# 눈이 편안한 네트워크 세미나

leeopop

네트워크란?

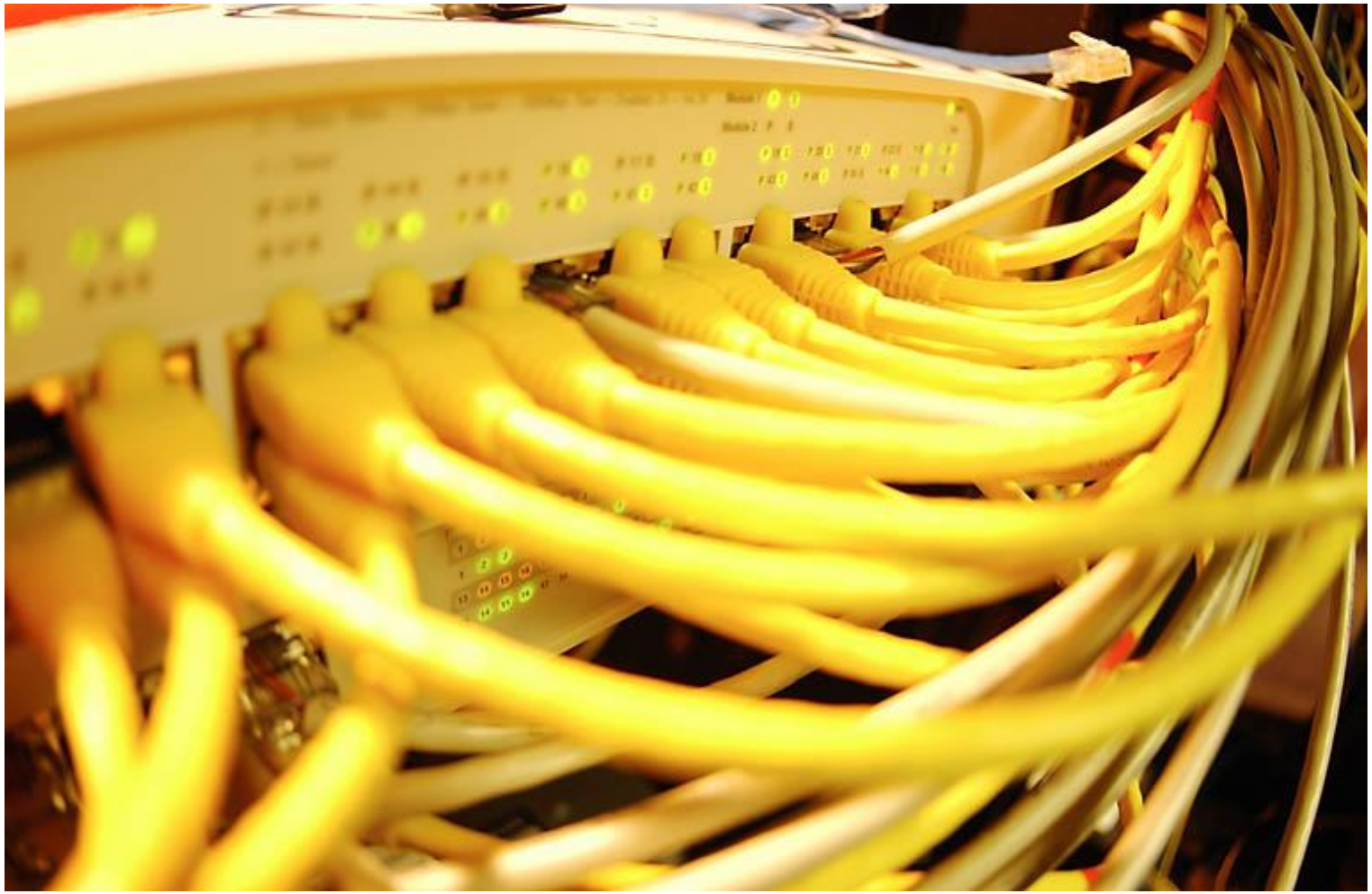
a collection of computers and devices
interconnected by communications channels
that facilitate communications
and allows sharing of resources and information
among interconnected devices

**자원과 정보를 공유할 수 있게 서로 연결된 컴퓨터들의 집합**

# 세미나 순서

OSI level 1부터 7까지 달립니다.

시간 많아요 ㅋㅋ



쌀자루 회원의 TCP세미나서 발췌


위키피디아 : http://en.wikipedia.org/wiki/Category_5_cable

랜선... 과연 올바른 명칭일까요??
아닙니다...

**Category 5 Twisted Pair Cable 이 정확한 명칭입니다 >< **

100 Mbps – 무엇의 약자일까요??

**100 Mbps 케이블의 선에는 100MHz의 교류 전류가 흐르게 됩니다.**

고작 **0.5mm 간격**으로 붙어있는 전선에 **1V이상**의 단위로 **100MHz**의 전류가 **구리선** 위에서 흐르면 그 **자기장**이 엄청나겠죠??

무슨 뜻인지 모르시면... 물리 1,2 책을 보시는 것을 추천하고 싶습니다...

Twisted Pair 케이블에서는 +, - 반대 방향의 전류가 흐르는 케이블 **두 개를 꼬아서** **자기장을 서로 상쇄시킵니다.**

**자, 그러면 랜선에는 모두 몇 개의 선이 들어있을까요??**

그렇다면 그 중 실제로 사용하는 선은 몇 개 있을까요?

**하나의 신호를 전송하기 위해서 무조건 2개의 전선이 필요합니다. (Twisted Cable)**

즉, 랜선.. 아.. 흔히 UTP케이블이라고 합니다. Unshielded Twisted Pair

UTP케이블 하나로 **최대 4가지**의 서로 다른 전기적 신호를 전송할 수 있습니다.

님들아. 전화선 보면 단자가 4개잖아요..

얘는 category 1 cable입니다...

저는 그리 유복한 환경에서 자라지 못해서 모뎀(모듈레이터 + 디모듈레이터)
이라는 구시대 장비로 인터넷을 했었습니다.

근데 이게 전화랑 인터넷을 동시에 쓸 수 없어요...
근데 어느날 하X로 인터넷 아저씨가 와서 전화랑 인터넷을 둘 다 쓸 수 있게
해 주셨어요!

**하얀 바탕에 줄그어져 있는 애랑**
**그 줄 색깔인 애랑 짝이에요**


랜선은 이 중

초록, 주황을 씁니다
(1,2), (3,6)

그러면 cross cable은 반대편 연결이 어떻게 될까요?

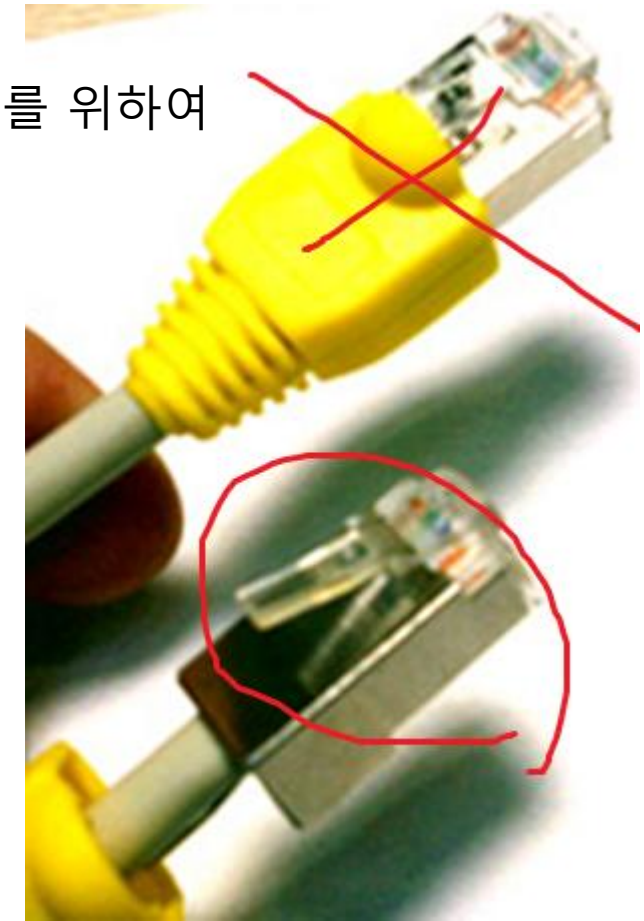
아.. 색깔은 선 규격마다 다른데
**흰색바탕에 줄이랑 단색이랑 서로 짝이에요**

기가비트 랜카드를 고민하는 분들

# 진짜 기가비트를 쓸 수 있습니까??

100MHz 이상의 신호를 전송하려면 전자기장 차폐를 위하여

**은박지로 쌉니다**

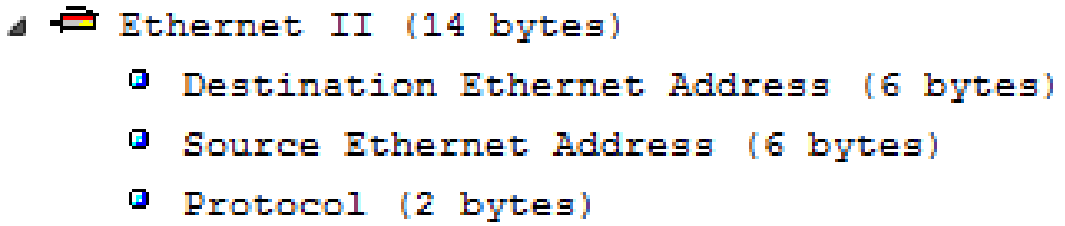



아마 휠이 되면 랜선정도는 직접 만들어 쓸 날이 곧 올껍니다...

**여기까지 OSI level 1이었습니다.**

이더넷 : OSI level 2 중에 하나로 무선랜 프로토콜 등도 포함됩니다.

Ethernet transmission layer (not necessarily accessible to the user):
48.bit: Ethernet address of destination
48.bit: Ethernet address of sender
16.bit: Protocol type = ether_type




00:30:48:83:09:EA (Supermicro Computer, Inc.) (143.248.234.158)

48:5B:39:6F:7B:47 (143.248.234.126)

0x0800 IPv4

이더넷 바로 뒤에 후속 프로토콜이 등장!

| | |
|---|---|
| Internet Protocol Version 4 (IPV4) (20 bytes) | |
| Version (4 bits) | 4 |
| Header Length (4 bits) | 5 |
| Differentiated Services Field DSCP=0x00 (Default) | ECN=0x00 |
| Total Length (2 bytes) | 52 |
| Identification (2 bytes) | 0x269A |
| Fragmentation Flags (3 bits) | 010 (Don't Fragment: on, |
| Fragment offset (13 bits) | 0 |
| Time to Live (1 byte) | 128 |
| Protocol (1 byte) | 0x06 (6) (TCP) |
| Checksum (2 bytes) | 0xDF1B (Correct) |
| Source IP address (4 bytes) | 143.248.234.126 |
| Destination IP address (4 bytes) | 143.248.234.158 |

실제로는 이렇게 그냥 붙어있습니다. 나뉘어 있지도 않고요

여기까지 이더넷

```
0x0000   00 30 48 83 09 EA 48 5B 39 6F 7B 47 08 00 45 00
0x0010   00 34 26 9A 40 00 80 06 DF 1B 8F F8 EA 7E 8F F8
0x0020   EA 9E                                5C 72 F1 00 00 00 00 80 02
0x0030   20 00                                04 05 B4 01 03 03 02 01 01
0x0040   04 02
```

여기까지 인터넷

그냥 첫 14바이트가 이더넷이고 그 다음 20바이트가 인터넷,
그리고 그 다음 20바이트가 TCP/UDP입니다.

이거 값을 알고 싶다면??

그냥 다음과 같이 하면 됩니다.

```
struct ethernet
{
        char source[6];
        char destination[6];
        short protocol;
};

ethernet buffer;
read(socket, (char*)&buffer, 14);
```

그냥 구조체에 14바이트 읽으면 끝!

참 쉽죠?

허브의 역할??

**전기적 신호를 여러 갈래로 “복사” 해 준다.**

스위치 (L2 스위치)의 역할??

**Level 2 프로토콜 (이더넷) 신호를 읽고 분석해서**
**올바른 주소를 가진 랜선에’만’ 신호를 전달해 준다.**

# ARP ( Address Resolution Protocol )

자세히 다루지는 않겠지 "나는 경기도 안양의 이준영이다!"
식으로 "지금 이 전선이 바로 이 맥 주소 입니다" 라고
알려주는 프로토콜 입니다

ARP spoofing이란 – A 에서 B로 가야 하는 패킷이 있는데 C라는 컴퓨터가 "내가 B 이다" 라고 스위치한테 알려줘서 A에서 B로 가는 패킷이 C로 가도록 가로채는 기법입니다. 요즘 똑똑한 스위치는 잘 잡아요.

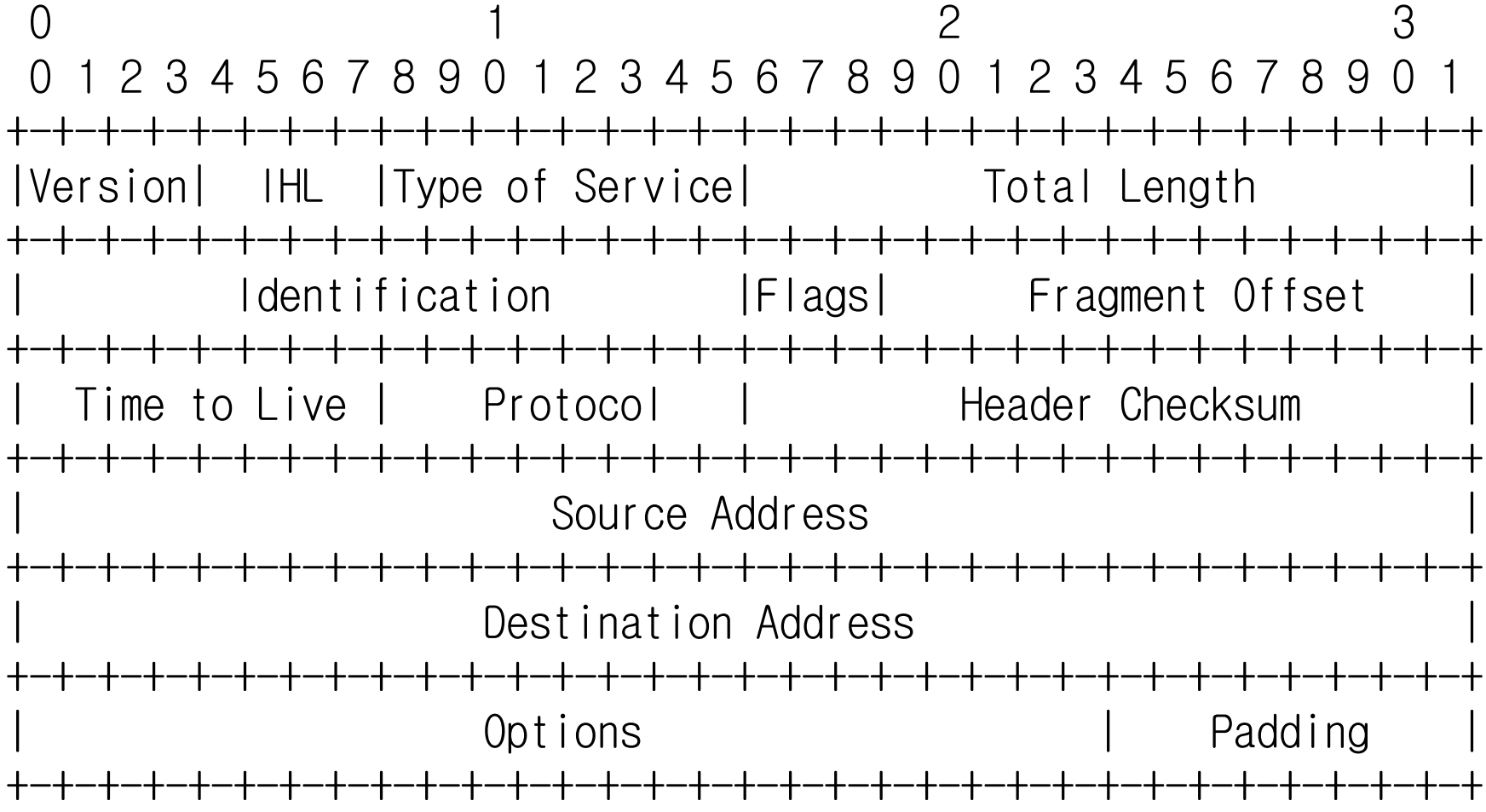
여기서 배운 내용이 바로

# OSI layer 2

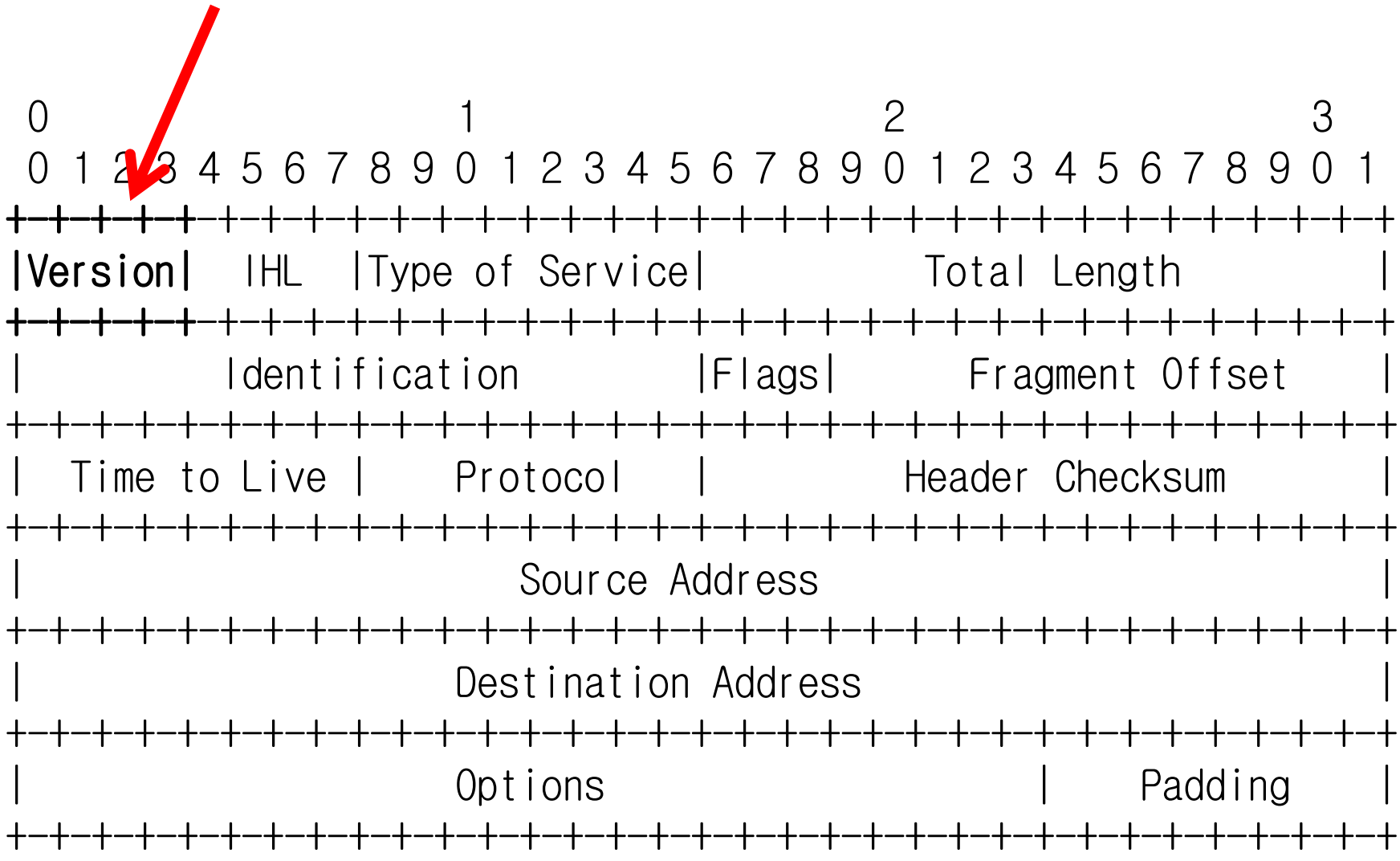
그렇다면 L2 스위치란? Ethernet 프로토콜 내용을 다루는 스위치 – 맥 주소를 다룬다.

Level 2에서 패킷을 작성하고 싶다면??
CAP_NET_RAW 권한이 있어야 한다!

# 자 인터넷 프로토콜 표준을 볼까요?

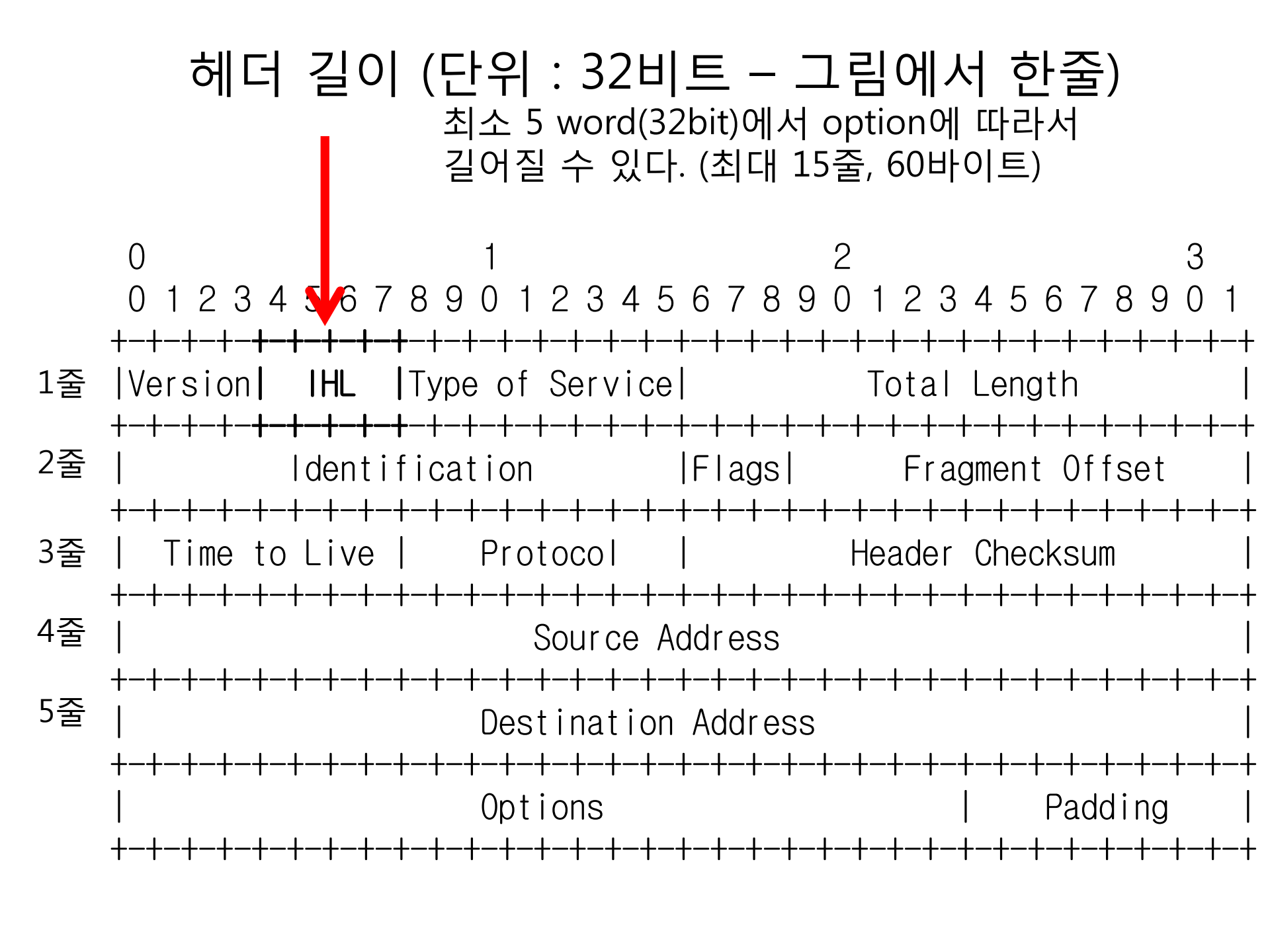


# 버전 정보 : 첫 4비트 – 이론상으로 IPv16까지

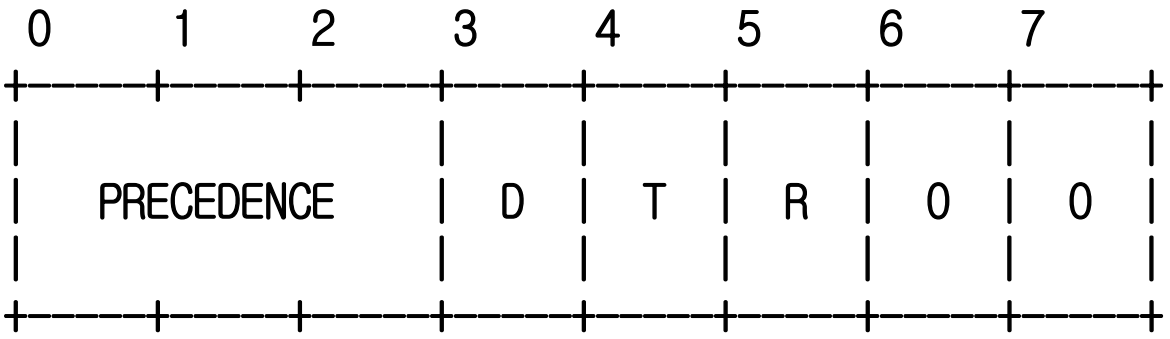


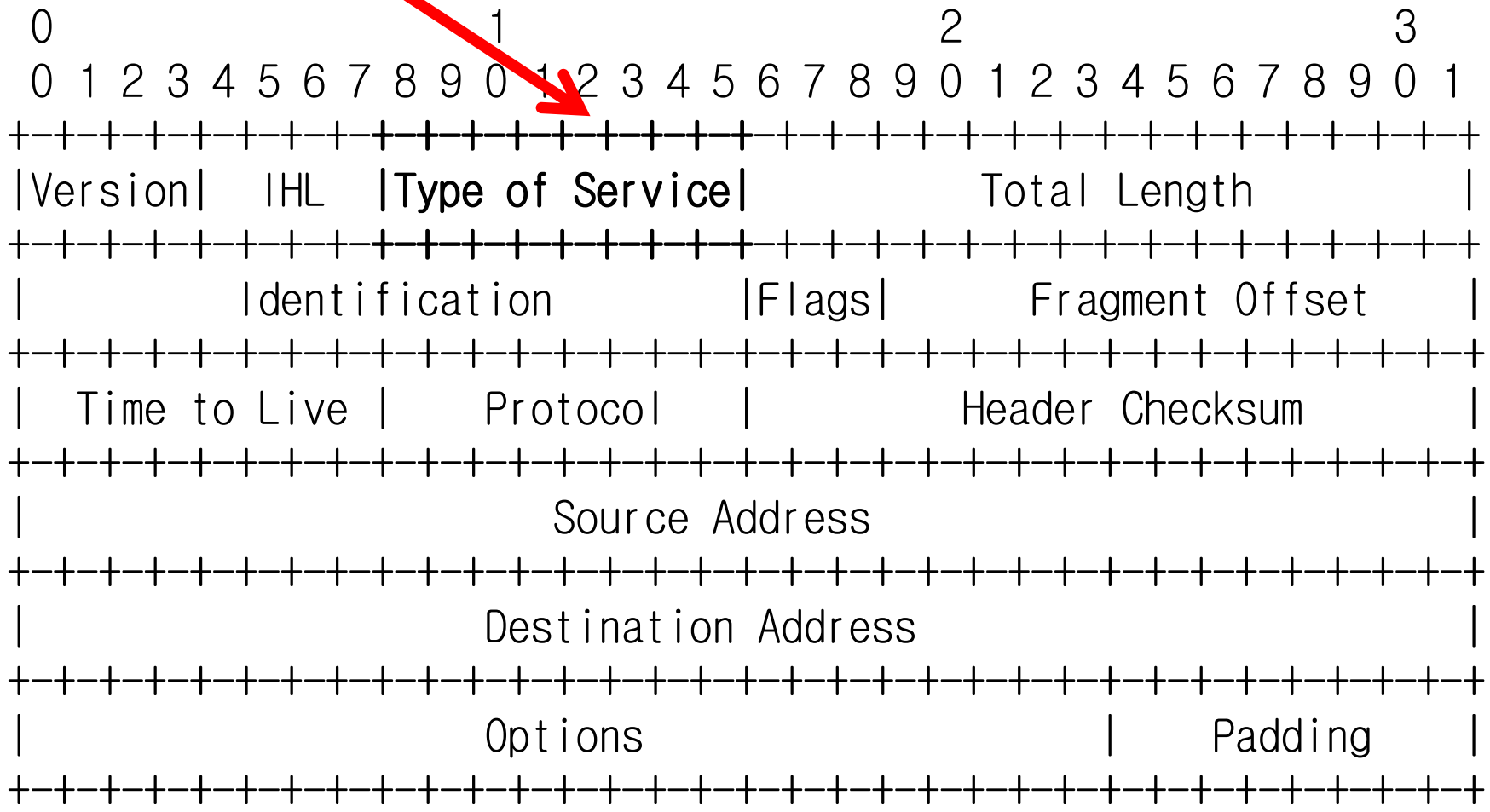
# 헤더 길이 (단위 : 32비트 – 그림에서 한줄)

최소 5 word(32bit)에서 option에 따라서
길어질 수 있다. (최대 15줄, 60바이트)

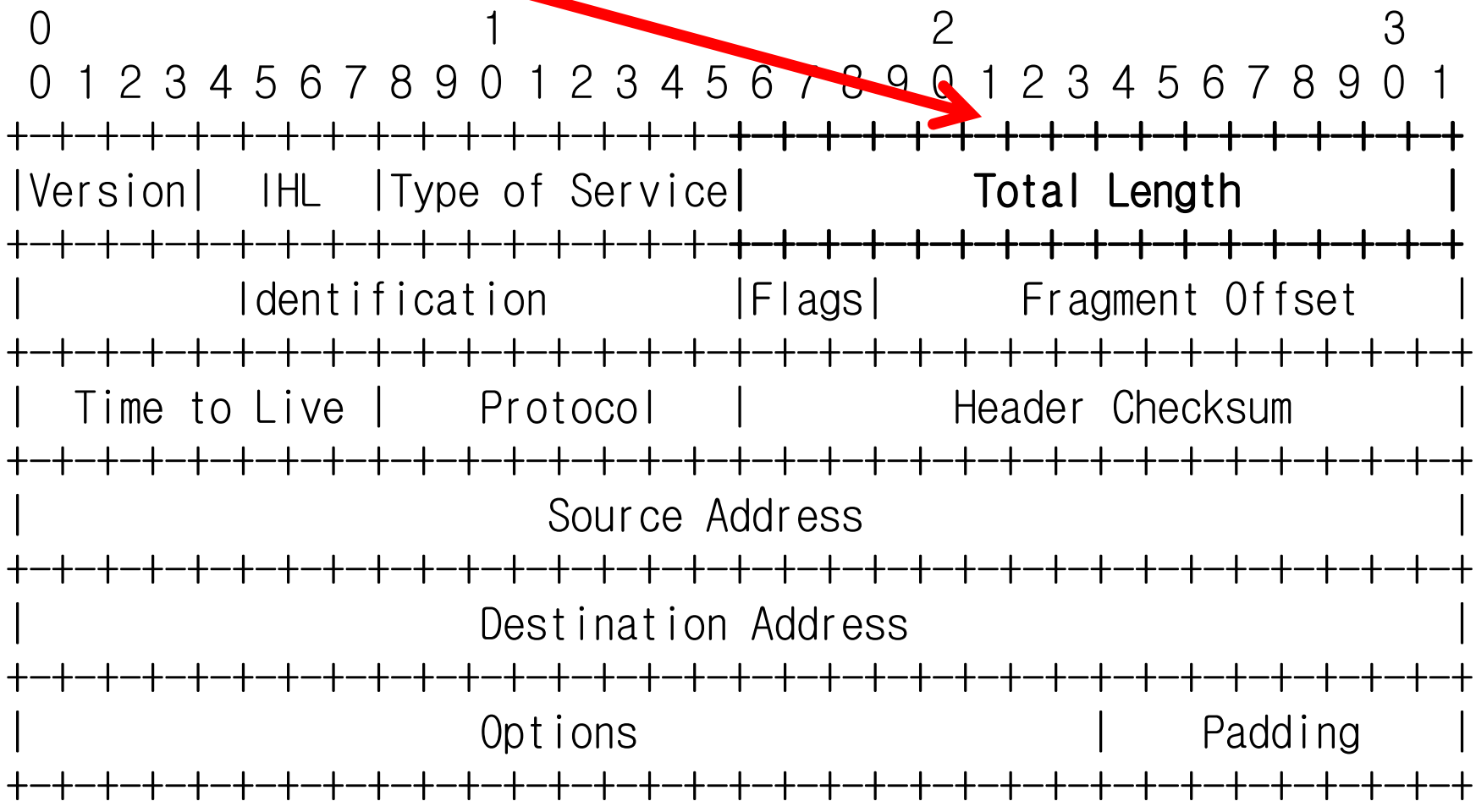


서비스 종류


0 1 2 3 4 5 6 7
PRECEDENCE
D
T
R
0
0


0 1 2 3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
Version
IHL
Type of Service
Total Length
Identification
Flags
Fragment Offset
Time to Live
Protocol
Header Checksum
Source Address
Destination Address
Options
Padding

"전체 길이" : 인터넷 헤더 + 뒤에 있는것들
인터넷 헤더 + TCP 헤더 + 데이터



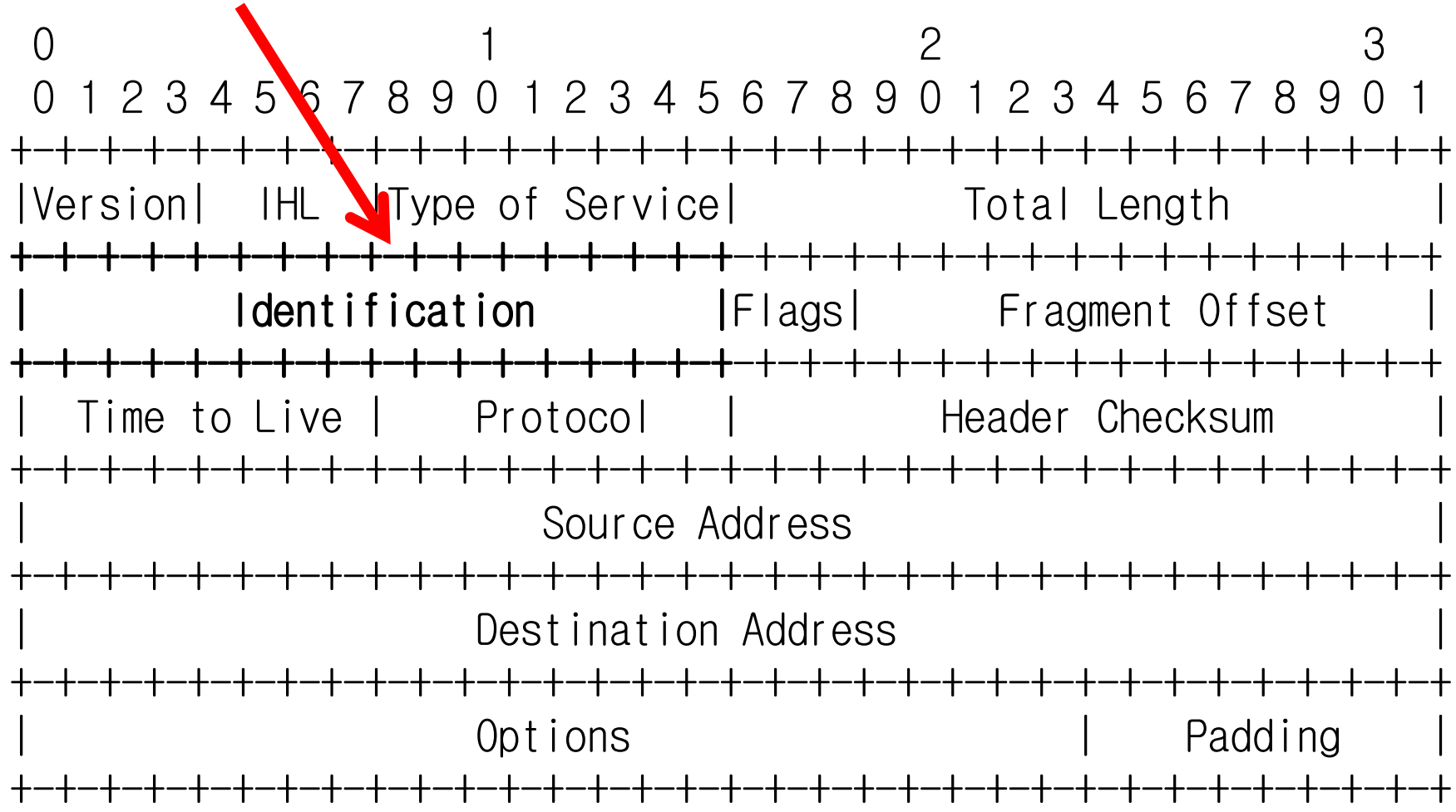
# 휴... 한줄 했네요...

이런걸 왜 하지... 라는 생각이 많이 드시겠지만...

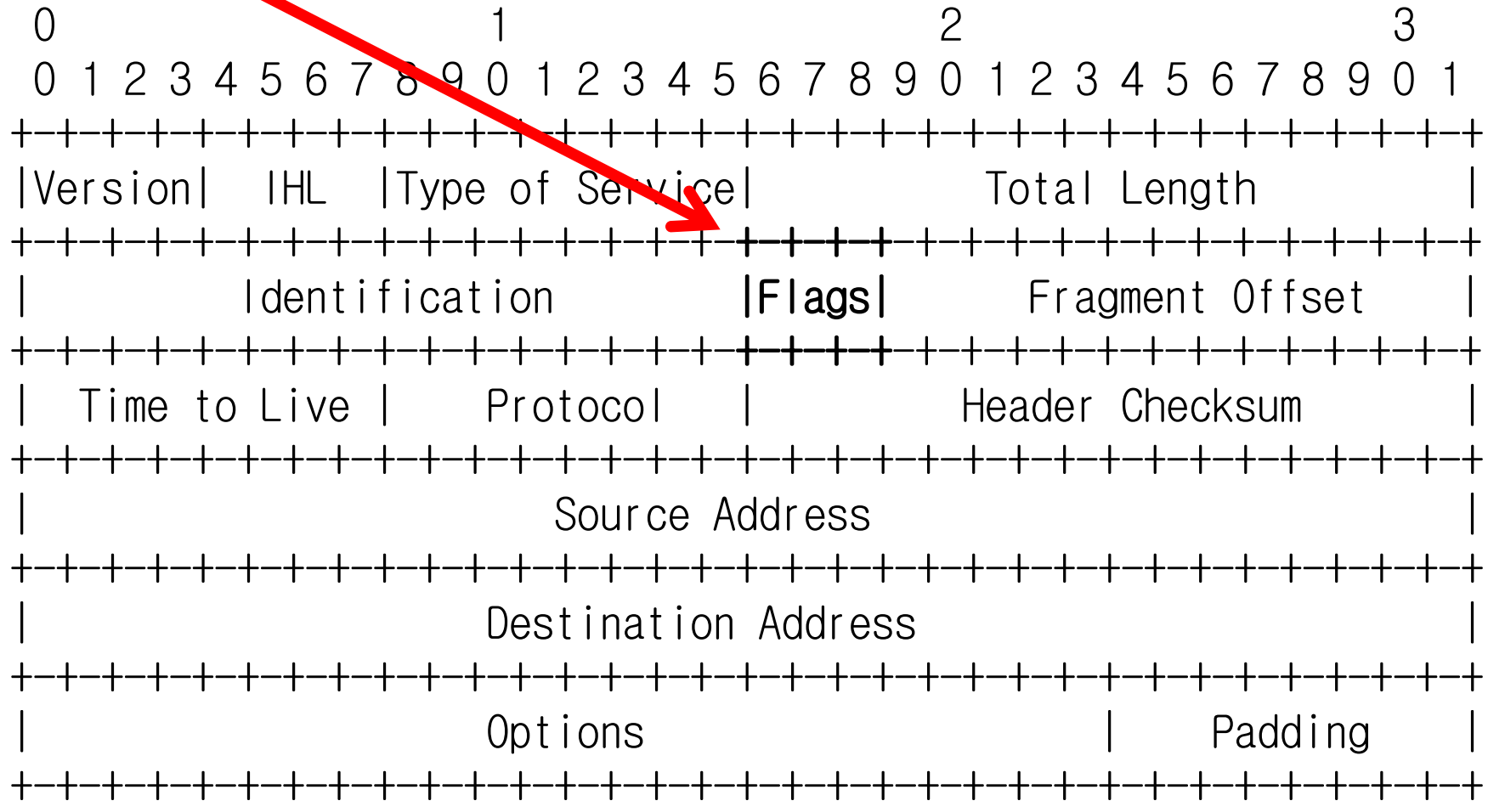
사실 ifconfig eth0 192.168.0.1 이것만 쳐도 되거든요...

그래도 tracert 하면 어떻게 라우터를 찾는지를 알면 더욱 좋겠죠

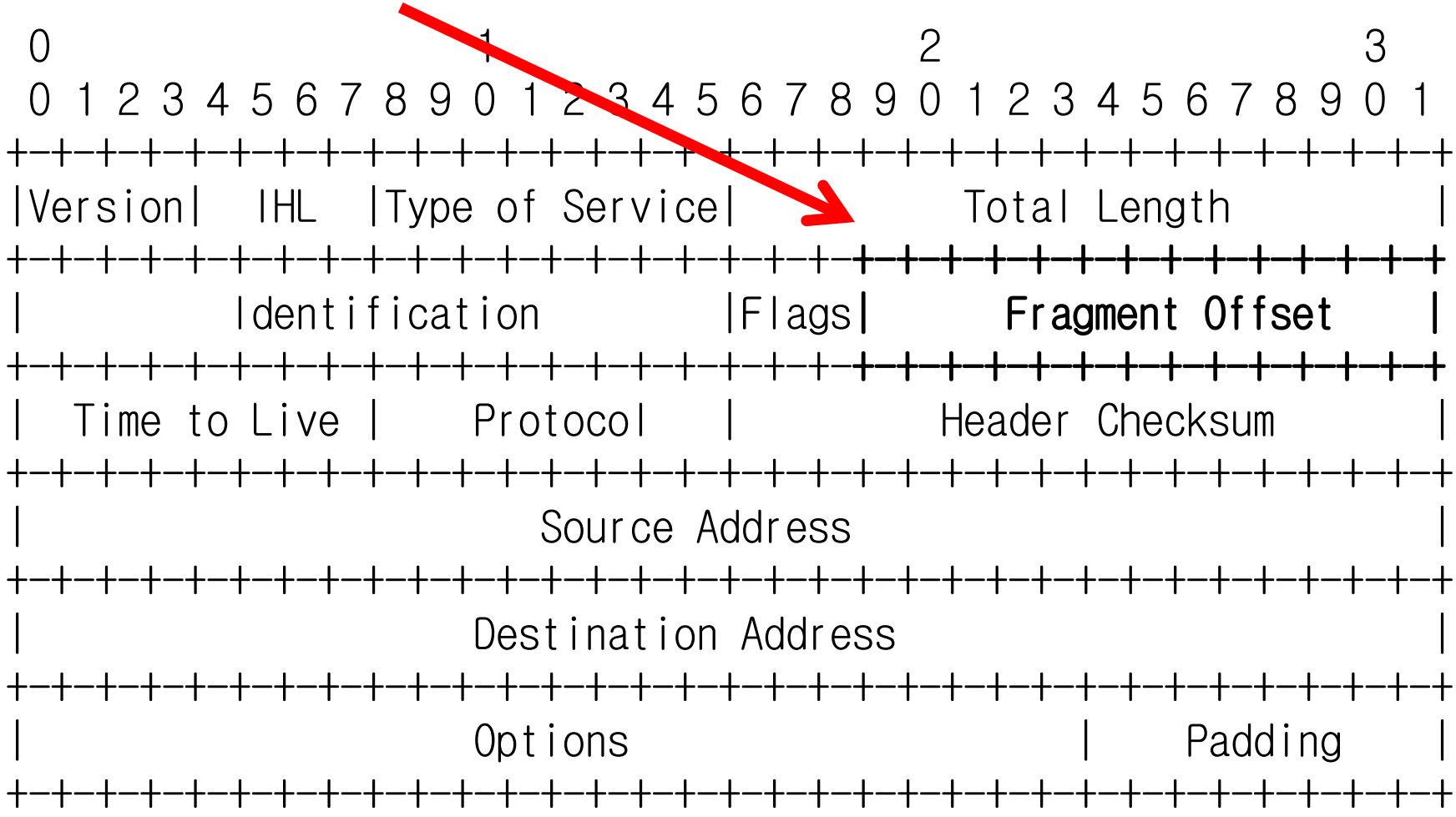
패킷 식별 번호 : 패킷은 쪼개질 수도 있어요
이 때, **"같은 패키지 상품입니다"** 라고 말해주는 값



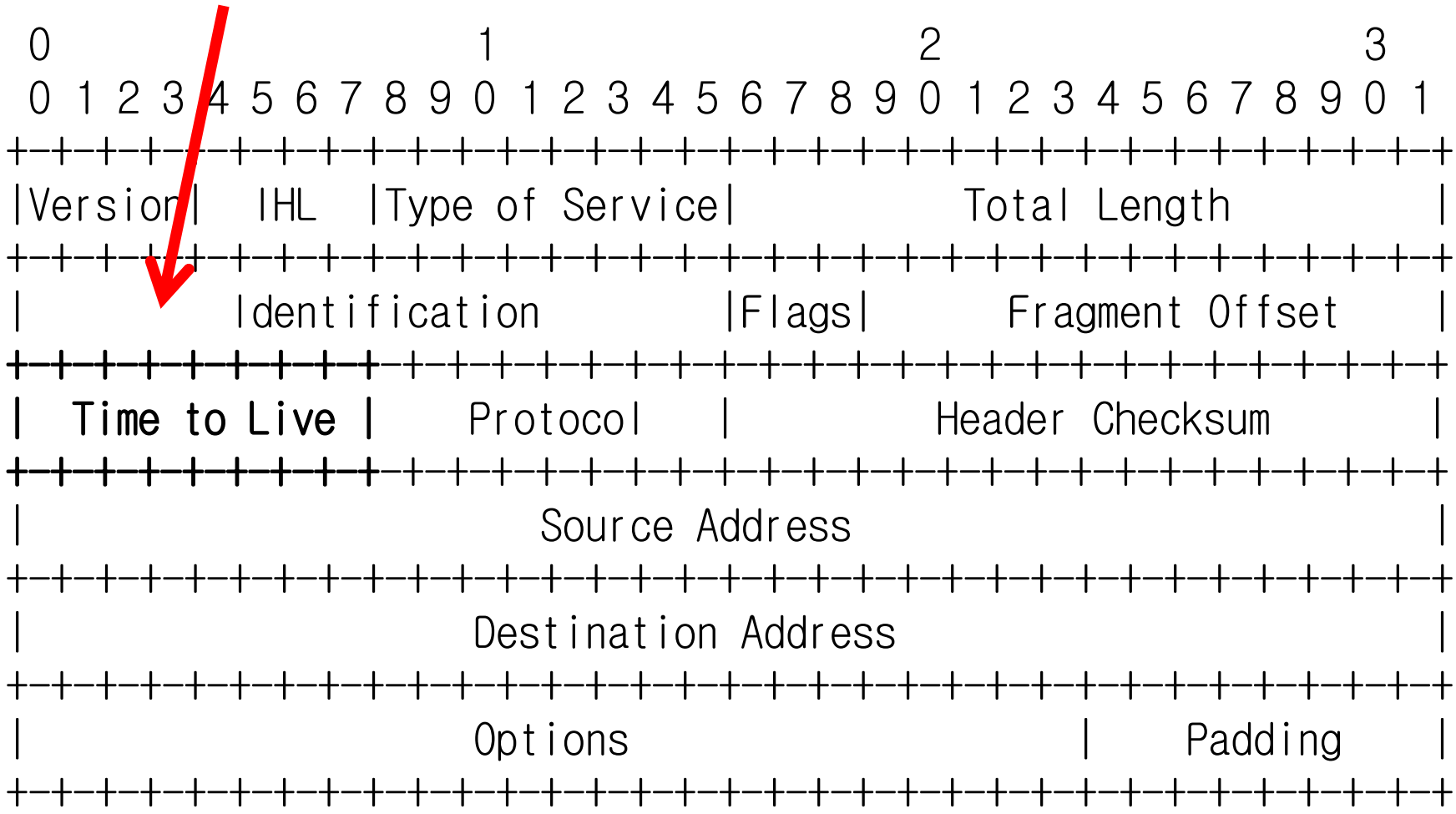
이 쪼개짐을 위한 플래그 입니다(3bit)
첫 bit는 안쓰고요;; (무조건 0)
두번째 bit는 "쪼개짐 여부" 세번째 bit는 "마지막 조각"



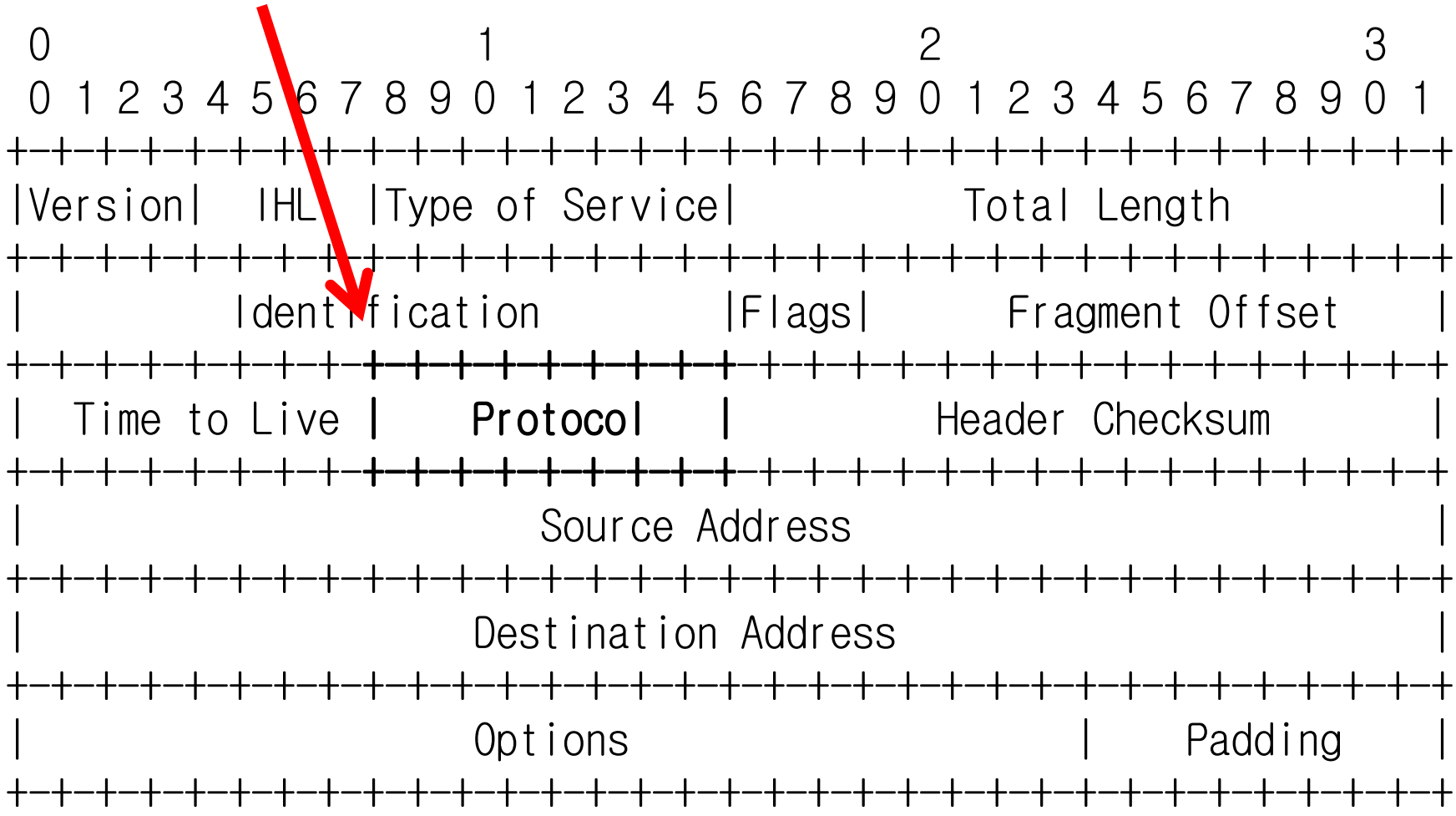
내가 n번째 조각입니다. 라고 하는 값입니다.



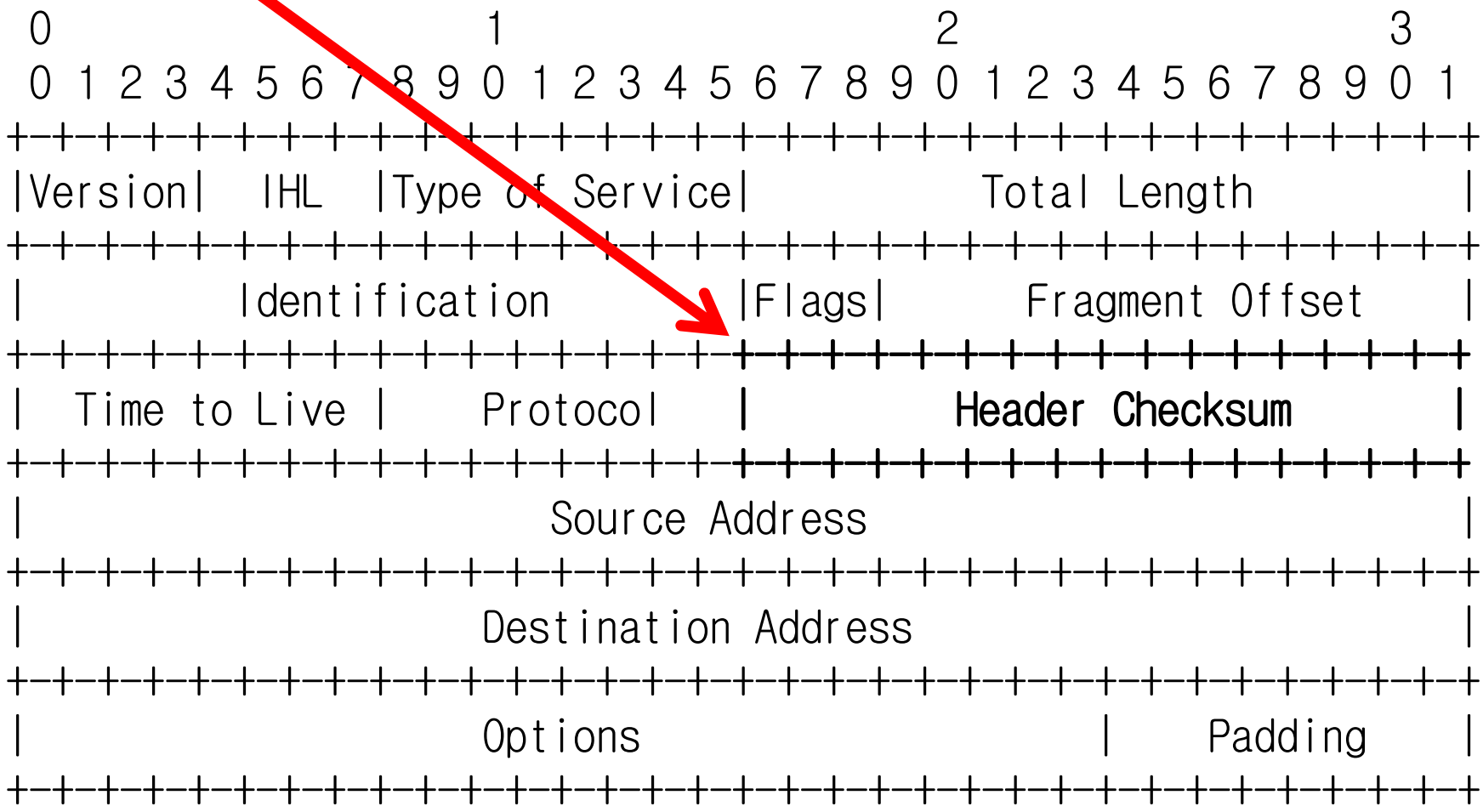
패킷이 앞으로 생존할 수 있는 거리를 뜻합니다.
라우터를 지나면 1씩 감소하며 0이 되면 소멸합니다.
이게 없다면, 패킷이 무한이 많아지겠죠?



다음에 TCP가 이어진다, UDP가 이어진다
이런 정보를 담고 있습니다.



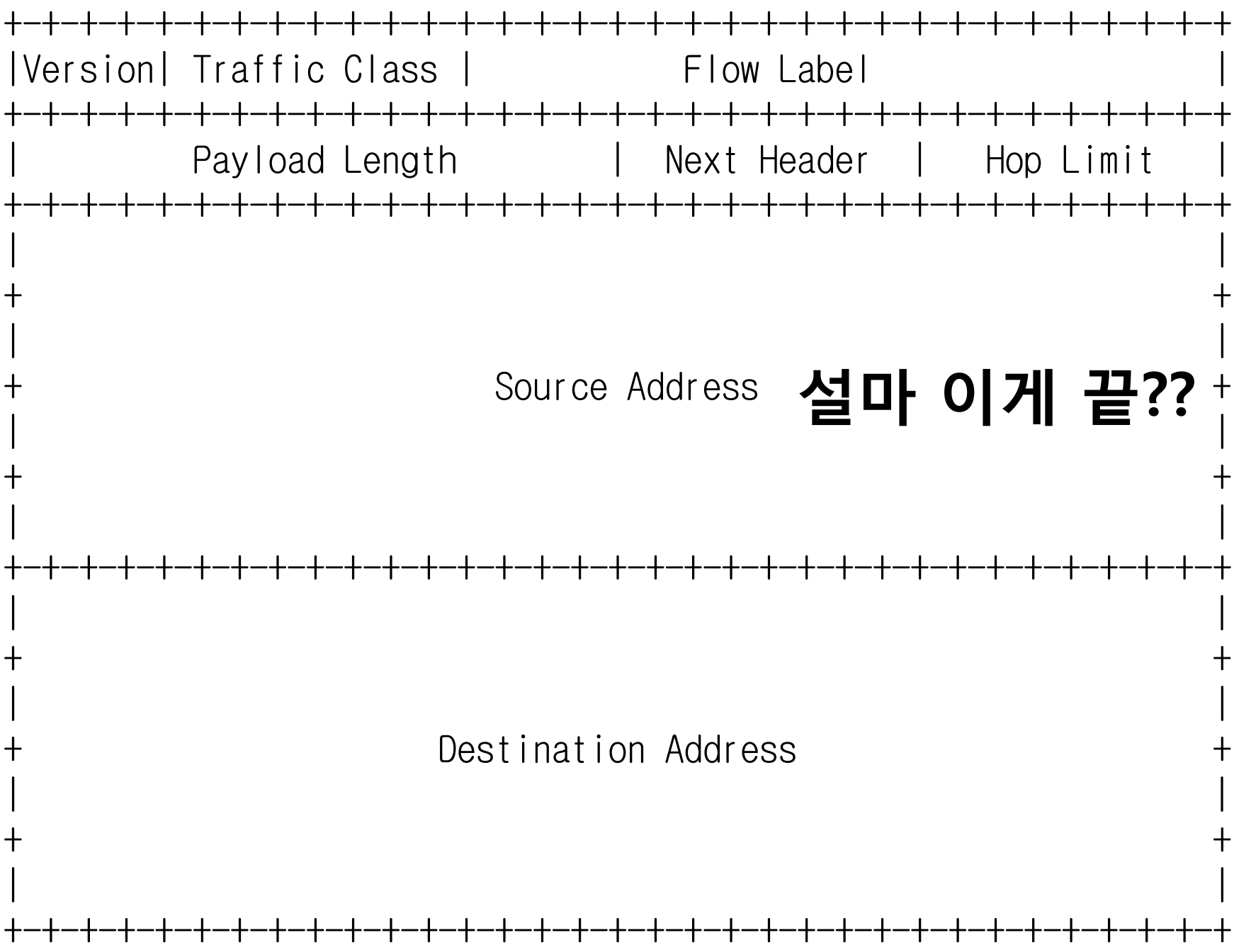
체크섬 계산 방법 : 체크섬을 0이라고 가정하고
16비트 단위로 1의 보수로 덧셈을 하고
마지막으로 다시 1의 보수를 취한다.



```
int sum = 0;
while(in.hasNextShort())
{

        short temp = in.nextShort();
        sum += temp;
}

sum = sum >> 16 + sum & 0xFFFF;
sum = sum >> 16 + sum & 0xFFFF;

return (short)sum;
```

이제 서버들이 IPv6를 달아야 할지도 몰라요

IPv6에 대하여 알아봅시다.


Version
Traffic Class
Flow Label
Payload Length
Next Header
Hop Limit
Source Address
설마 이게 끝??
Destination Address

# 네, 끝입니다.

**다른 정보들도 중요해 보이는데 그 정보들은 어디에??**

NEXT HEADER 필드가 바로 다음 정보를 가리킵니다.

마치 데이터 구조 시간에 배운 링크드 리스트 처럼요

IPv6와 IPv4를 동시에 사용할 수 있나요? **예**

하나의 랜 카드에서 이더넷 패킷을 받고
거기에 써 있는 프로토콜을 보고
IPv4와 IPv6를 분리해서 마치 두 개의 랜카드가 있는 것 처럼

=> 듀얼 스택 기법

MTU란 무엇일까요? **최대 전송 단위**

1024 길이의 데이터를 보내고 싶은데 MTU가 512 면 두번에 나눠서 보내야 합니다.

여기서 배운 내용이 바로

# OSI layer 3

그렇다면 L3 스위치란? Internet 프로토콜 내용을 다루는 스위치 – 아이피 주소를 다룬다.

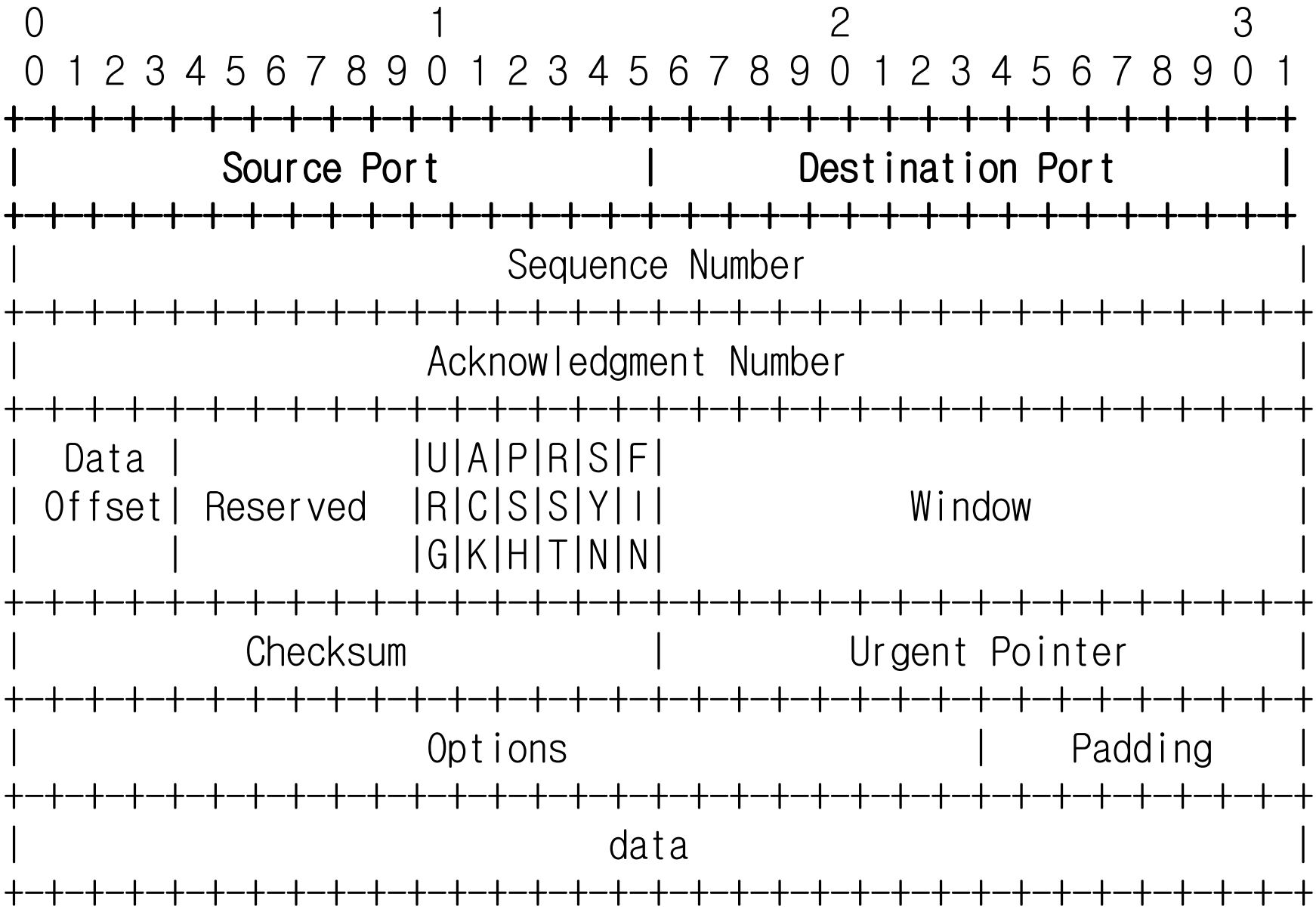
이 레벨에서부터 “라우팅” 이라는 것이 가능해짐

123.xxx.xxx.xxx 를 한국으로 보내고
124.xxx.xxx.xxx 를 일본으로 보내는 일이 가능

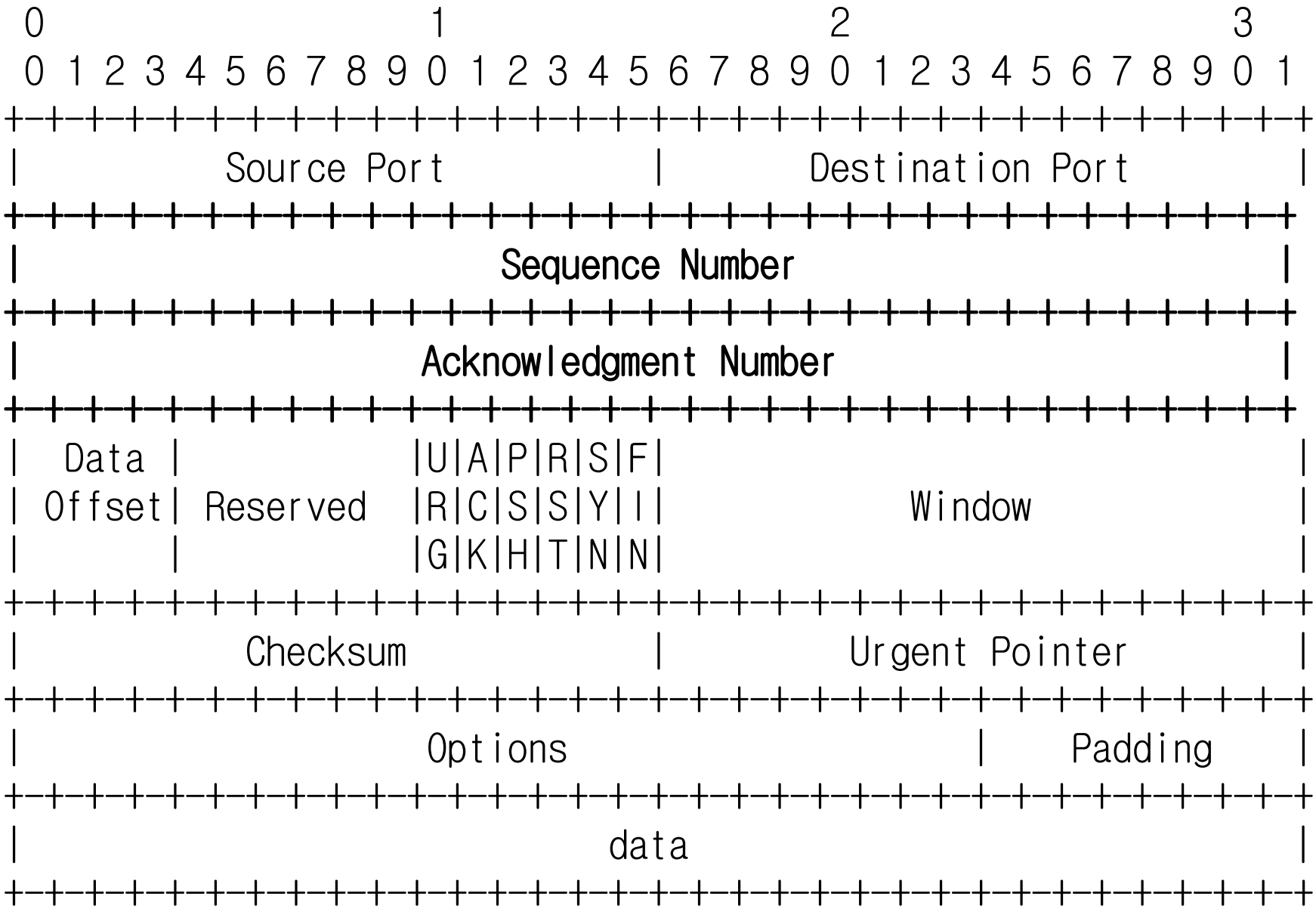
아.. 힘들다...

제가 설마 layer 7 까지 하겠습니까? 4까지만 하고
실제 휠로서 사용해야 할 것 같은 내용으로 가도록 하겠습니다.

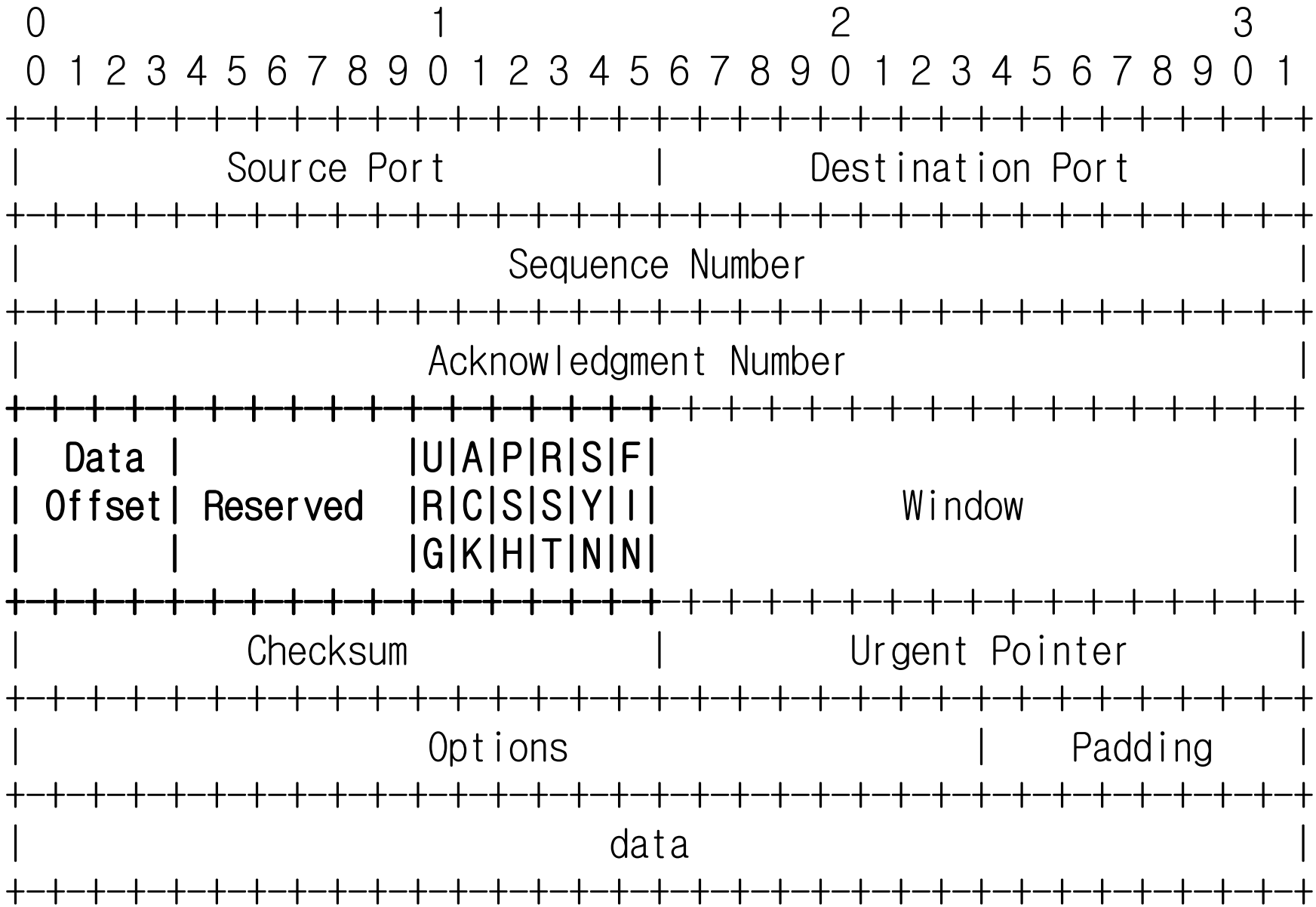
**출발지, 도착지의 포트 번호를 담고 있습니다.**



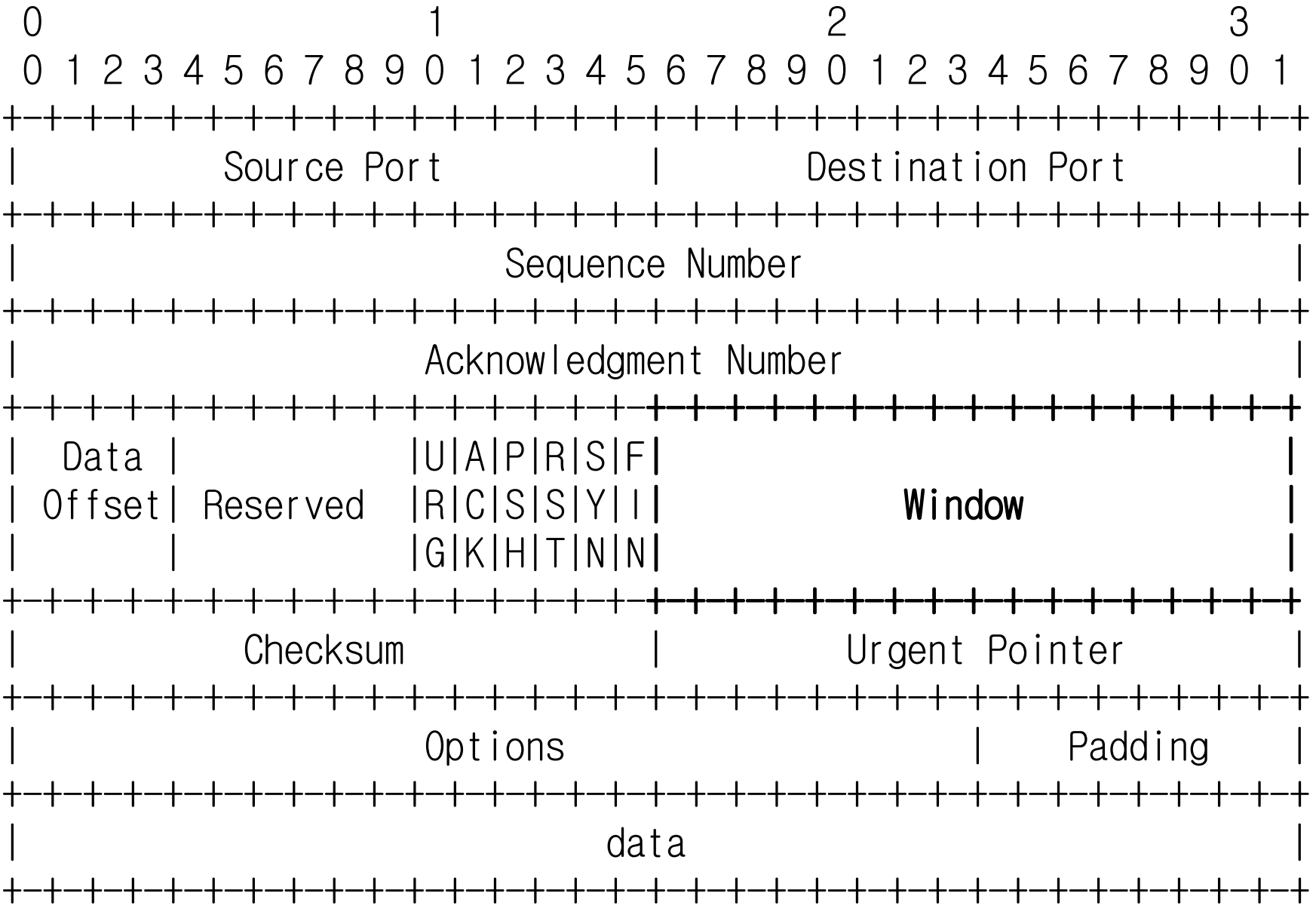
**패킷 손실을 확인하기 위한 번호입니다.**



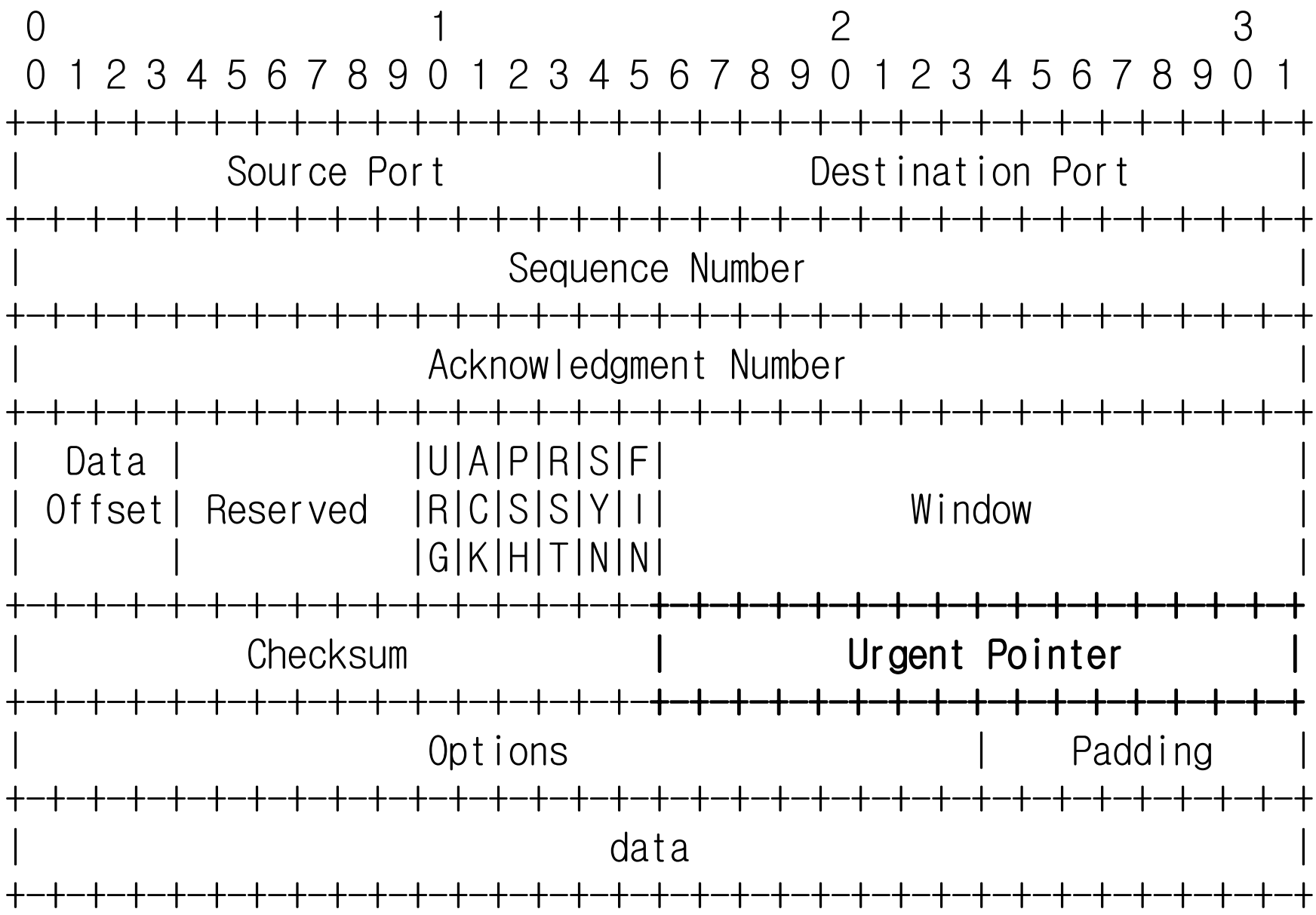
옵션들입니다. Data offset은 역시 헤더의 길이이고 32bit단위입니다.



**창문의 크기 입니다. 수용 가능한 패킷의 크기를 의미합니다(최대 ushort max)**
**컴퓨터의 성능이 구려서 버퍼 크기가 줄면 값이 줄어듦니다.**



긴급 포인터 : 현재의 seq + urgent pointer 를 하면 긴급 데이터 처리를 해야 할 패킷의 seq가 나온다.

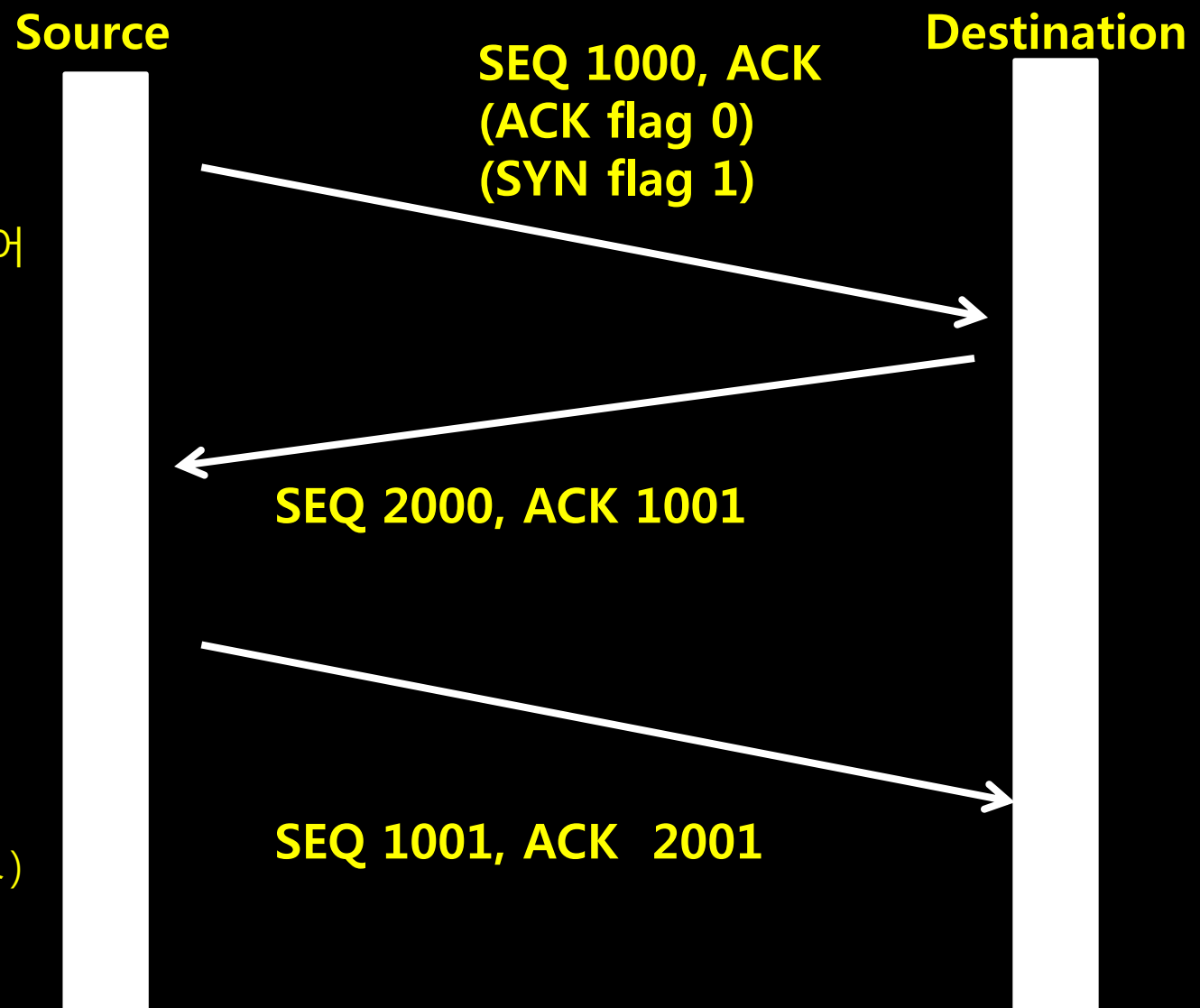


# 최초 연결(Sync, connect)

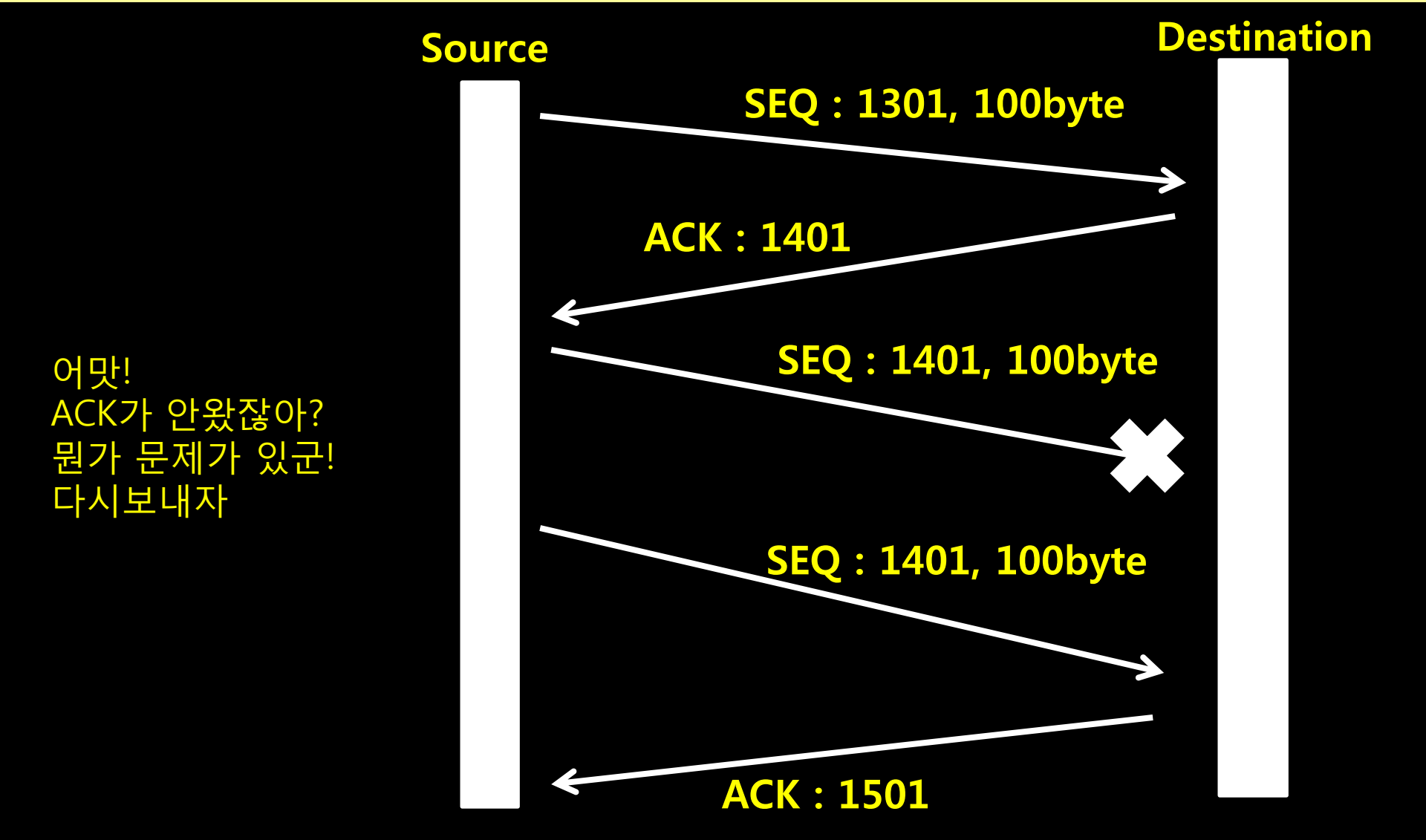
1. 안녕
2. 나도 안녕
3. 너의 안녕 잘 들었어
   이제 우린 친구야!

혹은

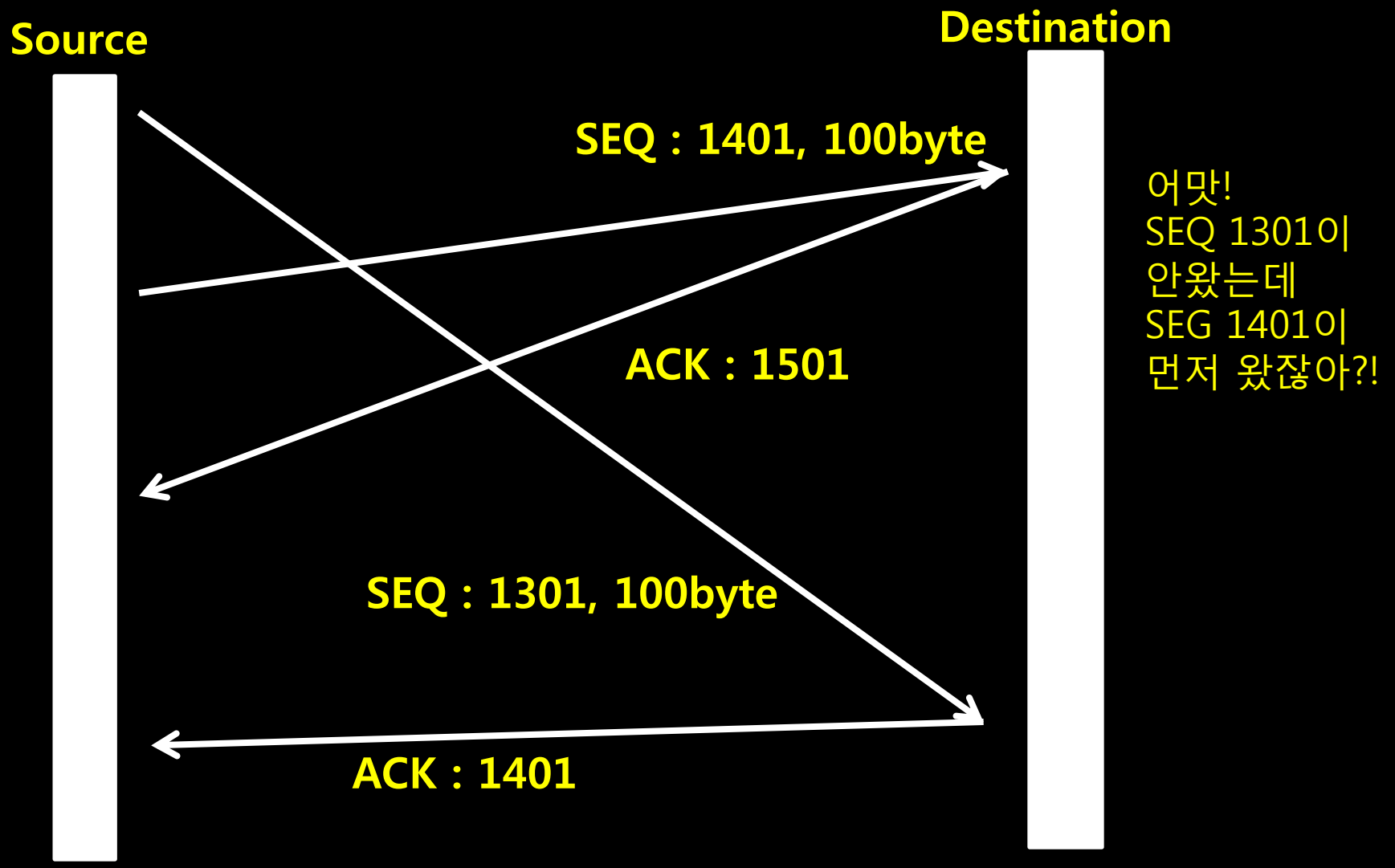
1. 안녕
2. …
3. 다시 안녕
4. 나도 안녕
   (지금들음 ㅈㅅ)
5. …
6. 다시 나도 안녕
7. 안녕(지금들음 ㅈㅅ)



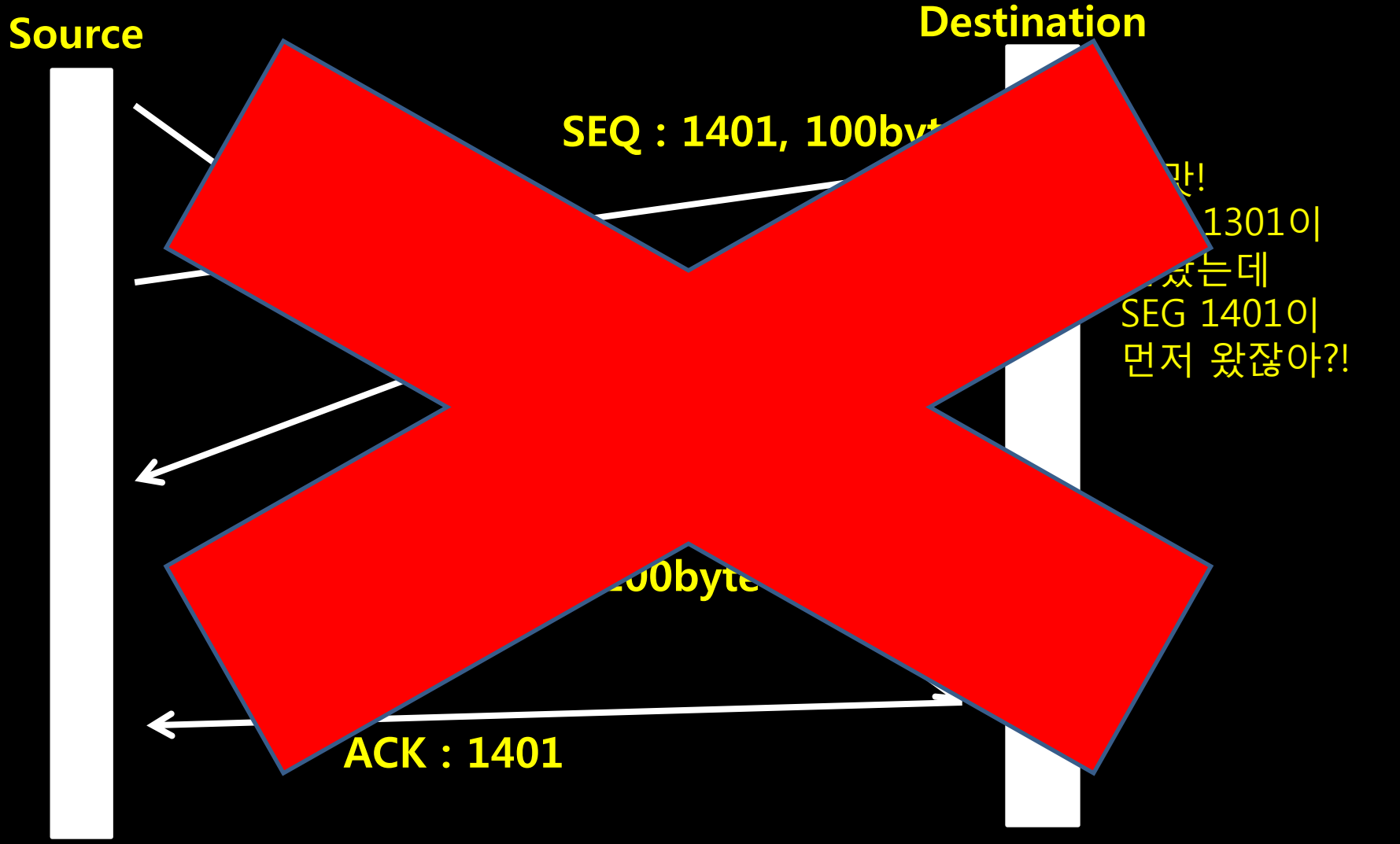
# 데이터 전송(Seq, Ack)



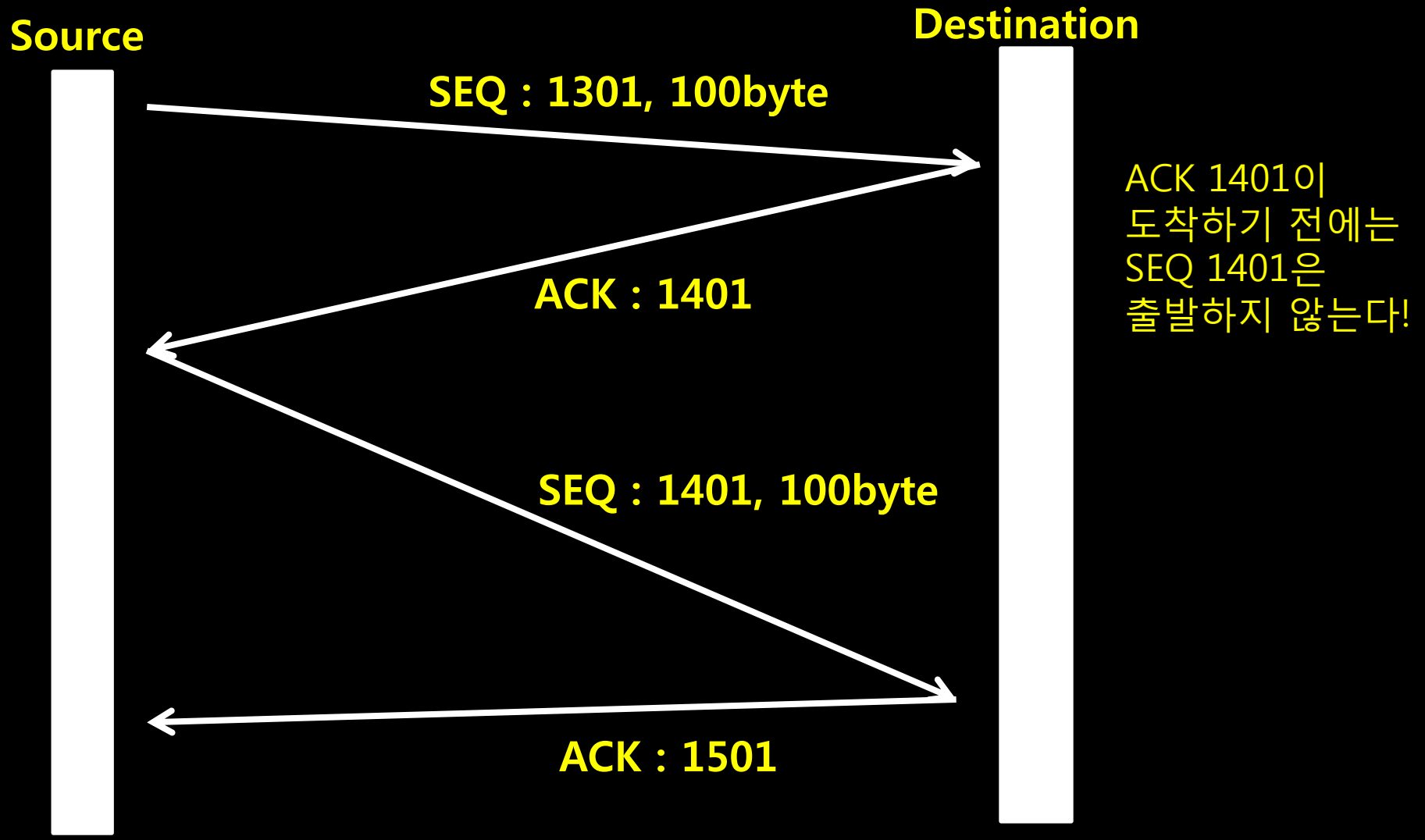
# 데이터 전송(Nagle)


Source
Destination
SEQ : 1401, 100byte
ACK : 1501
SEQ : 1301, 100byte
ACK : 1401
어맛!
SEQ 1301이
안왔는데
SEG 1401이
먼저 왔잖아?!

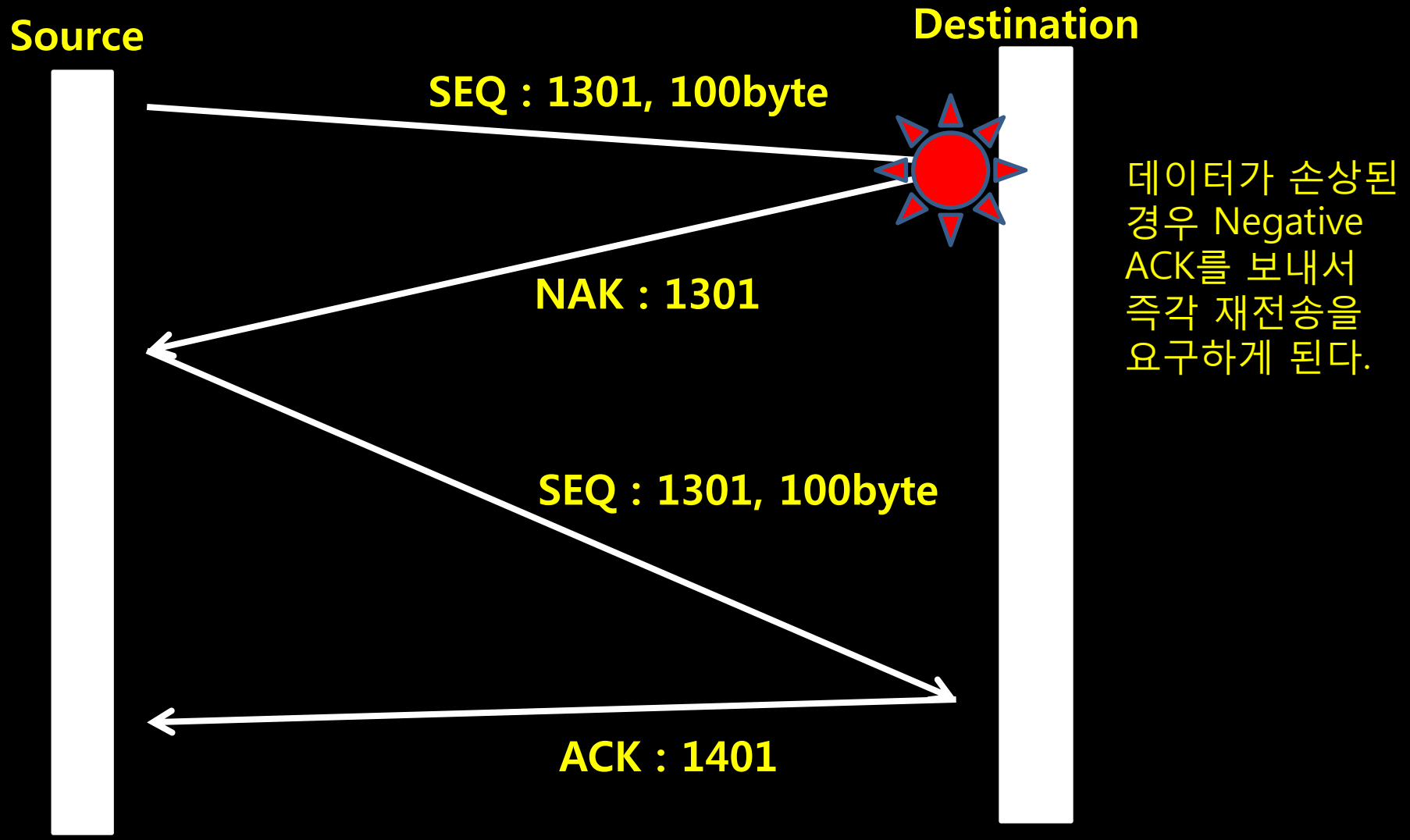
# 데이터 전송(Nagle)



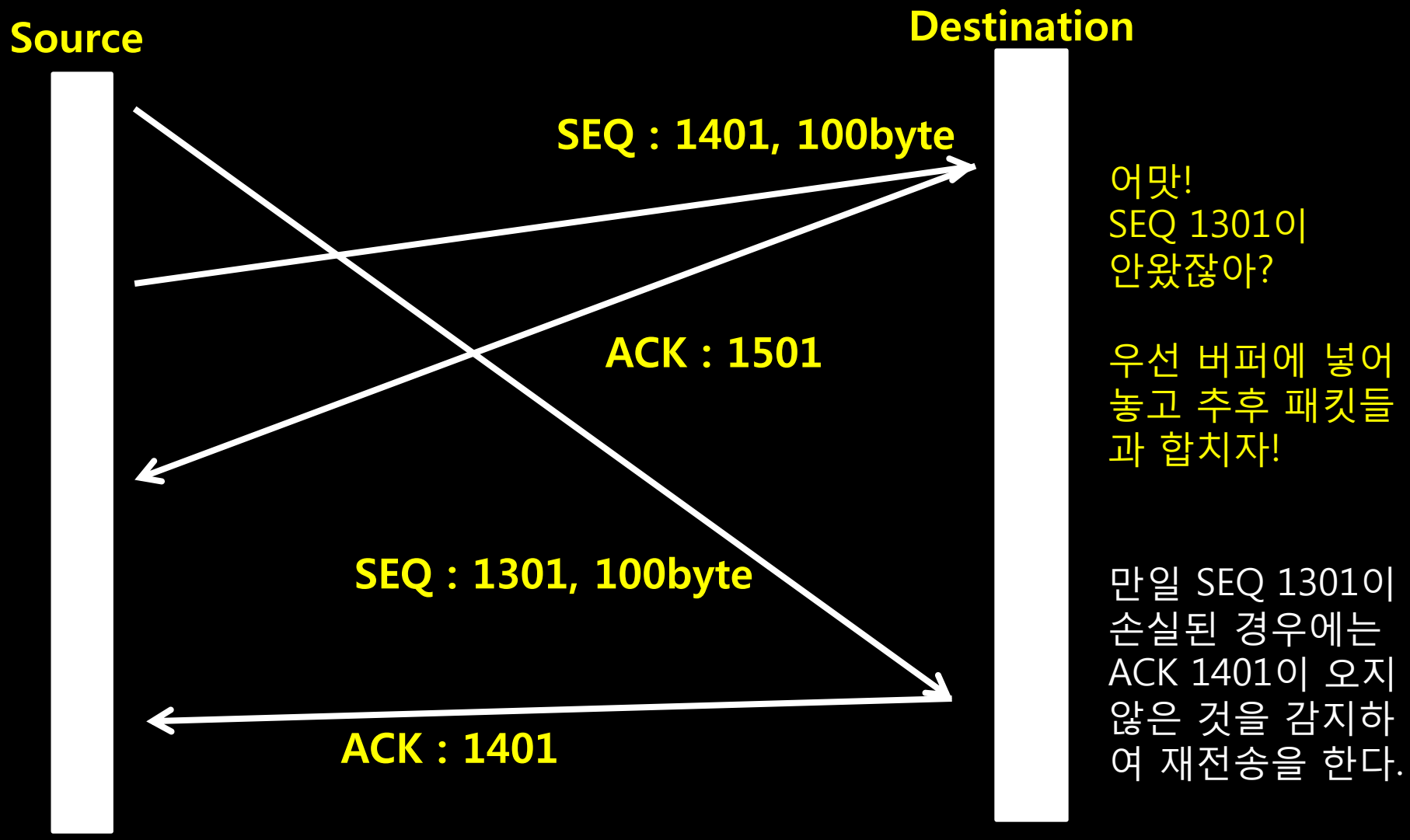
# 데이터 전송(Nagle)


Source
Destination
SEQ : 1301, 100byte
ACK : 1401
SEQ : 1401, 100byte
ACK : 1501
ACK 1401이
도착하기 전에는
SEQ 1401은
출발하지 않는다!

# 데이터 전송(Nagle)


Source
Destination
SEQ : 1301, 100byte
NAK : 1301
SEQ : 1301, 100byte
ACK : 1401
데이터가 손상된 경우 Negative ACK를 보내서 즉각 재전송을 요구하게 된다.

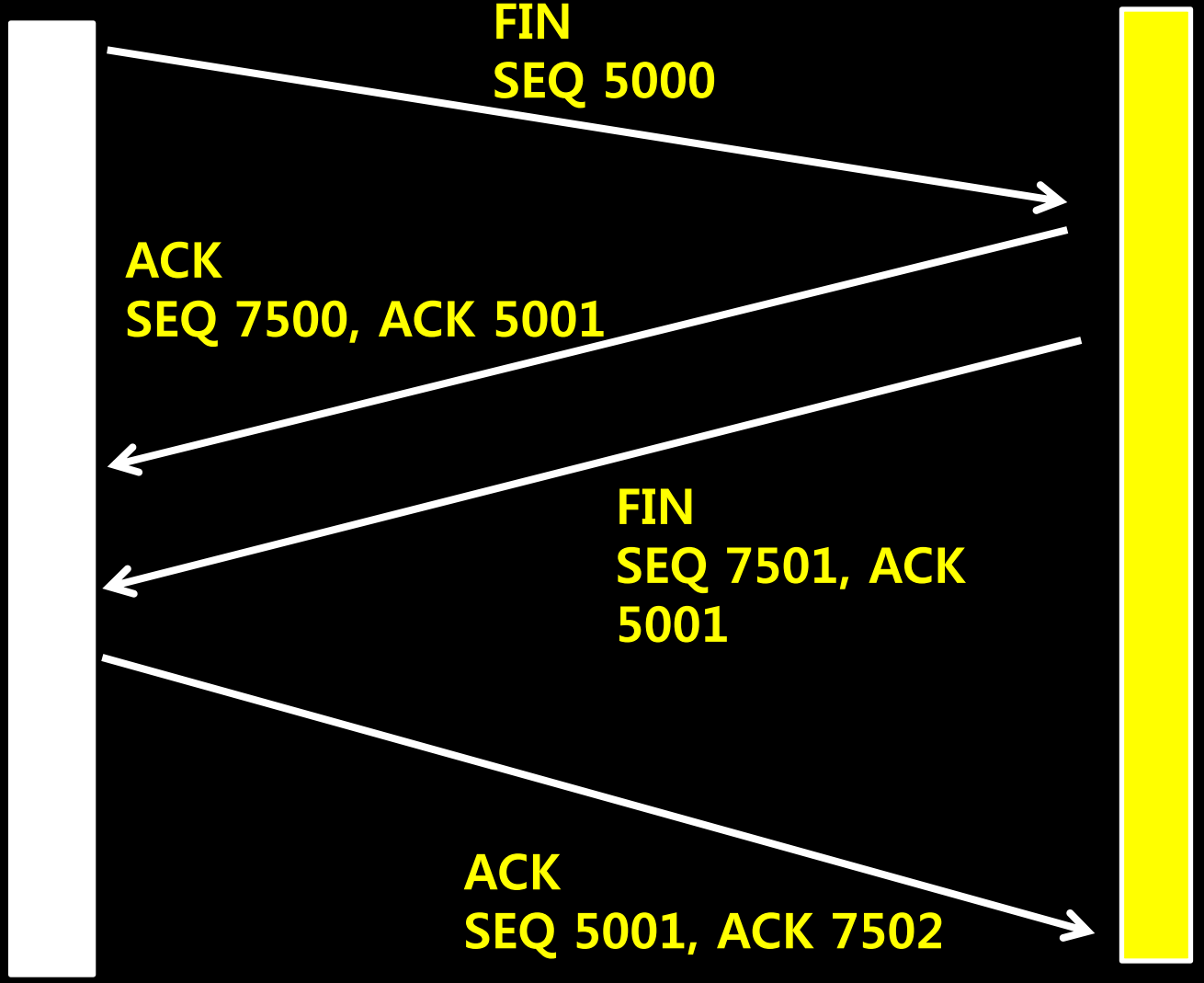
# 데이터 전송(Nagle off)



# 연결 끊기(Fin, close)

1. 전화 끊을게~
2. 응응 알았어 잠시만
3. 오키
   이제 끊어도돼~
   준비완료^^
4. 진짜로 끊는다~
5. (말한뒤 진짜로 끊음)
6. (끊는다는말 듣고
   끊음 - 소심함)

만일 노랑이가 끊는다는 말을 듣지 못하였을 경우 일정 시간 동안 계속 기다린다



리눅스 소켓 함수를 볼까요

int listen(int sockfd, int backlog);

The backlog argument defines the maximum length to which the queue of pending connections for sockfd may grow.

자 백로그가 뭘까요 **요청 처리 중에 다른 요청이 들어왔을 때 저장해 놓는 버퍼**

**만일 클라이언트가 계속 SYN 만 보내고 ACK를 보내지 않는다면??**

**백로그가 모두 차서 다른 클라이언트의 접속 차단(Sync flooding)**

우리 조금 나쁜 방향으로 생각해 봅시다

인터넷 프로토콜이 패킷을 알아서 쪼개고 합쳐주져??
근데 1~100번 패킷 중 2번 패킷 빼고 다 도착했다면??

**1번은 처리되었고 2번이 올 때 까지 3번부터 100번까지를 버퍼에 저장해 놓고 기다려야 하죠**

이처럼 IP에서 패킷을 조각내고 다시 합치는 과정에
비정상적인 신호를 보내서 OS가 뻗어버리게 하는 공격패턴을

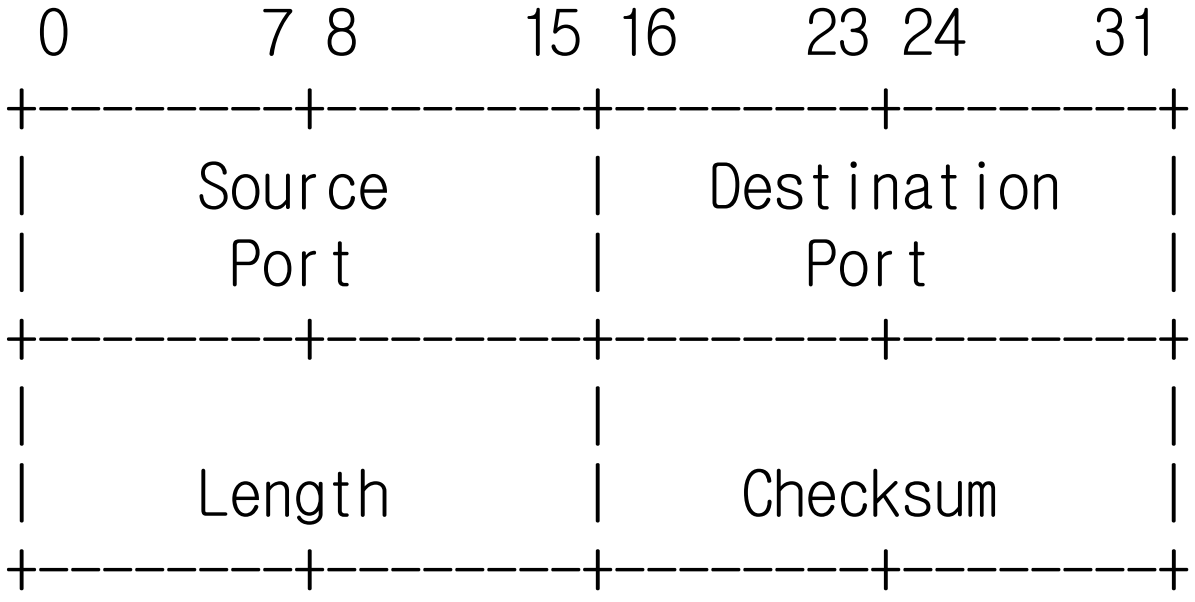
## Teardrop attack 이라고 합니다.

**주로 MS 윈도나 리눅스 2.0.32와 2.1.63 이전 버전의 OS에 영향을 줍니다.**
**너무 낮은 버전의 커널을 사용하지 맙시다.**

**정말 눈물나도록 아픈 공격이라 이름이 저렇게 지어졌대요...**
**변종도 정말 엄청 많습니다.**

UDP : 정말 단순하죠
TCP에서 연결 유지, 패킷 손실 감지, 패킷 순서 검증 등의 기능을 뺀 다이어트 프로토콜이라고 생각하셔도 좋습니다.

## TCP 포트와 UDP 포트는 공유 가능합니다!



**UDP의 가장 중요한 특징**

**255.255.255.255:port 로 UDP패킷을 보내면**
**로컬 네트워크의 모든 컴퓨터가 패킷을 수신한다.**

# UDP를 이용한 loopback 공격

에코 서비스 아시죠? 받은 내용을 source-destination 바꿔서 똑같이 보내주는 서비스...

만일 내가 나한테 echo를 한다면...
(source, destination IP 모두 나, source, destination port 모두 7)

비슷한 서비스로

19번 포트의 CHARGEN(찰진) 이 있습니다.

# Dueling Carls

여기서 배운 내용이 바로

# OSI layer 4

그렇다면 L4 스위치란? TCP, UDP프로토콜 내용을 다루는 스위치 – 포트를 다룬다.

이 레벨에서부터 “로드 밸런싱” 이라는 것이 가능해짐

예를 들어 sparcs.org:80 는 ara.sparcs.org로 보내고
sparcs.org:21은 juk.sparcs.org로 보내고
sparcs.org:22는 bit.sparcs.org로 보내는 일이 가능!

그렇다면 L7 스위치란? L7은 프로토콜 최상위에 있는 것으로 어플리케이션에서 직접 사용하는 프로토콜을 읽고 처리할 수 있습니다.

값도 열라게 비싸고 하는일도 열라게 많습니다

얘는 패킷 내용을 분석해요.
패킷 내부에 욕이 들어있으면 필터링도 되고

ara.kaist.ac.kr/images로 시작하는 주소를
이미지 파일만 모아놓는 서버로
ara.kaist.ac.kr/boards로 시작하는 주소를
보드 내용을 처리하는 서버로 보낼 수 있습니다.

수고하셨습니다.

잠깐 쉬고 실용적인 내용으로 넘어가도록 하겠습니다.

# PING

다른 컴퓨터와의 연결 상태를 확인하기 위한 유틸리티.
네트워크 지연 시간을 확인할 때에도 쓰인다.

**내가 ping을 다른 컴퓨터에 보내면...**
**그 컴퓨터는 ping 패킷을 “반사” 해 준다...**

# 사용법

ping khl.sparcs.net

ping khl.sparcs.net (-옵션 값)

-s 패킷 크기
-t TTL 값
-b 브로드캐스트

# 원리

TCP가 데이터를 전송하기 위한 프로토콜이라면
같은 OSI level 4에 있는 **ICMP라는 프로토콜은**
인터넷 장비끼리 통신하기 위한 프로토콜이다!

**ICMP_ECHO_REQUEST 를 보내면**
**ICMP_ECHO_REPLY를 상대방이 무조건 보내도록 약속한다.**

# 방어

핑을 통해서 서버에 공격을 할 수도 있고(ping of death)
서버의 on/off 유무를 파악할 수도 있습니다.

*ping of death란
Ping의 크기는 65535 로 제한되어 있는데
이보다 큰 크기의 패킷을 보내서 운영체제를 죽게 하는 공격법입니다.
Windows 98, NT등에서 나타납니다.
큰 패킷의 ping은 네트워크를 지나면서 잘게 분할되어 패킷이 엄청 많아집니다.

/proc/sys/net/ipv4
에서 네트워크 설정을 바꿀 수 있습니다.

**$echo 0 > /proc/sys/net/ipv4/icmp_echo_ignore_all**
**$echo 1 > /proc/sys/net/ipv4/icmp_echo_ignore_broadcast**

# Tracert (패키지 이름 : traceroute)

내가 보낸 패킷이 어느 장비들을 거쳐서 목적지에 도달하는지 알고 싶을 때

네트워크의 어느 부분에서 오류가 발생하는지를 알고 싶을 때

**tracert khl.sparcs.net**

# 원리

장비가 없어서 연결을 못하는 경우
: ICMP_TIME_EXCEEDED

UDP 포트가 닫혀서 연결을 못하는 경우
: ICMP_PORT_UNREACHABLE

**절대로 사용하지 않을 것만 같은 포트번호로 패킷을 보내서 (약 3만번 이상) 돌아오는 ICMP_PORT_UNREACHABLE을 확인한다.**

**TTL을 차근차근 증가시켜보면서 거리 1에 있는 애, 거리 2에 있는 애 이런식으로 탐색한다.**

# 주의사항

Traceroute 가 탐색하는 경로는 일정하지 않다.

예를 들어서
가 – 라 까지 가는 경로가
가 – 나 – 다 – 라
가 – A – B – 라
두 가지가 있을 때

Tracert는 가 – A – 다 – 라 의 경로를 보여줄 수 있다
(실제로는 존재하지 않는 경로)

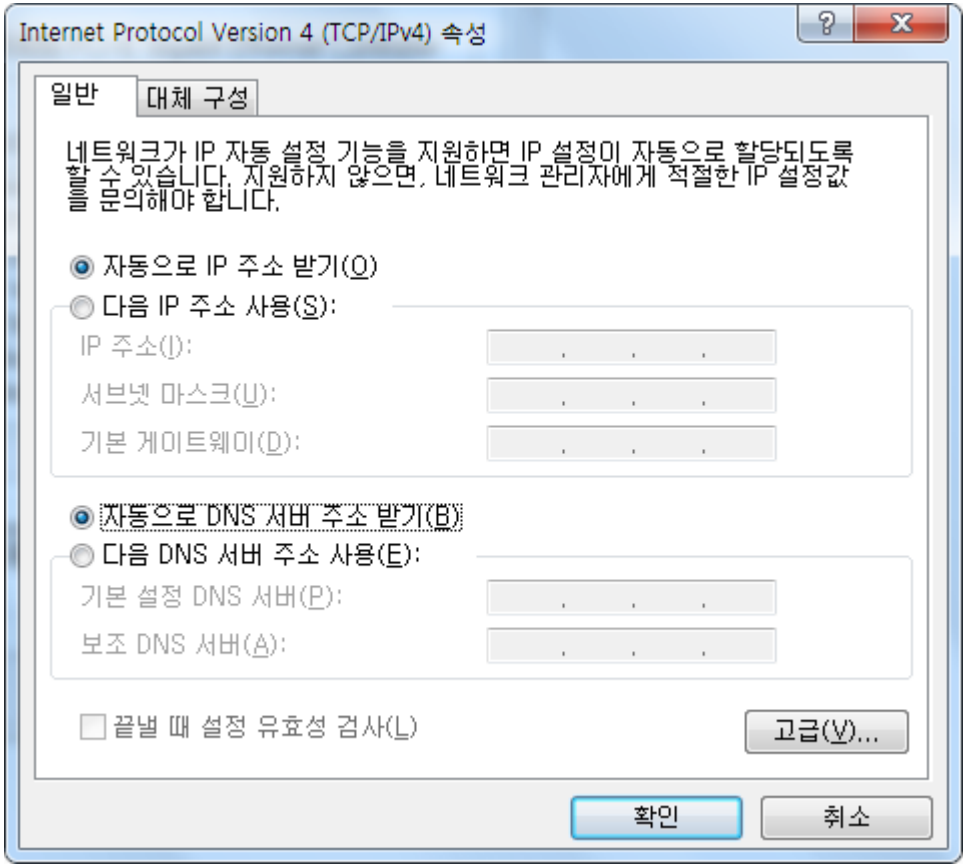
# DHCP

동적 호스트 설정 통신 규약(Dynamic Host Configuration Protocol)

이를 위한 프로토콜입니다.

"나 주소 필요함,
어디 누구 없음?"

"그래 너 이거 써라"
"예 알겠스빈니다"




**위 대화는 UDP broadcast 를 통하여 이루어진다**

# 설정법

내가 주소가 필요한 경우 :

sudo dhclient (장비 이름 : eth0)

내가 주소를 주는 경우 :
dhcpcd 패키지를 설치

**주의 : 한 서브넷(같은 게이트웨이를 공유하는)에는 오직 하나의 DHCP서버만 있어야 한다.**

**이 설정은 ‘스위치’ 에서 차단할 수 있다.**
**(DHCP spoofing 방지)**

# netstat

현재 네트워크 상태를 보여준다.

-n : 주소를 숫자로 보여준다(khl.sparcs.net 대신 143.248.....)
-p : 각 연결을 들고 있는 프로그램을 보여준다.
-a, -t –u : 각각 전부, TCP만, UDP만 보여준다

# ifconfig

네트워크 설정 확인 및 변경

**ifconfig : 네트워크 설정 보기**

**ifconfig eth0 192.168.0.123 netmask 255.255.255.255 broadcast 192.168.0.1**

ifconfig eth0 up : 켜기

ifconfig eth0 down : 끄기

# 이 설정은 재부팅 되면 없어집니다!

데비안 기준
/etc/network/interfaces 에 있는 세팅 파일을 수정한 뒤

수도 권한으로
/etc/init.d/networking restart 를 하면 됩니다.

이 설정은 부팅 시 마다 불러옵니다.

ip alias 를 설정하려면? (한 컴퓨터에 여러 아이피 할당)

/etc/network/interfaces 파일에서

eth0:0 의 인터페이스를 설정하고

Address와 netmask만 설정해 주면 된다.

# 방화벽

컴퓨터로 들어오는 패킷을 필터링 해 주는 녀석

패킷을 허용, 차단할 수 있다.
차단 기준은 프로코톨, 포트, 주소 등을 받을 수 있다.

iptables , iptables-restore , iptables-save

Iptables로 설정을 하고
Iptables-save로 설정파일을 남기고
Iptables-restore로 설정파일을 불러온다.

iptables -A INPUT -p udp --dport 123 -j ACCEPT
iptables -A OUTPUT -p tcp --dport 80 -j DROP

의미 : A : append, chain의 맨 뒤에다가 추가(우선순위 낮음)
INPUT : 들어오는 UDP : udp연결 중 dport : 목적지 포트가 123인 패킷을
ACCEPT : 허락한다.

-A대신에 쓰일 수 있는 것들
-I : insert – 맨 앞에 삽입한다(우선순위 높음)
-R : replace – 해당하는 주소에 대한 내용을 덮어쓴다.

여러 조건에 해당할 경우에는 우선순위 높은 것을 따른다.

예를 들어 앞의 필터에서는 차단당하고 뒤에 필터에서 허락받으면 **차단**
앞의 필터에서 허용되었으면 뒤에 필터에서 차단당해도 **허락**