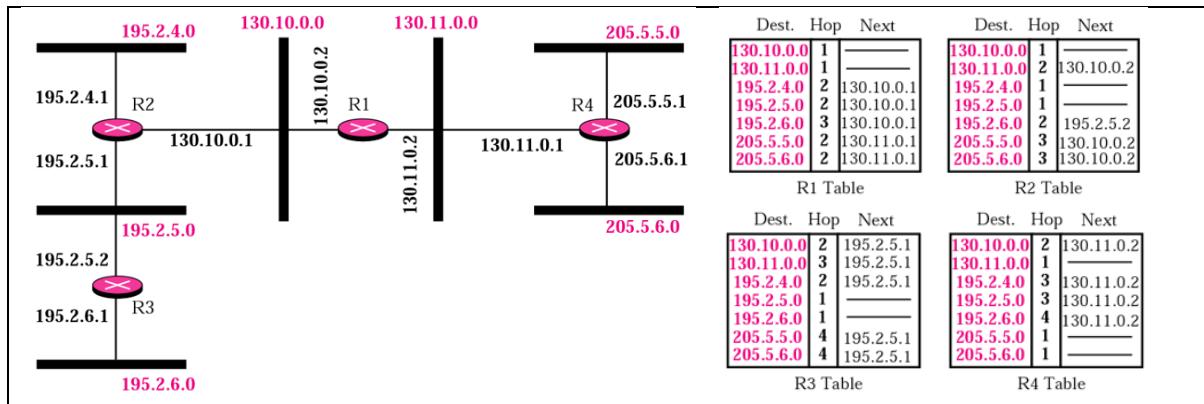
→ 각 node는 직접 연결된 다른 node와 라우팅 테이블을 공유하여 주기적으로/변동이 있을 때 업데이트한다.

<Distance Vector Routing에서의 업데이트>

	<p>A's Modified Table: C의 Routing Table의 cost 값에 A's Old Table의 A->C의 cost 값을 더한다. 모두 C로 한다.</p> <p>A's New Table: 각 node로의 cost는 min(기존 cost, A's Modified Table의 해당 cost)의 값으로 한다. 업데이트되는 부분을 Modified Table의 해당 node로 한다.</p>
	<p><Two-node instability></p> <p>After failure: A의 라우팅 테이블의 node X에 대한 항목이 무한대로 업데이트된다.</p> <p>After A receives update from B: A의 라우팅 테이블의 X에 대한 항목은 $(\text{old } B \rightarrow X \text{ cost}) + (A \rightarrow B \text{ cost}) = 6+4 = 10$으로 업데이트된다.</p> <p>After B Receives update from A: B의 라우팅 테이블의 X에 대한 항목은 $(\text{old } A \rightarrow X \text{ cost}) + (B \rightarrow A \text{ cost}) = 10+4 = 14$로 업데이트된다.</p> <p>Finally: 이것이 반복되어 결국 무한대가 된다.</p>
	<p><Three-node instability></p> <p>After A sends ... : A가 B에 route를 전송하였으므로 A, B의 라우팅 테이블의 node X에 대한 항목은 모두 무한대가 된다. (C는 전송 실패)</p> <p>After C sends the route to B: B의 라우팅 테이블의 node X에 대한 항목은 $(\text{old } C \rightarrow X \text{ cost}) + (B \rightarrow C \text{ cost}) = 5+3 = 8$로 업데이트된다.</p> <p>After B sends the route to A: A의 라우팅 테이블의 node X에 대한 항목은 $(\text{old } B \rightarrow X \text{ cost}) + (A \rightarrow B \text{ cost}) = 8+4 = 12$로 업데이트된다.</p>

01-03. RIP (Routing Information Protocol)

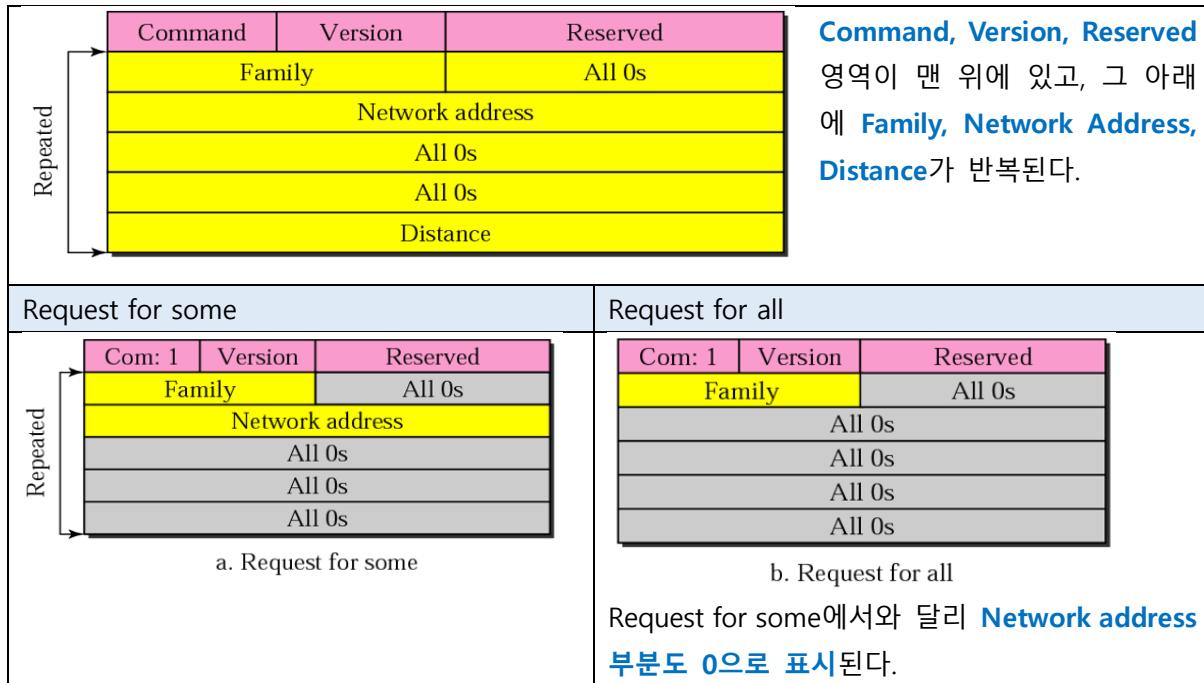
Routing Information Protocol (RIP): autonomous system에서 사용되는 **intradomain 라우팅** 프로토콜로, **distance vector routing**에 기반한 매우 간단한 프로토콜이다.



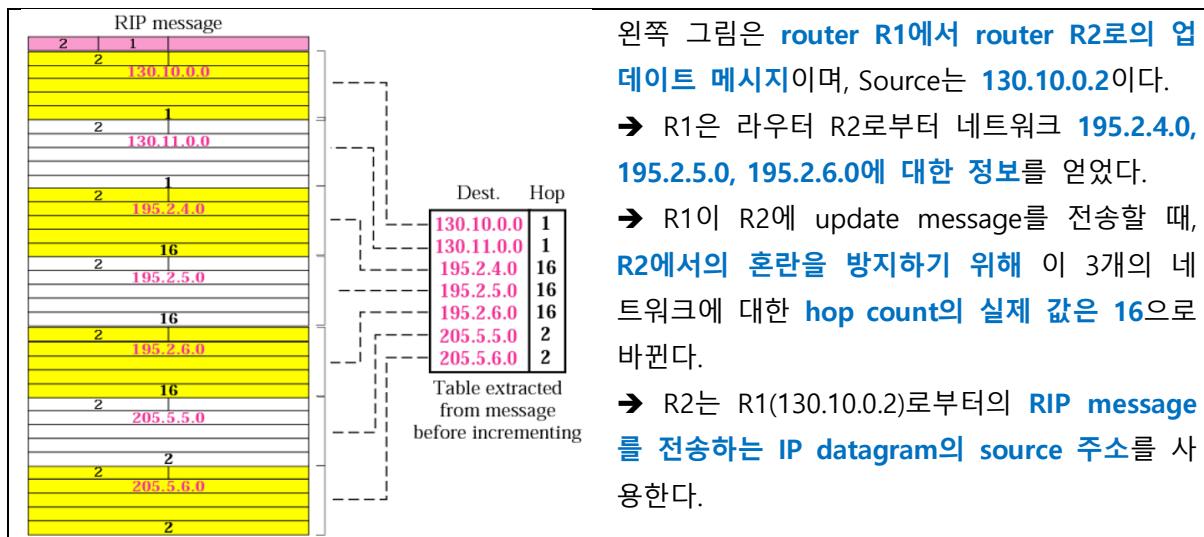
각 Router의 Table에서...

Dest.	네트워크로 연결된 각각의 다른 node의 IP 주소
Hop	Dest. Node로 최적의 경로로 이동하기 위해서 거쳐야 하는 Hop의 수
Next	Dest. Node로 최적의 경로로 이동하기 위한 다음 지점의 IP 주소

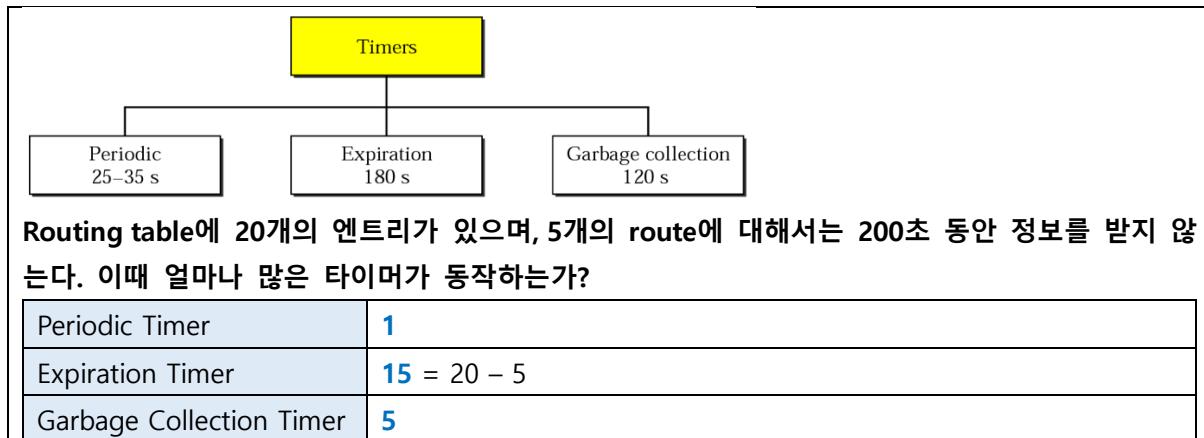
<RIP 메시지 형식>



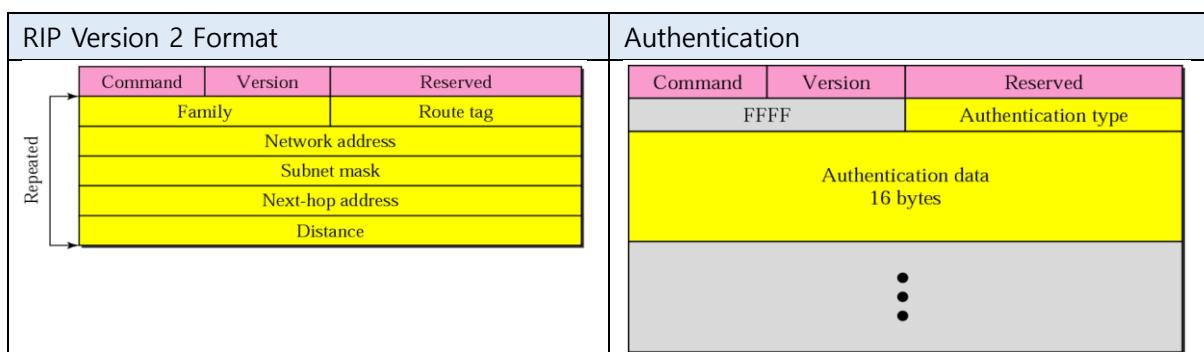
<RIP 예제 1>



<RIP 예제 2>



<RIP Version 2 메시지 형식>

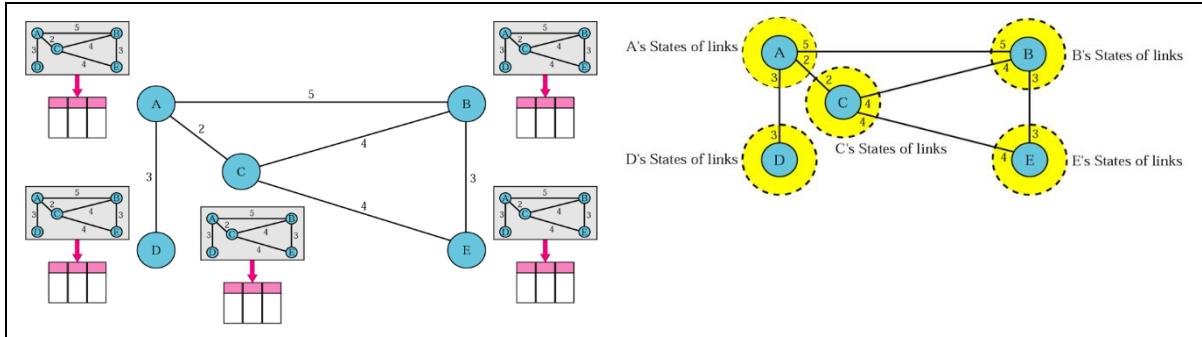


→ RIP은 well-known port 520에서 UDP 서비스를 이용한다.

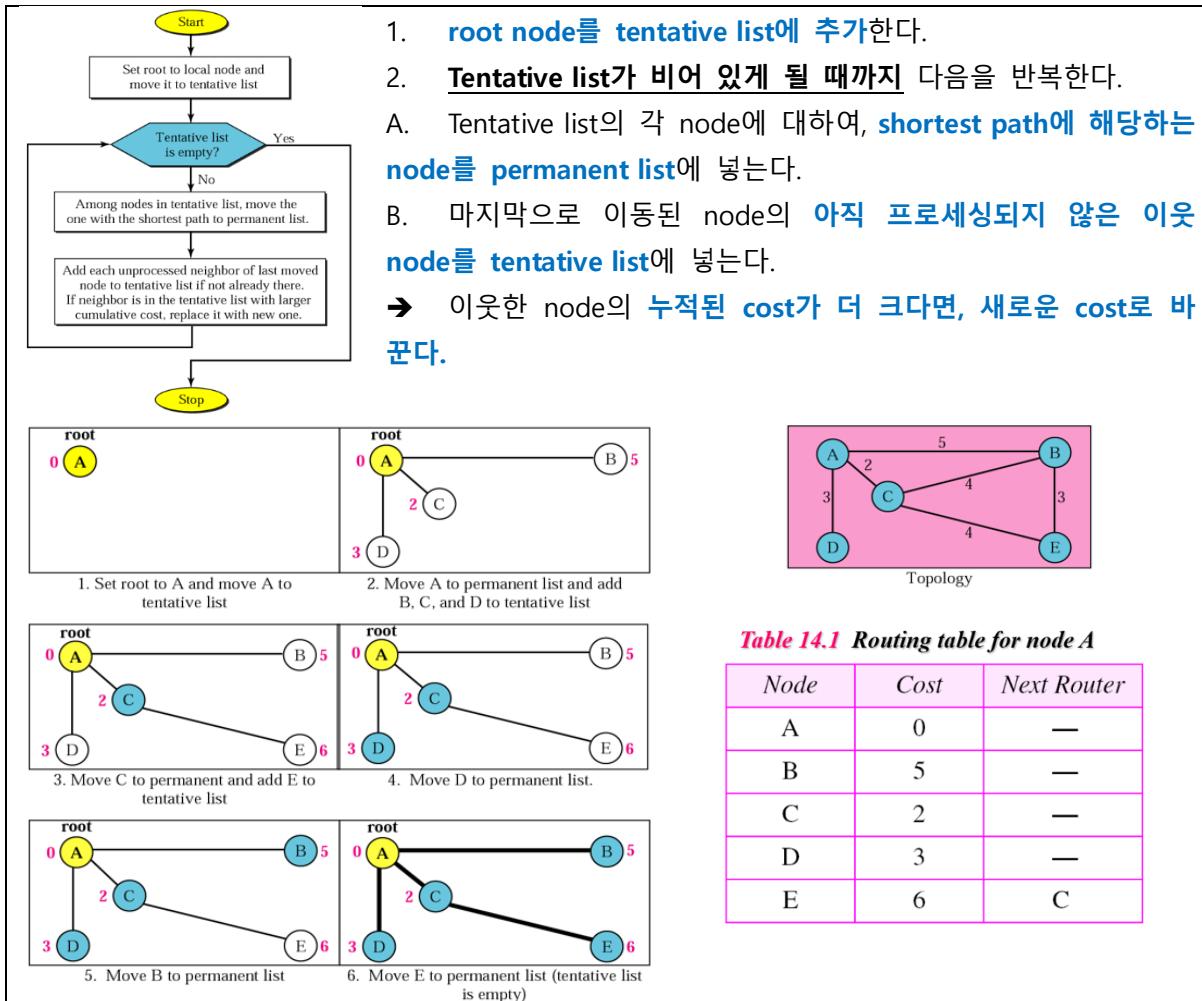
01-04. Link State Routing

Link State Routing: 도메인에 있는 각 node가 도메인의 전체 topology(연결 방식)를 알고 있다면, node는 Dijkstra's Algorithm을 사용하여 Routing Table을 생성할 수 있다.

<Link State Routing의 개념도>

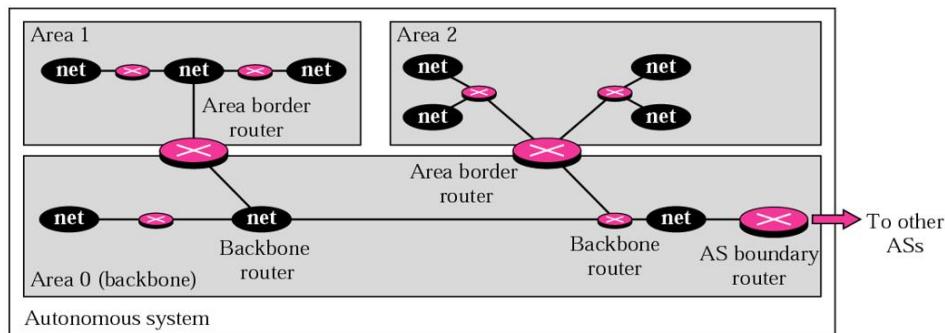


<Dijkstra Algorithm>



01-05. OSPF (Open Shortest Path First)

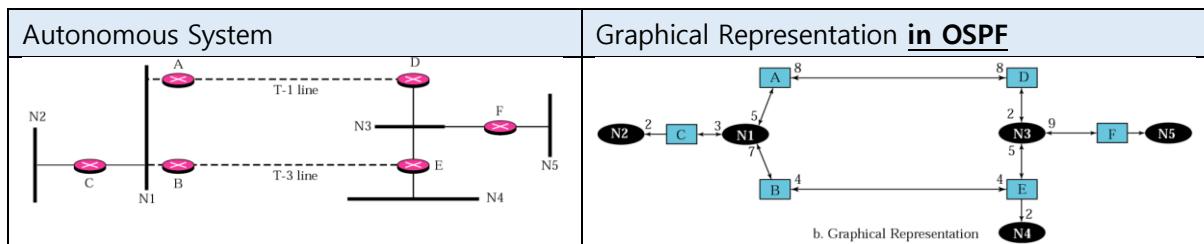
<Areas in an autonomous system>



<Types of links>

		Link Identification	Link Data
Point-to-Point link	 Router → Point-to-point network → Router	이웃 router의 주소	Interface number
Transient link	 a. Transient network b. Unrealistic representation c. Realistic representation	Designated router의 주소	라우터의 주소
Stub link	 a. Stub network b. Representation	네트워크 주소	Network mask
Virtual link		이웃 router의 주소	라우터의 주소

<Autonomous System and its graphical representation in OSPF>



<OSPF 패킷의 종류>

Hello	
Database description	
Link state request	
Link state update	<p>Router link, Network link, Summary link to network, Summary link to AS boundary router, External link</p> <p><Link state update packet></p> <pre> graph TD subgraph LSA_Packet [Link state update packet] direction TB H[OSPF common header 24 bytes Type: 4] --- NLA[Number of link state advertisements] NLA --- LSA[Link state advertisement Any combination of five different kinds (network link, router link, summary link to network, summary to boundary router, or external link)] LSA -.-> LSA end </pre>
Link state acknowledgment	

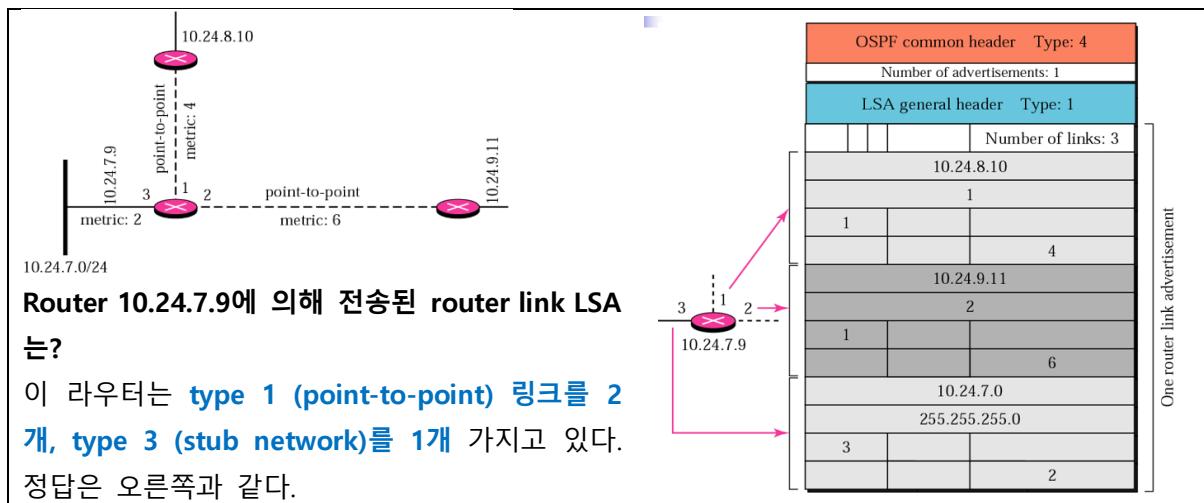
<OSPF common header and LSA (Link State Advertisement) general header>

OSPF common header	LSA general header																																																	
<table border="1"> <tr> <td>0</td> <td>7 8</td> <td>15 16</td> <td>31</td> </tr> <tr> <td>Version</td> <td>Type</td> <td>Message length</td> <td></td> </tr> <tr> <td colspan="4">Source router IP address</td> </tr> <tr> <td colspan="4">Area Identification</td> </tr> <tr> <td>Checksum</td> <td colspan="3">Authentication type</td> </tr> <tr> <td colspan="4">Authentication (32 bits)</td> </tr> </table>	0	7 8	15 16	31	Version	Type	Message length		Source router IP address				Area Identification				Checksum	Authentication type			Authentication (32 bits)				<table border="1"> <tr> <td>Link state age</td> <td>Reserved</td> <td>E</td> <td>T</td> <td>Link state type</td> </tr> <tr> <td colspan="4">Link state ID</td> <td></td> </tr> <tr> <td colspan="5">Advertising router</td> </tr> <tr> <td colspan="5">Link state sequence number</td> </tr> <tr> <td>Link state checksum</td> <td colspan="4">Length</td> </tr> </table>	Link state age	Reserved	E	T	Link state type	Link state ID					Advertising router					Link state sequence number					Link state checksum	Length			
0	7 8	15 16	31																																															
Version	Type	Message length																																																
Source router IP address																																																		
Area Identification																																																		
Checksum	Authentication type																																																	
Authentication (32 bits)																																																		
Link state age	Reserved	E	T	Link state type																																														
Link state ID																																																		
Advertising router																																																		
Link state sequence number																																																		
Link state checksum	Length																																																	

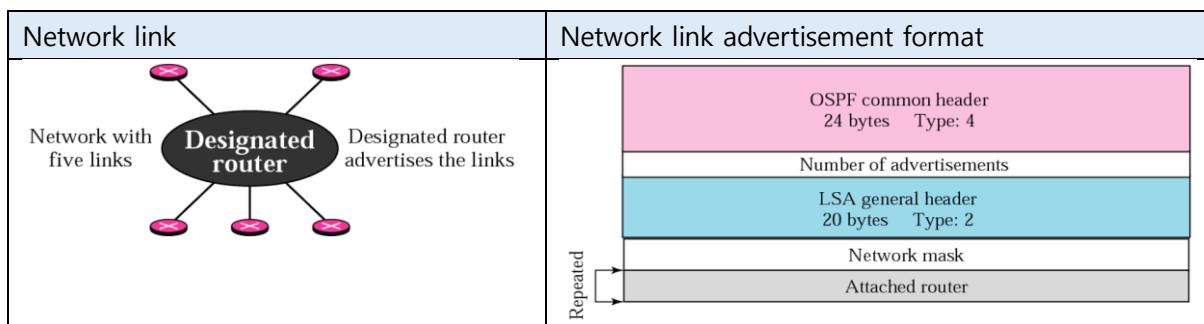
<Router link and Router link LSA>

Router link	Router link LSA																				
<p>Advertising router with four links</p> <p>To transient network</p> <p>Point-to-point</p> <p>To stub network</p> <p>Virtual</p>	<p>OSPF common header 24 bytes Type: 4</p> <p>Number of advertisements</p> <p>LSA general header 20 bytes Type: 1</p> <table border="1"> <tr> <td>Reserved</td> <td>E</td> <td>B</td> <td>Reserved</td> <td>Number of router links</td> </tr> <tr> <td colspan="3">Link ID</td> <td colspan="2">Link data</td> </tr> <tr> <td>Link type</td> <td># of TOS</td> <td colspan="3">Metric for TOS 0</td> </tr> <tr> <td>TOS</td> <td>Reserved</td> <td colspan="3">Metric</td> </tr> </table> <p>Repeated</p> <p>Repeated</p>	Reserved	E	B	Reserved	Number of router links	Link ID			Link data		Link type	# of TOS	Metric for TOS 0			TOS	Reserved	Metric		
Reserved	E	B	Reserved	Number of router links																	
Link ID			Link data																		
Link type	# of TOS	Metric for TOS 0																			
TOS	Reserved	Metric																			

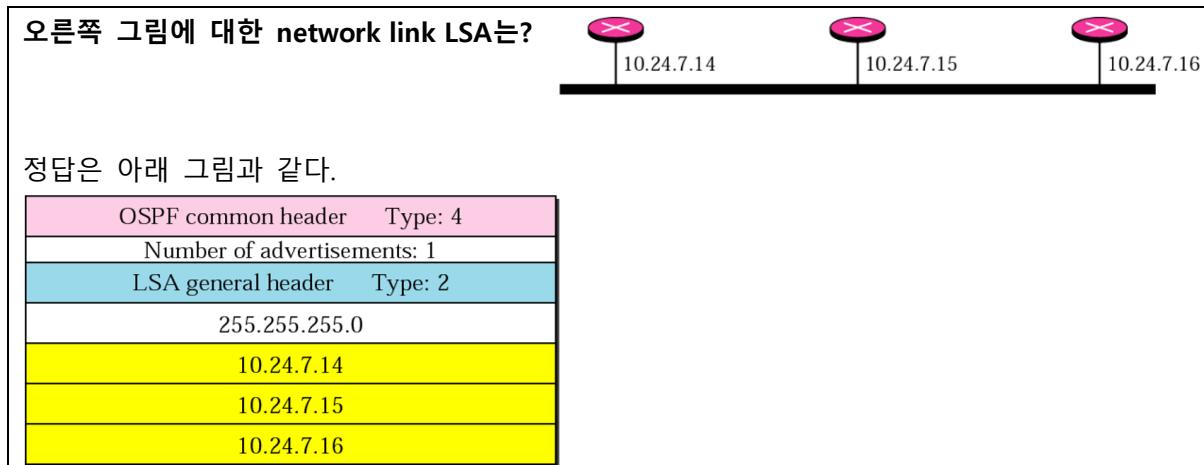
<Example 3>



<Network link and its advertisement format>



<Example 4>



<Example 5 and Example 6>

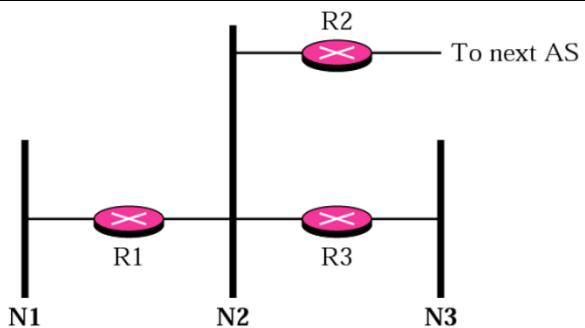
<Example 5>

오른쪽 그림에서 어떤 라우터가 router link LSA를 보내는가?

정답:

모든 라우터가 router link LSA를 보낸다.

- R1은 N1, N2의 2개의 link
- R2는 N1의 1개의 link
- R3은 N2, N3의 2개의 link



<Example 6>

위 그림에서 어떤 라우터가 network link LSA를 보내는가?

정답:

모든 네트워크가 network link를 광고해야 한다.

- N1에 대한 광고는 R1에 의해 이루어지며, 연결된 라우터가 R1뿐이므로 R1은 자동으로 designated router가 된다.
- N2에 대한 광고는 R1, R2 또는 R3에 의해 이루어진다. (이들 중 designated router로 선택된 라우터)
- N3에 대한 광고는 R3에 의해 이루어지며, 연결된 라우터가 R3뿐이므로 R3은 자동으로 designated router가 된다.

<OSPF Packet Museum>

Summary link to network	Summary link to network LSA						
	<table border="1"> <tr><td>OSPF common header 24 bytes Type: 4</td></tr> <tr><td>Number of advertisements</td></tr> <tr><td>LSA general header 20 bytes Type: 3</td></tr> <tr><td>Network mask</td></tr> <tr><td>TOS</td></tr> <tr><td>Metric</td></tr> </table>	OSPF common header 24 bytes Type: 4	Number of advertisements	LSA general header 20 bytes Type: 3	Network mask	TOS	Metric
OSPF common header 24 bytes Type: 4							
Number of advertisements							
LSA general header 20 bytes Type: 3							
Network mask							
TOS							
Metric							
Summary link to AS boundary router	Summary link to AS boundary router LSA						
	<table border="1"> <tr><td>OSPF common header 24 bytes Type: 4</td></tr> <tr><td>Number of advertisements</td></tr> <tr><td>LSA general header 20 bytes Type: 4</td></tr> <tr><td>All 0s</td></tr> <tr><td>TOS</td></tr> <tr><td>Metric</td></tr> </table>	OSPF common header 24 bytes Type: 4	Number of advertisements	LSA general header 20 bytes Type: 4	All 0s	TOS	Metric
OSPF common header 24 bytes Type: 4							
Number of advertisements							
LSA general header 20 bytes Type: 4							
All 0s							
TOS							
Metric							

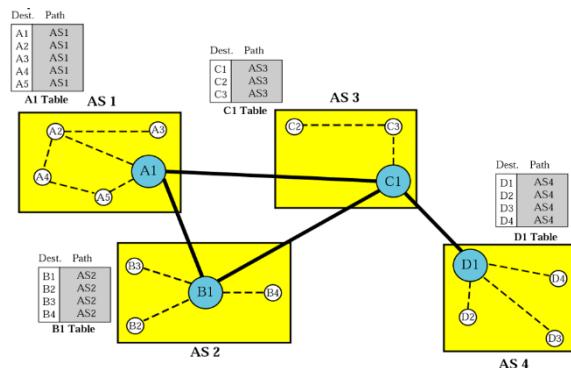
<OSPF Packet Museum>

External link 	External link LSA
Hello packet 	Database description packet
Link state request packet 	Link state acknowledgment packet

→ OSPF 패킷은 IP datagram으로 캡슐화된다.

01-06. Path Vector Routing

Path Vector Routing: distance vector routing과 비슷하지만, 각 AS에 speaker node가 있으며, speaker node는 routing table을 생성하고 그것을 이웃한 AS의 speaker node에 전송한다.



<각 autonomous system의 stabilized table>

Dest. Path	Dest. Path	Dest. Path	Dest. Path
A1 AS1 ... A5 AS1	A1 AS2-AS1 ... A5 AS2-AS1	A1 AS3-AS1 ... A5 AS3-AS1	A1 AS4-AS3-AS1 ... A5 AS4-AS3-AS1
B1 AS1-AS2 ... B4 AS1-AS2	B1 AS2 ... B4 AS2	B1 AS3-AS2 ... B4 AS3-AS2	B1 AS4-AS3-AS2 ... B4 AS4-AS3-AS2
C1 AS1-AS3 ... C3 AS1-AS3	C1 AS2-AS3 ... C3 AS2-AS3	C1 AS3 ... C3 AS3	C1 AS4-AS3 ... C3 AS4-AS3
D1 AS1-AS2-AS4 ... D4 AS1-AS2-AS4	D1 AS2-AS3-AS4 ... D4 AS2-AS3-AS4	D1 AS3-AS4 ... D4 AS3-AS4	D1 AS4 ... D4 AS4

A1 Table

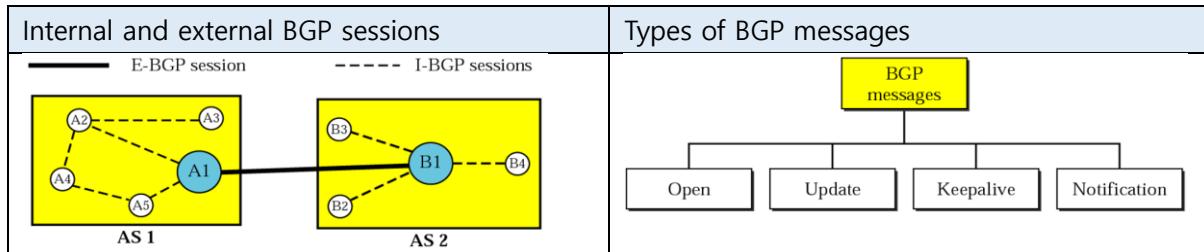
B1 Table

C1 Table

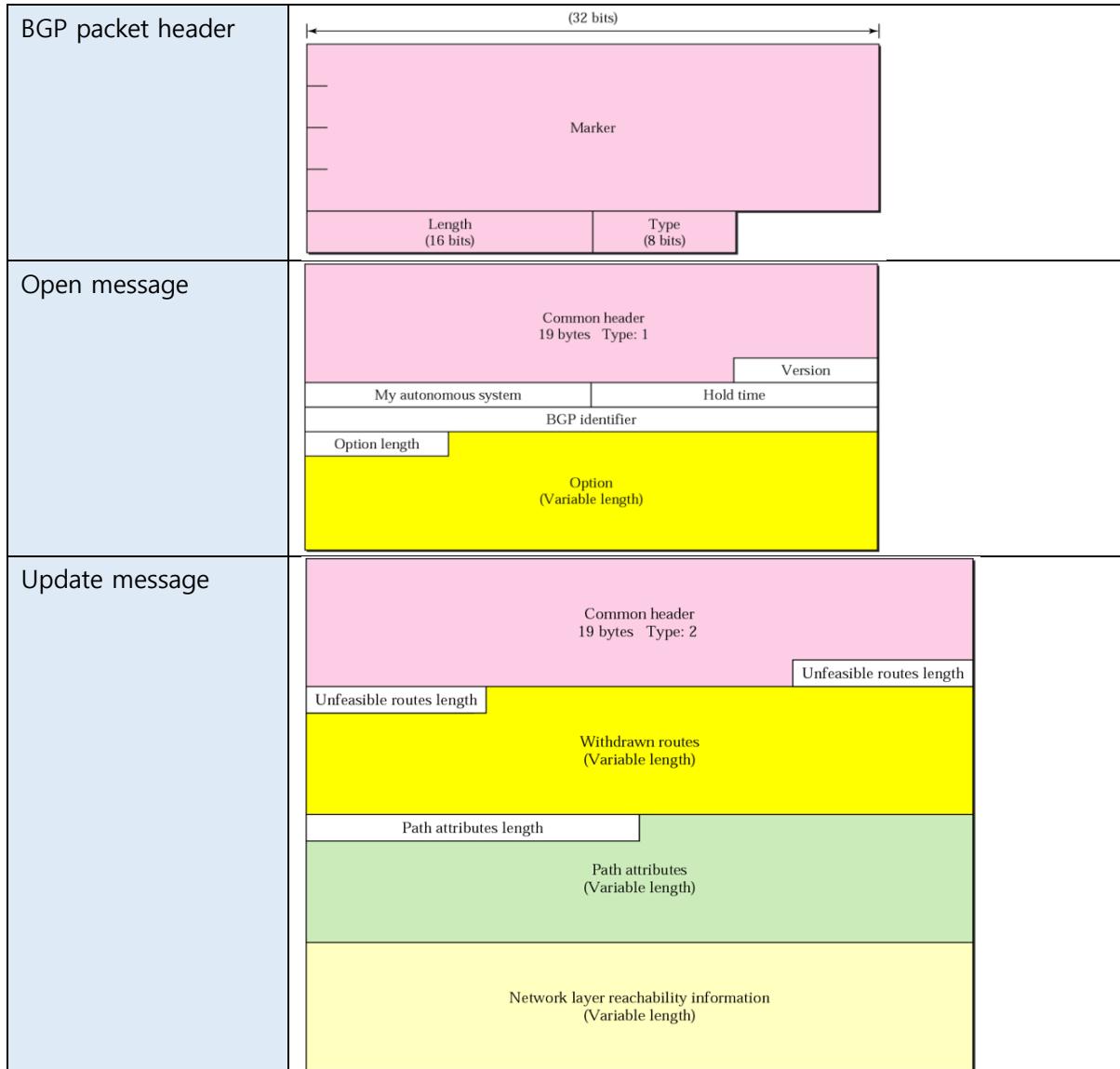
D1 Table

01-07. Border Gateway Protocol (BGP)

Border Gateway Protocol (BGP): path vector routing을 이용한 **interdomain routing** protocol

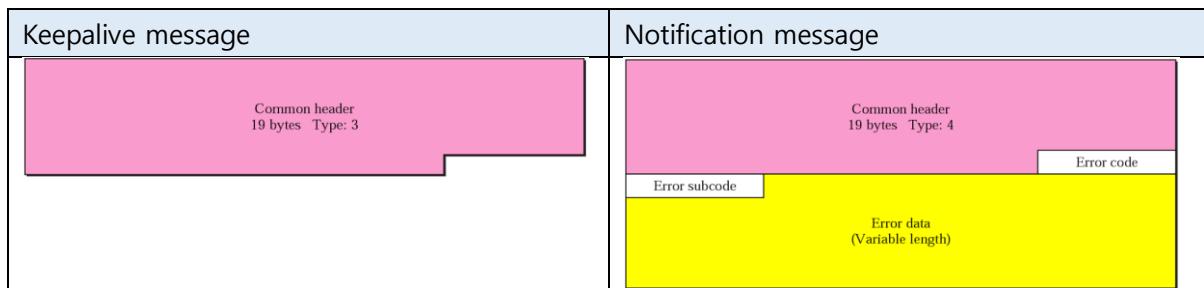


<BGP packet>



→ BGP는 **classless addressing**과 **CIDR (classless inter-domain routing)**을 지원한다.

<BGP packet>

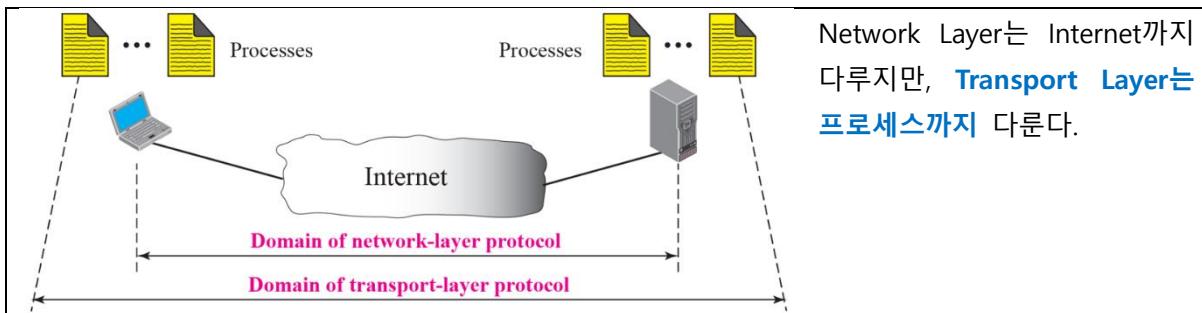


<BGP의 오류 코드>

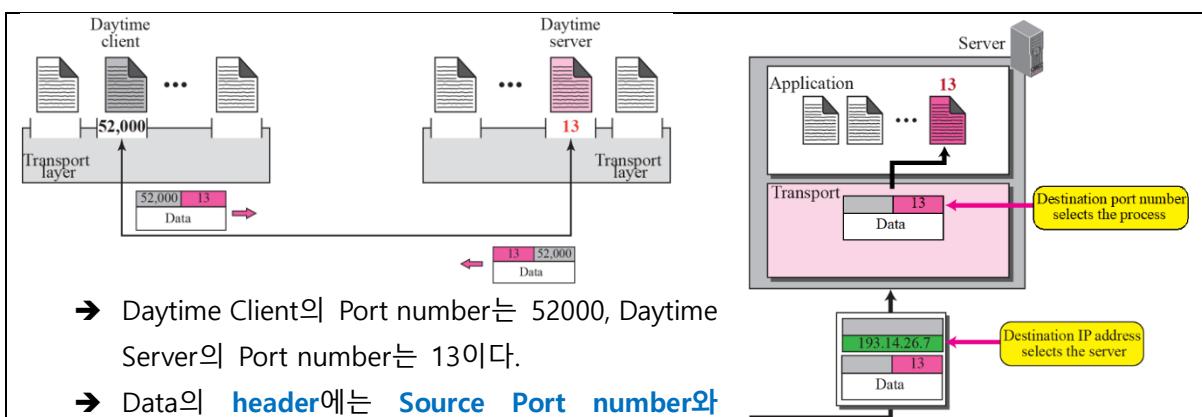
코드	Error Code 설명	Error Subcode 설명
1	Message header error	3개의 서로 다른 subcode ➔ Synchronization problem (1) ➔ Bad message length (2) ➔ Bad message type (3)
2	Open message error	6개의 서로 다른 subcode ➔ Unsupported version number (1) ➔ Bad peer AS (2) ➔ Bad BGP identifier (3) ➔ Unsupported optional parameter (4) ➔ Authentication failure (5) ➔ Unacceptable hold time (6)
3	Update message error	11개의 서로 다른 subcode ➔ Malformed attribute list (1) ➔ [Unrecognized (2) / Missing (3)] well-known attribute ➔ Attribute [flag (4) / length (5)] error ➔ Invalid [origin (6) / next hop (8)] attribute ➔ AS routing loop (7) ➔ Optional attribute error (9) ➔ Invalid network field (10) ➔ Malformed AS_PATH (11)
4	Hold timer expired	Subcode 없음
5	Finite state machine error	Procedural error를 정의하므로 Subcode 없음
6	Cease	Subcode 없음

➔ BGP는 port 179에서 TCP 서비스를 이용한다.

02-01. Network Layer vs. Transport Layer



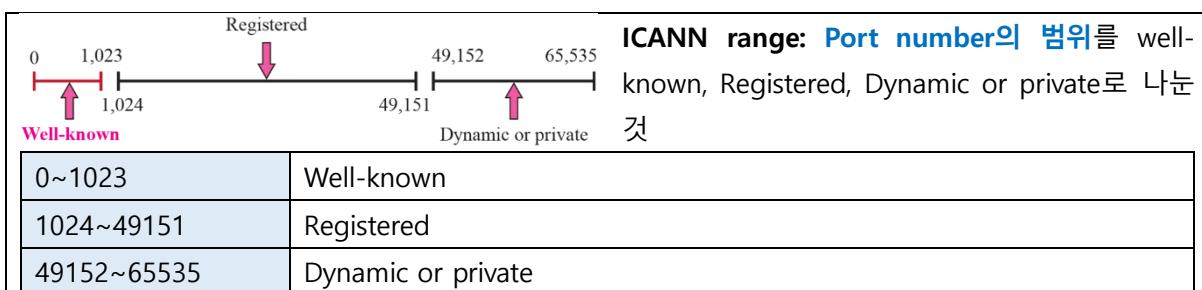
02-02. Port Number



<Port number vs. IP address>

Destination Port number	프로세스를 선택한다.
Destination IP address	서버를 선택한다.

<ICANN ranges>



<Example 1>

UNIX에서는 well-known port가 /etc/services 파일에 저장되어 있다.

→ 이 파일의 각 line은 server의 이름과 well-known port 번호를 저장하고 있다.

→ 다음 그림은 TFTP에 대한 port를 보여 준다.

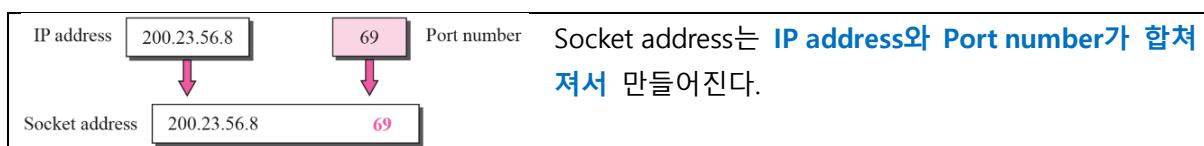
■ **TFTP는 UDP 또는 TCP에서 port 69를 사용할 수 있다.**

→ **SNMP는 161, 162의 서로 다른 목적의 2개의 port 번호를 가지고 있다.**

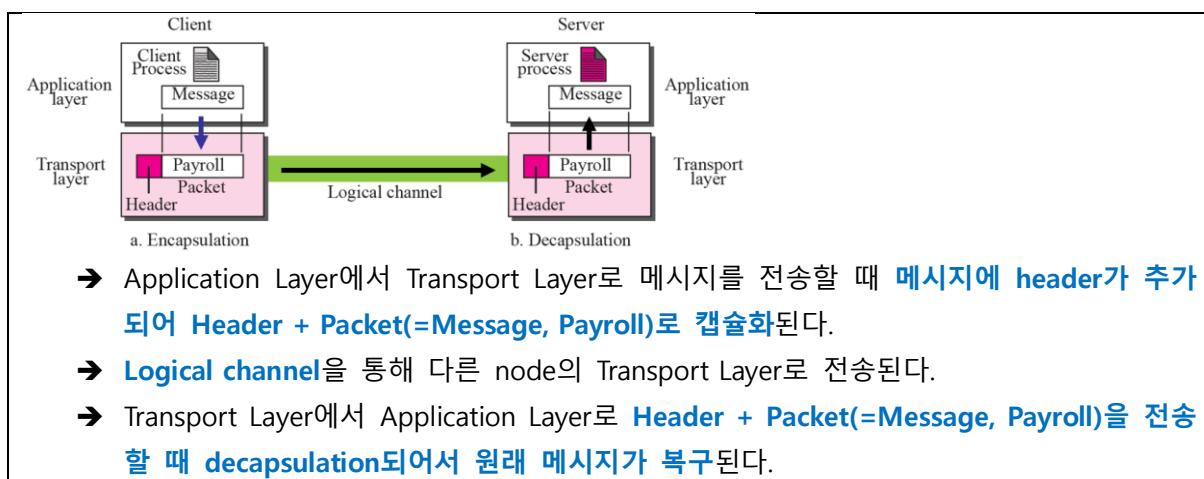
```
$grep tftp /etc/services
tftp          69/tcp
tftp          69/udp
```

```
$grep snmp /etc/services
snmp161/tcp#Simple Net Mgmt Proto
snmp161/udp#Simple Net Mgmt Proto
snmptrap162/udp#Traps for SNMP
```

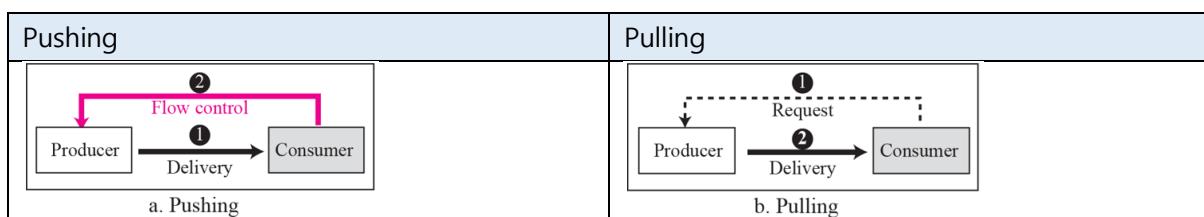
<Socket address>

02-03. Encapsulation and Decapsulation

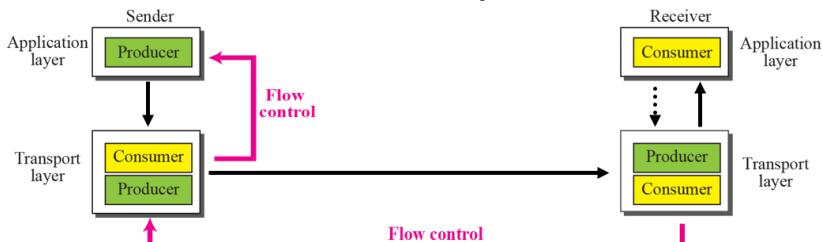
<Encapsulation and Decapsulation>

02-04. Flow and Error Control

<Pushing and Pulling>



<FLOW control at the TRANSPORT layer>



Pushing (Flow control)	송신 측의 Application Layer와 Transport Layer 송신 측과 수신 측의 Transport Layer
Pulling (request)	수신 측의 Transport Layer와 Application Layer

Buffer가 가득 차거나 비어 있을 때 Consumer와 producer와 통신해야 한다.

- Consumer와 producer가 1개의 슬롯이 있는 버퍼를 사용하면 통신이 쉬워진다.
- 전송 측의 transport layer에 있는 이 슬롯이 비어 있으면 그 transport layer는 application layer에 다음 chunk에 대한 note를 보낸다.
- 수신 측의 transport layer에 있는 이 슬롯이 비어 있으면 그 transport layer는 전송 측의 transport layer에 다음 패킷에 대한 acknowledgment를 보낸다.

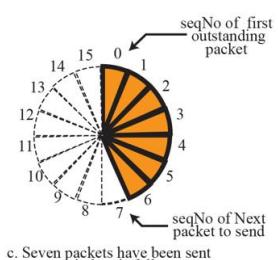
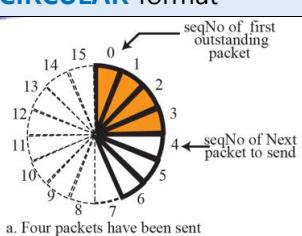
<ERROR control at the TRANSPORT layer>



- Sequence number는 2^m 으로 나눈 나머지이다. (m 은 sequence number 필드의 비트 수)

<Sliding window in CIRCULAR and LINEAR format>

CIRCULAR format



LINEAR format



- a. Four packets have been sent



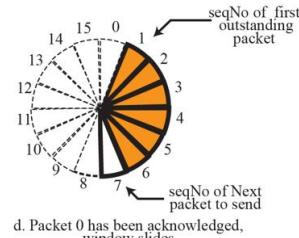
- b. Five packets have been sent



- c. Seven packets have been sent
window is full

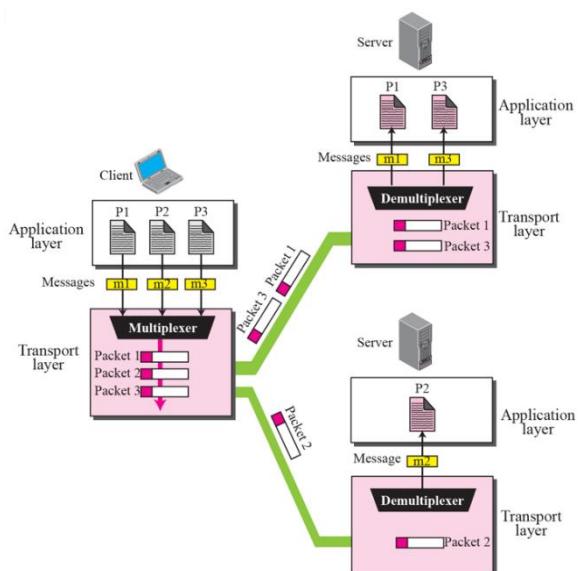


- d. Packet 0 have been acknowledged
and window slid



02-05. Multiplexing and Demultiplexing

<Multiplexing and Demultiplexing>



Multiplexing: application layer에서 여러 개의 페이지를 메시지로 만들어서 합쳐서 패킷으로 변환시키는 것

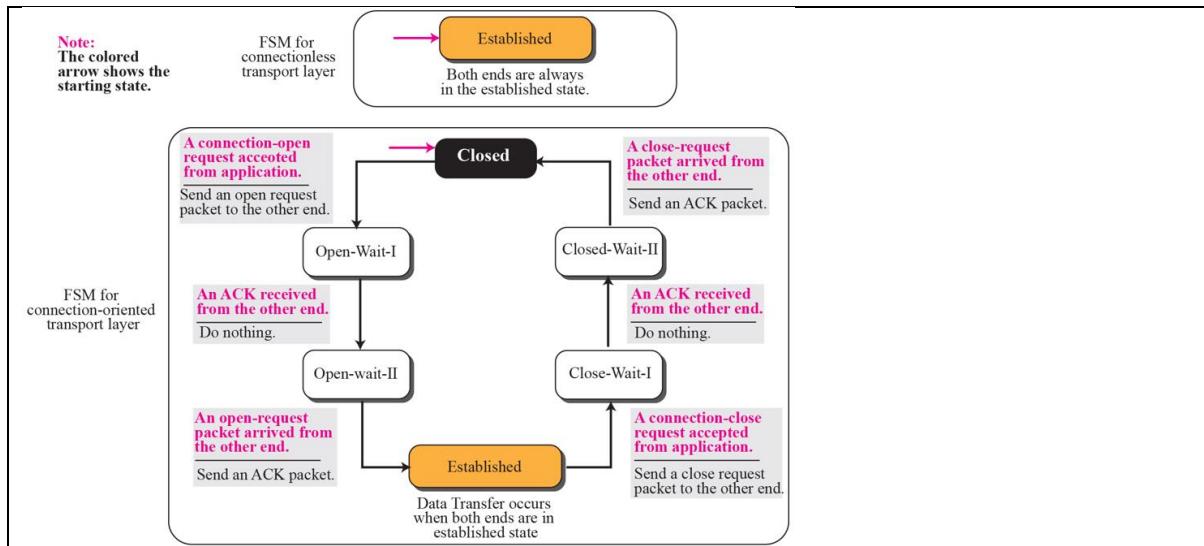
Demultiplexing: Transport layer에서 패킷을 합쳐서 메시지로 만들고, 그 메시지를 페이지로 변환시키는 것

→ 클라이언트에서 여러 개의 패킷을 서버로 보낼 때 각각 서로 다른 demultiplexer로 전송된다.

02-06. Connectionless and Connection-oriented Services

Connectionless service	<p>Sequence diagram for connectionless service:</p> <ul style="list-style-type: none"> Client process → Client transport layer: Message 0, Message 1 Client transport layer → Server transport layer: Packet 0, Packet 1 Server transport layer → Server process: Message 0 Client process → Client transport layer: Message 2 Client transport layer → Server transport layer: Packet 2 Server transport layer → Server process: Message 2 Message 2 is delivered out of order 	Client에서 Message 0, 1, 2를 전달했는데 Packet 1이 Packet 2보다 먼저 도착 하여 Message 2는 순서에 어긋나게 도착하는 것과 같은 유형의 오류 발생
Connection-oriented service	<p>Sequence diagram for connection-oriented service:</p> <ul style="list-style-type: none"> Client process → Client transport layer: Connection-open request Client transport layer → Server transport layer: Connection-open request Server transport layer → Server process: Connection-open request Client process → Client transport layer: ACK Client transport layer → Server transport layer: ACK Client process → Client transport layer: Message 0, Message 1 Client transport layer → Server transport layer: Packet 0, Packet 1 Server transport layer → Server process: Message 0 Client process → Client transport layer: Message 2 Client transport layer → Server transport layer: Packet 2 Server transport layer → Server process: Message 2 Message 1 is held in window Messages 1, 2 delivered Client process → Client transport layer: ACK Client transport layer → Server transport layer: ACK Client process → Client transport layer: Connection-close request Client transport layer → Server transport layer: Connection-close request Server transport layer → Server process: Connection-close request Client process → Client transport layer: ACK Client transport layer → Server transport layer: ACK 	Client와 Server가 connection 을 형성한 후 메시지를 전송한다. 따라서 Packet 2가 도착한 후 전송되는 Packet 1의 메시지인 Message1은 window에 hold 된다.

<Connectionless and connection-oriented services as FSMs>

**03-01. TCP Services**

<Well-known ports used by TCP>

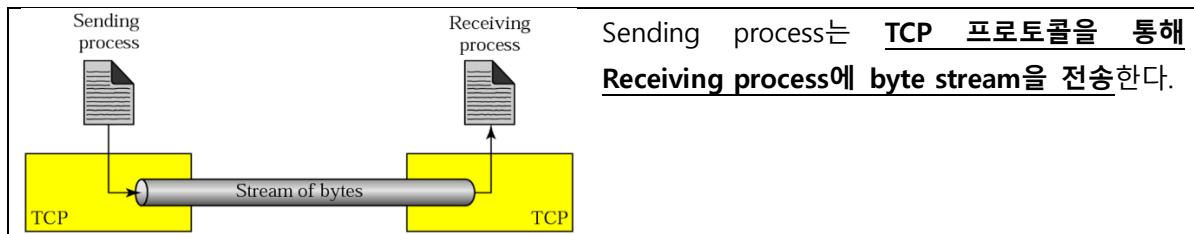
Port	Protocol	설명
7	Echo	수신된 datagram 을 sender 에게 재전송 한다.
9	Discard	수신된 어떤 datagram 이라도 기각 한다.
11	Users	Active users
13	Daytime	날짜와 시간 반환
17	Quote	Day 의 quote 를 반환
19	Chargen	Character 의 string 을 반환
20	FTP, Data	파일 전송 프로토콜 (data connection)
21	FTP, Control	파일 전송 프로토콜 (control connection)
23	TELNET	Terminal Network
25	SMTP	간단한 메일 전송 프로토콜
53	DNS	Domain Name Server
67	BOOTP	Bootstrap Protocol
79	Finger	Finger
80	HTTP	HyperText Transfer Protocol
111	RPC	Remote Procedure Call

<Example 1>

Grep 유ти리티를 통해 목표한 application에 대응되는 line을 추출할 수 있다. 아래는 FTP에 대한 포트 번호이다.

```
$ grep ftp /etc/services
```

<Stream delivery>



03-02. TCP Features

각 connection에서 전송되는 데이터의 각 바이트에는 TCP에 의해 숫자가 매겨진다.

→ 넘버링은 랜덤하게 생성된 특정 값에서 시작된다.

<Example 2>

TCP connection이 5000바이트의 파일을 전송하고 첫 번째 바이트가 10001로 넘버링되었다고 할 때, 파일이 5개의 segment로 쪼개지고 각 segment가 1000바이트이면 sequence number는 어떻게 되는가?

[정답]

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)

필드	나타내는 것
segment에서 sequence number	해당 segment에 포함된 첫 번째 데이터의 number
segment에서 acknowledgment	수신할 것으로 예상되는 다음 byte의 number → Acknowledgment number는 cumulative하다.

03-03. Segment

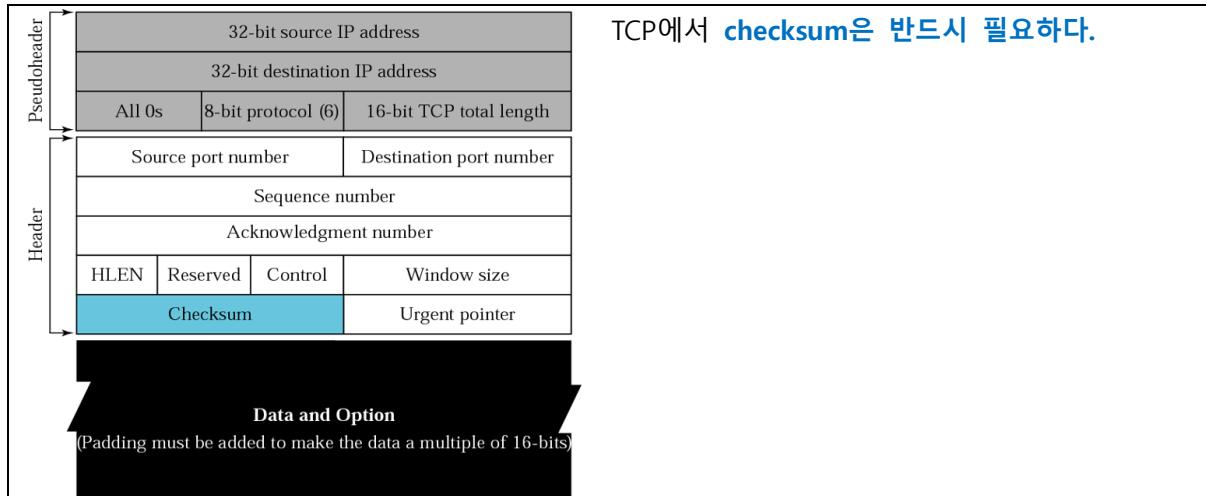
Segment: TCP에서의 패킷

TCP Segment format		Control field
Header	Data	URG Urgent pointer is valid
Source port address 16 bits	Destination port address 16 bits	ACK Acknowledgment is valid
Sequence number 32 bits	Acknowledgment number 32 bits	PSH Request for push
HLEN 4 bits	Reserved 6 bits	RST Reset the connection
URG R C S A P R S T S Y I N F H	Window size 16 bits	SYN Synchronize sequence numbers
Checksum 16 bits	Urgent pointer 16 bits	FIN Terminate the connection
Options and Padding		

<Description of flags in the control field>

URG	Urgent pointer field 의 값이 유효하다.
ACK	Acknowledgment field 의 값이 유효하다.
PSH	Push the data
RST	Connection이 reset 되어야 한다.
SYN	연결 중일 때 sequence number 을 synchronize한다.
FIN	Connection을 종료한다.

<Pseudoheader added to the TCP diagram>

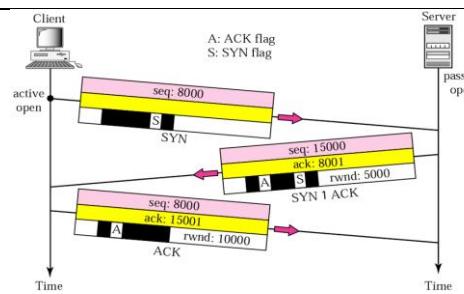


03-04. TCP connection

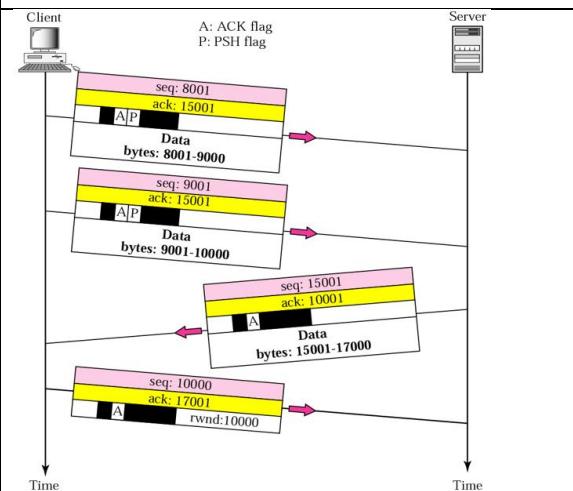
<TCP connection and Data transfer>

TCP connection: Three-way handshaking을

이용하여 이루어진다.



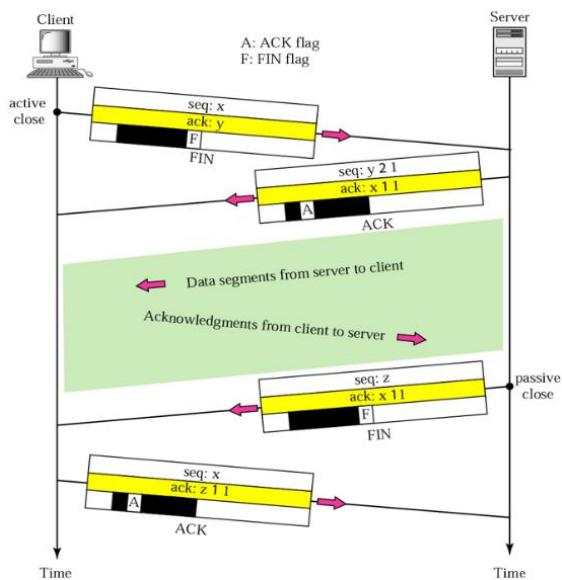
Data transfer



<TCP에서의 segment>

SYN segment	Data를 이동시킬 수 없지만, 1개의 sequence number 를 사용한다.
SYN+ACK segment	
ACK segment	Data를 이동시키지 않으면 sequence number를 사용하지 않는다.
FIN segment	Data를 이동시키지 않으면 sequence number를 사용한다.
FIN+ACK segment	

<Half-close>

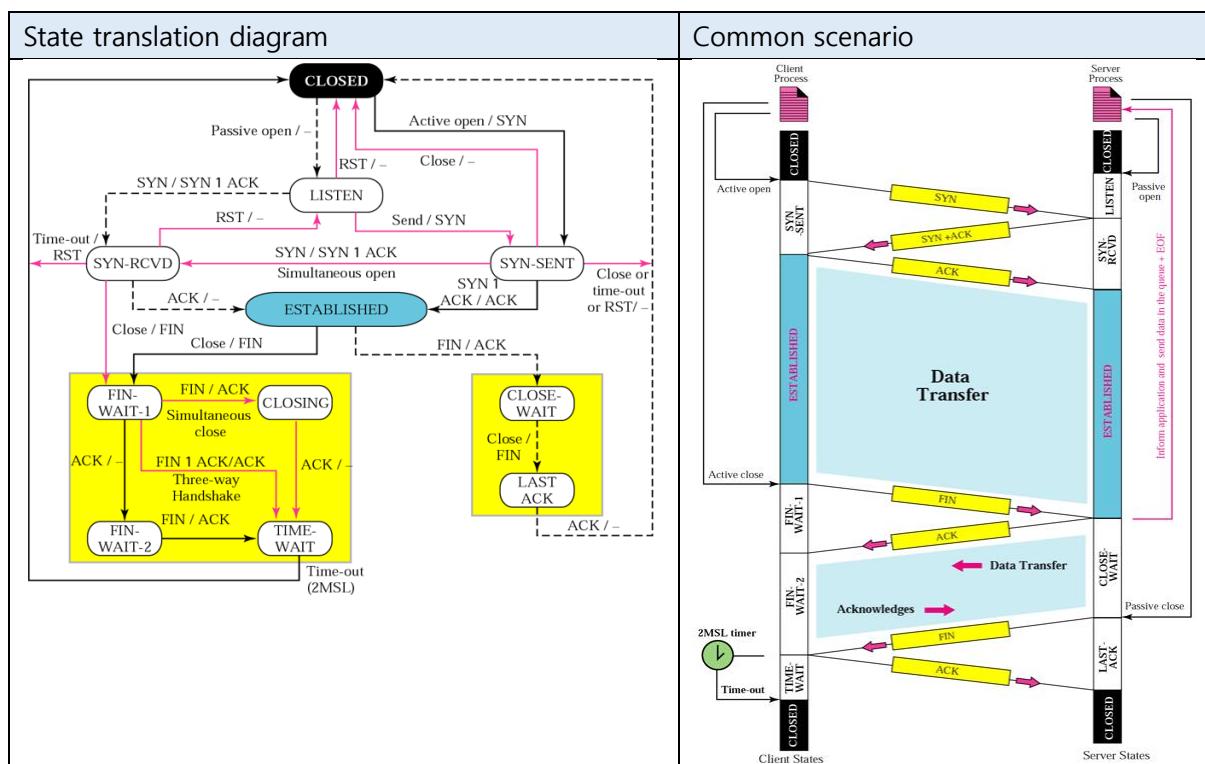


1. Client가 **FIN 패킷을 Server에 전송**하고, 이에 대해 Server는 **ACK 패킷을 Client에게 전송**한다.
2. **Server는 Client에 data segment를 전송**하고, Client는 이에 대한 ACK를 Server에 전송한다.
3. **Server는 Client에 FIN 패킷을 전송**하고, Client는 이에 대한 ACK 패킷을 Server에 전송하여 통신을 종료한다.

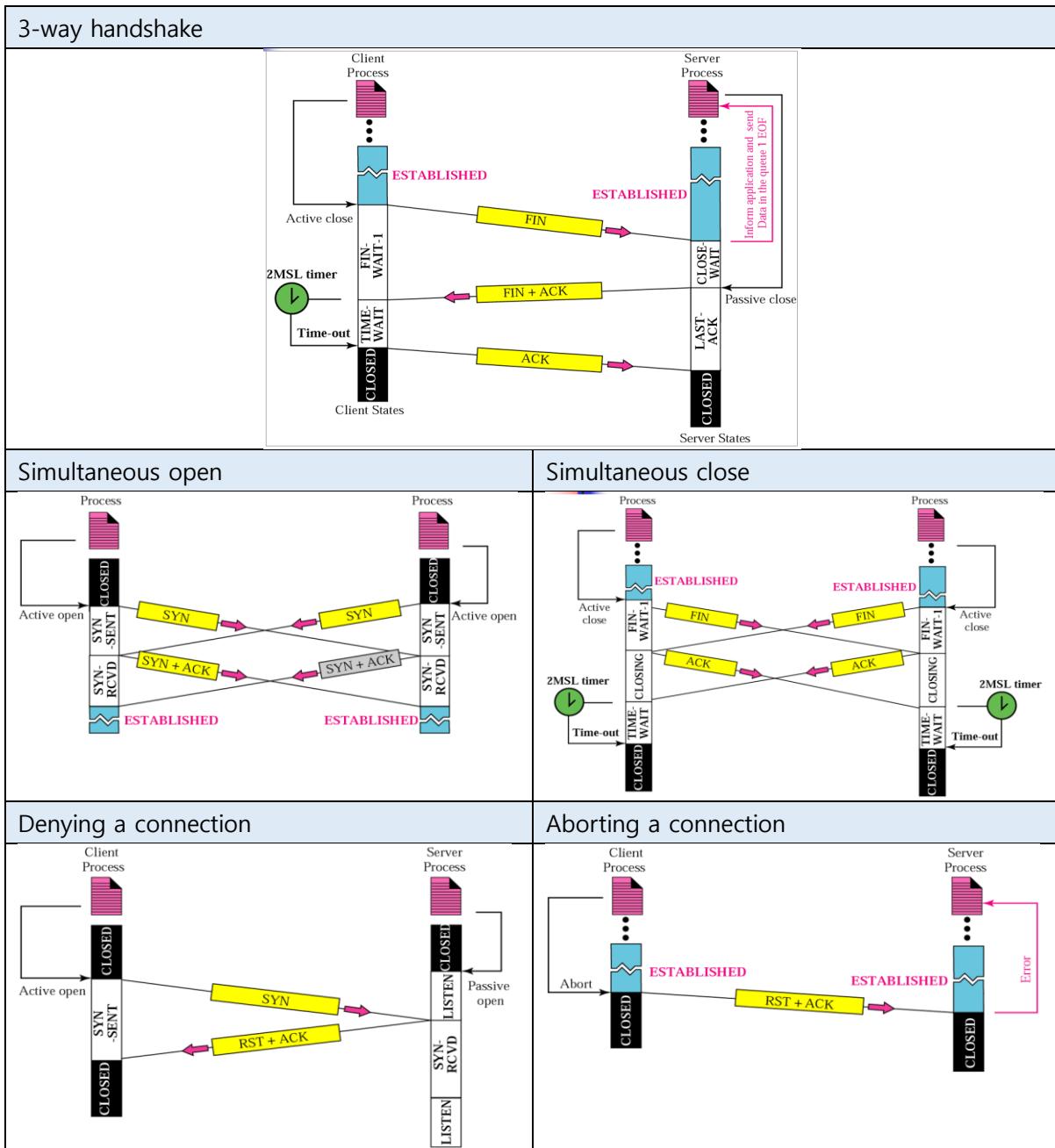
03-05. State Transition Diagram

<State for TCP>

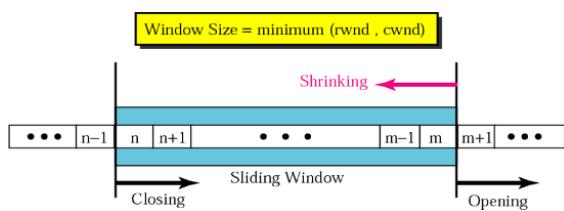
State	Sent	Received	Waiting for	Other
CLOSED				연결 없음
LISTEN		Passive open	SYN	
SYN-SENT	SYN		ACK	
STN-RCVD	SYN+ACK		ACK	
ESTABLISHED				연결 생성됨, 데이터 전송 중
FIN-WAIT-1	1 st FIN		ACK	
FIN-WAIT-2		ACK to 1 st FIN	2 nd FIN	
CLOSE-WAIT	ACK	1 st FIN	앱 종료	
TIME-WAIT	ACK	2 nd FIN	2MSL time-out	
LAST-ACK	2 nd FIN		ACK	
CLOSING				양측이 동시 종료하기로 결정



→ MSL은 보통 30초에서 1분 사이의 값이다.



03-06. Flow Control



Sliding window: 전송을 효과적으로 하면서 데이터의 흐름을 제어할 수 있으므로, destination은 데이터에 지배되지 않는다.

→ TCP의 Sliding window는 byte-oriented이다.

<Example 3>

Receiver (host B)의 버퍼 크기가 5000byte이고 received and unprocessed data의 크기가 1000byte일 때, host A의 receiver window (rwnd) 의 값은?

[정답]

Host B는 $5000 - 1000 = 4000$ byte의 데이터를 버퍼에 더 받을 수 있으므로, rwnd = 4000이다. 이 때 host B는 이 값을 host A로의 다음 segment에 광고한다.

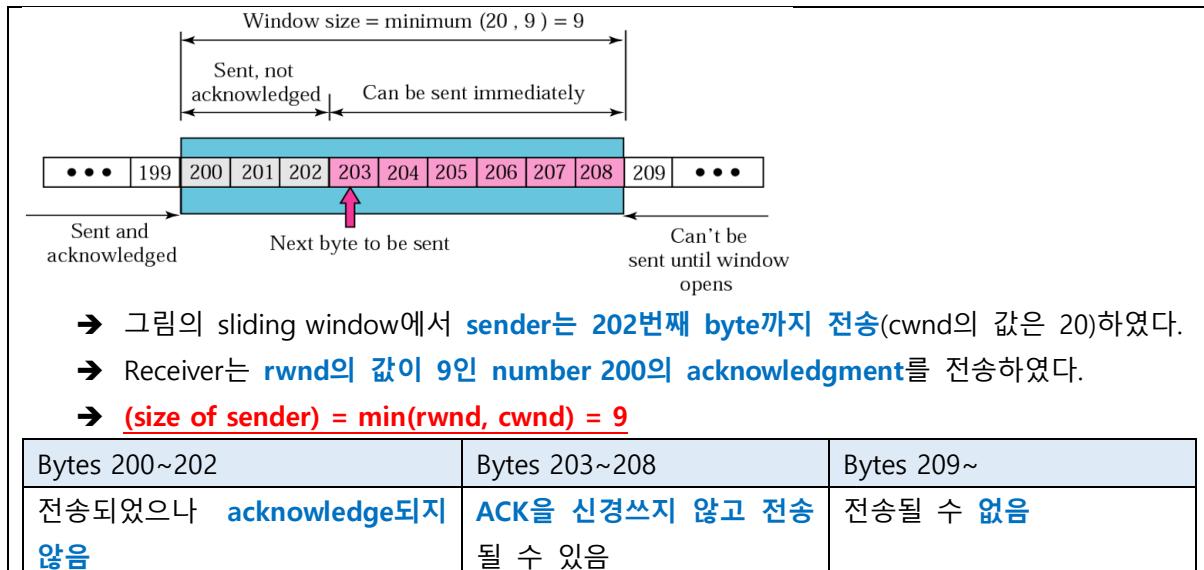
<Example 4>

Rwnd의 값이 3000이고 cwnd의 값이 3500일 때, host A의 window size는?

[정답]

$(\text{size of window}) = \min(\text{rwnd}, \text{cwnd}) = \min(3000, 3500) = 3000 \text{ bytes}$

<Example 5>



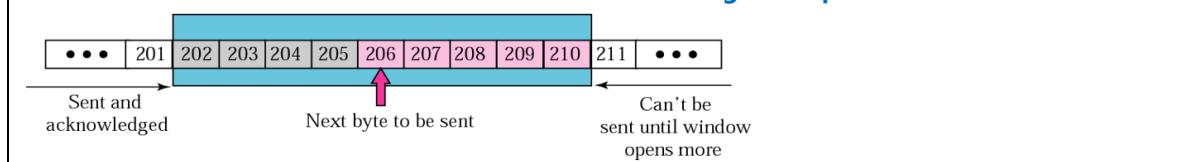
<Example 6>

위 그림에서 Server는 acknowledgement value 202, rwnd 9의 패킷을 수신한다. Host가 byte 203~205를 이미 전송했을 때 cwnd의 값은 여전히 200이다. 이때 새로운 window는?

[정답]

Acknowledgement value 202는 200, 201이 수신되었고 sender는 그것을 더 이상 신경쓰지 않아도 된다는 것이다. 따라서 window는 그것을 넘어갈 수 있다. Size of window는 바뀌지 않는다.

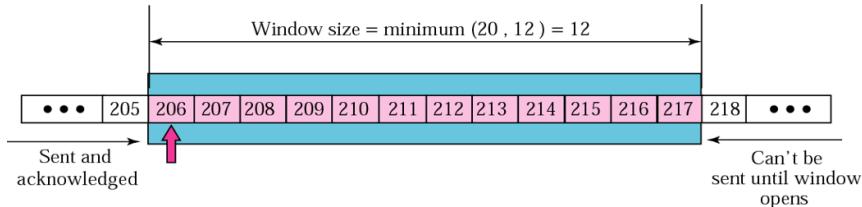
→ 이 경우는 window가 left를 close하고 그만큼 right를 open하는 예시이다.



<Example 7>

위 그림에서 Sender는 acknowledgment value 206, rwnd 12의 패킷을 받는다. 이때 host는 새로운 바이트를 보내지 않았고, cwnd의 값은 여전히 20이다. 이때 새로운 window는?

[정답]

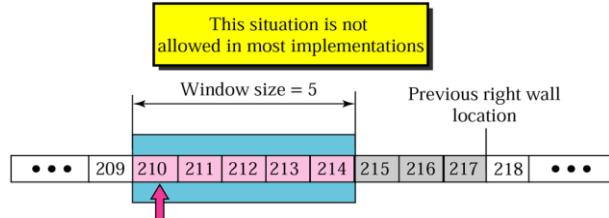


- Acknowledgment value=206이므로 205까지는 sender가 더 이상 신경쓰지 않아도 됨
- Window size = min(rwnd, cwnd) = 12
- 마지막으로 전송된 바이트는 205

<Example 8>

위 그림에서 host는 acknowledgment value 210의 패킷(rwnd=5)을 받았다. Host는 206~209의 바이트를 전송하였으며, cwnd의 값은 여전히 20이다. 이때의 window는?

[정답]



- Acknowledgment value=210이므로 209까지는 sender가 신경쓰지 않아도 됨
- Window size = min(rwnd, cwnd) = 5
- 마지막으로 전송된 바이트는 209

<Example 9>

Receiver가 Example 8에서 window shrinking이 발생하는 것을 어떻게 방지할 수 있는가?

[정답]

Receiver는 last acknowledgment number와 last rwnd를 추적해야 한다.

- Acknowledgment number를 rwnd에 더하면 right wall에 대한 byte number를 얻을 수 있다.
 - Right wall의 shrinking (moving to the left) 을 방지하려면 다음과 같은 관계식을 따라야 한다. (두 식은 동치)
- $$(new \ ack) + (new \ rwnd) \geq (last \ ack) + (last \ rwnd)$$
- $$(new \ rwnd) \geq (last \ ack) + (last \ rwnd) - (new \ ack)$$

- Sender window가 shrinking하는 것을 방지하기 위해서, receiver는 버퍼에 더 많은 공간이 생길 때까지 기다려야 한다.

[TCP Sliding window의 핵심 포인트]

1. Window size는 **min(rwnd, cwnd)**
2. Source는 full window's worth만큼의 데이터를 보낼 필요가 없다.
3. Window는 receiver에 의해 **open/close**될 수 있지만, **shrink**되면 안 된다.
4. Destination은 shrinking window를 유발하지 않는 한 언제든지 **acknowledgment**를 보낼 수 있다.
5. Receiver는 임시적으로 **window**를 닫을 수 있다.
 - A. 그러나 이때도 sender는 **1바이트의 segment**를 항상 전송할 수 있다.