School of Electrical and Electronic of Enginnering
eee 4474-01 : Experiments on Communication Networks
2조 2016142096 **조윤신**, 2017142043 **김재민**

Pre Report Week2

wireshark(2) : TCP UDP IP protocols

1 TCP protocol

TCP[1]는 OSI 7계층 중 Transport Layer에서 사용되는 프로토콜로, 통신과정에서 정보를 안정적으로, 순서대로 에러없이 교환할것을 보장해주는 기능을 수행하는 프로토콜이다.

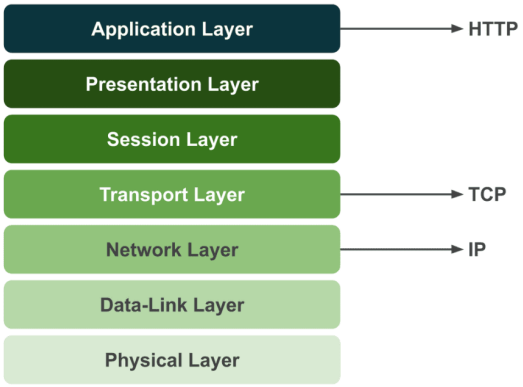


Figure 1: OSI 7 Layer

7계층 구조를 이룸으로서, 네트워크라는 수많은 기술의 집약체에서 각각 계층간의 역할 분담을 통해서 각 작업별로 신경써야 되는 범주 부분을 한정시켜준다. 예를들어 HTTP 라는 application 요소를 다룰때, 그때의 DNS나 packet의 처리 등등의 문제는 별도로 다룰수 있는 점을 들 수 있다.

Transport Layer로서 데이터의 전송하는 방법은 전송하고자 하는 Data를 Packet으로 나누고 이를 여러 Router를 거쳐 Destination 까지 전송하는 일련의 과정이고, TCP protocol은 이러한 Packet의 전송에서 완전성을 보장하는 역할을 수행한다. TCP packet이 어떻게 완전성을 보장하는지 TCP segment의 Header를 통해 packet의 구성을 확인해 보자.

1.a TCP packet의 구성

HTTP, TCP, IP와 같은 protocol들은 각 layer에서 고유의 역할을 수행하고, 각각 다루는 data들을 layer의 기능에 따라 data에 자신의 **Header**를 붙이는 방법을 통해서 정보를 표현한다.

TCP는 packet의 전송에 있어 신뢰성과, 흐름제어, 혼잡제어등의 역할을 수행하는 protocol이기 때문에, 각각의 Header에 기능을 수행하기 위한 값들이 포함되어 있다.

TCP Header

Offsets	Octet	0								1								2								3							
Octet	Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0	0	Source port																Destination port															
4	32	Sequence number																															
8	64	Acknowledgment number (if ACK set)																															
12	96	Data offset				Reserved 0 0 0				NS	CWR	ECE	URG	ACK	PSH	RST	SYN	FIN	Window Size														
16	128	Checksum																	Urgent pointer (if URG set)														
20 ...	160 ...	Options (if *data offset* > 5. Padded at the end with "0" bytes if necessary.) ...																															

Figure 2: TCP Header

TCP는 기본적으로 별도로 나누어진 20 Byte, 160 bit의 Binary를 Header로 사용한다. 만일 추가의 option field 이용시에 40 Byte, 최대 60 Byte까지 지원이 가능하다.

[1] Transmission Control Protocol

Source port, Destination port

이 필드에서 전송할 segment의 출발지와 목적지를 나타낸다. 각각 16 bits로 IP address 와 포트번호 를 통해서 표현한다. Transport Layer에 포함되는 TCP 의 경우 IP 주소는 한 계층 아래인 Network Layer 의 IP header에 담기기 때문에 TCP header에는 IP 주소를 나타내는 필드가 없고 포트번호만을 나타내는 필드만 존재한다.

Sequence Number

앞서 TCP의 base가 전송하고자 하는 data를 각각의 packet으로 나누어 전송하는것이고, sequence number를 통해서 쪼개진 Segemnent의 순서대로 data를 재조립한다. figure 2에서 확인할 수 있듯이 총 32bits를 할당받고, 이는 4,294,657,296 의 sequence를 담을 수 있음을 의미한다. 데이터를 최초로 전송할때 random한 번호로 initialize 시키며, 이후 보내는 데이터의 1 bytes 당 sequence number를 1씩 증가 시킨다.

Acknowledgement Number

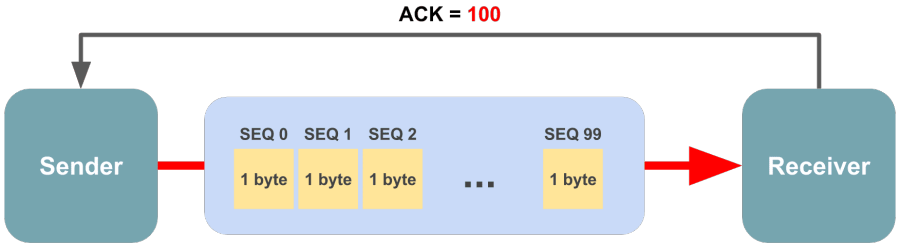


Figure 3: ACK flow diagram

ACK은 데이터를 받은 수신자가 예상하는 다음 Sequence Number를 의미하고 크기는 Sequence Numer와 동일한 32 bits 이다. 처음 연결을 설정하는 handshake 과정[2] 에서는 “상대방이 보낸 Sequence Number + 1” 로, 데이터를 주고받는 과정에서는 “상대방이 보낸 Sequence Number + 자신이 받은 데이터의 Bytes”로 ACK을 만들어낸다. 즉 ACK 은 다음에 보내주어야할 데이터의 시작점을 의미한다.

Data Offset

Data offset field 는 전체 segment 중에서 Header가 아닌 전송하고자 하는 data가 어디서부터 시작하는지 표기해준다. 이는 TCP의 옵션 field 부분의 길이가 사용여부에 따라 가변적이기 때문이다. 32 비트 워드를 사용하며, 이때 32 bit 체계에서는 1 Word = 4 bytes 이다. 즉 Data Offset field의 value에 4를 곱해주면 segment에서 header를 제외한 실제 데이터의 시작 위치를 확인할 수 있다.

Reserved (3 bits)

미래를 위해 예약된 필드로, 모두 0 3bit를 채워준다.

Flags (NS FIN)

9개의 비트 플래그이다. 이 플래그들은 현재 세그먼트의 속성을 나타낸다. 기존에는 6개의 플래그만을 사용했지만, 혼잡 제어 기능의 향상을 위해 Reserved 필드를 사용하여 NS, CWR, ECE flag가 추가되었다.

우선 기존에 사용되는 6가지 flag의 의미를 table로 정리해 보았다. 추가된 3개의 flag는 네트워크의 ECN[3]를 위해 사용되는 **NS, CWR, ECE** flag 이다. ECN은 앞서 ’000’로 남겨놓은 Reserved (3 bits) field의 3 bits를 이용하고, 기존의 Time-Out을 이용한 네트워크 혼잡인지 방법의 비효율성을 개선하기 위해 혼잡상태를 직접 알려주는 방법이다.

ECN은 **CWR, ECE, ECT, CE** flag들을 이용해서 상대방에서 네트워크의 혼잡도의 정보를 전달할 수 있는데 이중 **CWR, ECE** 는 Transport Layer인 TCP header에, **ECT, CE** flag는 Network Layer인 IP header에 존재한다.

[2]이는 Section 1.b 의 SYN segment 와 SYNACK의 Handshake 에서 다루고자 한다.
[3]Explict Congestion Notification, 명시적 혼잡통보

필드	의미
URG	Urgent Pointer 필드의 값이 채워져 있음을 알려주는 flag. 포인터가 가르키는 데이터가 우선적으로 처리되나, 50년대 고안된 TCP 성격상 요즘에는 사용되지 않는다.
ACK	Acknowledgement 필드의 값이 채워져 있음을 알려주는 flag 이 flag가 0 이면 ACK 값이 무시된다.
PSH	Push flag. 수신측에 이 데이터를 최대한 빨리 application에 전달하라는 것을 알려주는 flag. flag가 0이면 수신측은 자신의 buffer가 채워질때 까지 기다린다. 즉 flag가 1이라는 것은 이 segment이후로 연결된 segment가 없다는것을 의미한다.
RST	Reset flag.
SYN	Synchronize flag. 상대방과 연결을 생성할때 sequence number와 동기화를 위한 segment임을 의미한다.
FIN	Finish flag

Table 1: The meaning of 6 classic flags in Flags field in TCP's header

필드	의미
NS	ECN 에서 사용하는 CWR, ECE 필드가 실수나 악의적으로 은폐되는것을 대비해 추가된 필드
ECE	ECN Echo flag. 위 필드가 0 이면서, SYN flag가 1일때 ECN을 사용한다고 알리는 의미. 이때 SYN flag가 0이라는것은 네트워크가 혼잡하기 때문에 segment window 크기를 줄여달라는 의미
CWR	이미 ECE로 부터 flag를 받아서 segment의 크기를 줄였다는것을 확인하는 flag

Table 2: The meaning of 3 flags that used fof ECN in Flags field in TCP's header

Window Size

window size field의 값은 한번에 전송할 수 있는 data의 크기의 값을 가진다. 2^{16} = 65535 만큼의 값을 표현할 수 있고, 이는 윈도우의 최대 크기가 64kb 라는의미이다. 현대의 통신환경과는 맞지 않는 단위이므로, 비트를 왼쪽으로 shift 하는 방법으로 scale을 키워주는 **WSCALE** 방법들을 이용한다. **WSCALE**을 통해서 얼마나 shift 할지의 값은 option field에 존재한다.

Check Sum

데이터의 송신과정에서 발생할 수 있는 오류를 검출하는 역할을 수행한다. Wraparound를 이용해 carry를 더해준 값에서 1의 Complement를 취해 Checksum 값을 만들어 준다. 이때의 Wraparound의 값에 송신측에서 보내준 wraparound의 1의 complement인 cheksum 을 더해줘서 모든 비트가 1이라면 정상적인 것이고, 하나라도 0이 검출될 시에 이는 송신과정에서 오류가 발생했음을 확인할 수 있다.

Urgent Pointer

URG flag가 1의값을 가진다면, 수신측은 Urgent Pointer의 adr에 있는 데이터를 우선적으로 처리한다.

Options

TCP의 기능을 확장하는데 사용하는 필드이다. 앞서 언급된 Window Size의 scale을 조절하는 WSCALE , Selective Repeat등을 포함하는 SACK 등이 있으며, 이는 옵션의 포함여부에 따라서 Option의 필드의 크기가 가변적이게 된다. Options field의 길이가 변함에따라 header의 길이 또한 변하게 되고, 이는 수신측에서 어디서부터가 header이고 어디서부터가 data인지 확인하기 위해 offset field를 통해서 이를 전달한다.

즉 32비트 워드를 사용하는 Data Offset field의 값을 통해서 이 값의 4를 곱하는 지점에서 실제 헤더를 제외한 데이터가 시작하는 것을 알리는데, Data Offset field 의 크기가 4 bits 이므로, 최대 $4 \times 2^4 = 60$ bytes 까지 표현할 수 있다. 이때 options를 제외한 필수 Header 가 160 bits 즉 20 bytes = 4×5 (offset's vlaue) 를 차지하므로, TCP header의 size가 20 bytes ~ 60 bytes 사이일때 즉, data offset > 5 일때 20 bytes 에서 초과한 만큼 Options field에서 0 을 채워주어, 수신측에서 header의 size를 확인할 수 있게 해준다.

1.b SYN segment와 SYNACK segment의 역할

Terminology : Protocol Data Unit

프로토콜 데이터 단위[4]는 OSI 7 layer 구조로 표현할 수 있는 데이터통신에서 상위 계층에서 전달한 데이터에 붙이는 제어정보를의미한다.

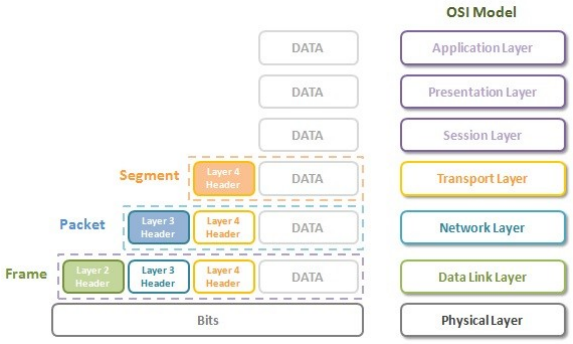


Figure 4: OSI 7 Layer

물리적으로 Bits로 표현되는 데이터에는 figure 에서 확인할 수 있는것과 같이 전송하고자 하는 데이터에 각 Layer별로 기능을 수행하는 기능의 제어정보를 각각의 terminology를 사용해 표현되는 header에 추가해 줌으로서, Layer에 있는 Protocol이 기능을 수행할 수 있게 해준다. 앞서 우리는 TCP의 기능을 확인해 보기 위해서 Network Layer의 제어정보인 Packet에 해당하는 TCP Header를 리뷰해 보았다.

SYN segment 와 SYNACK segment는 Transport Layer에서 TCP의 제어정보를 담은 segment이고, TCP header의 flag field에서 확인한 SYN 과 ACK field와 연관이 있는 unit이다.

다루게될 UDP와 다르게 TCP는 연결지향[5] 동작을 하는 protocol로서 연속적인 데이터 전송의 신뢰성을 보장해주는 기능을 수행하기위해 해당 segment들을 필요로한다. TCP는 패킷 전송 방식을 사용하기 때문에 보내려고 하는 데이터를 여러개의 패킷으로 쪼개서 보낸다.

이때 쪼개진 Packet들을 TCP를 이용해 통신하는 각 종단들은 어떤 TCP 옵션들을 사용할 지, 패킷의 순서 번호 동기화와 같이 통신에 필요한 몇 가지 segment들을 주고받는데, SYN segment와 SYNACK segment가 해당되고 이러한 최초 연결시의 동기화 과정을 **3-way handsake**라고 한다.

TCP Connection : 3-way handsake

TCP가 통신을 시작할 때 거치는 과정을 3 Way Handshake, 통신을 마칠 때 거치는 과정을 4 Way Handshake 라고 한다.

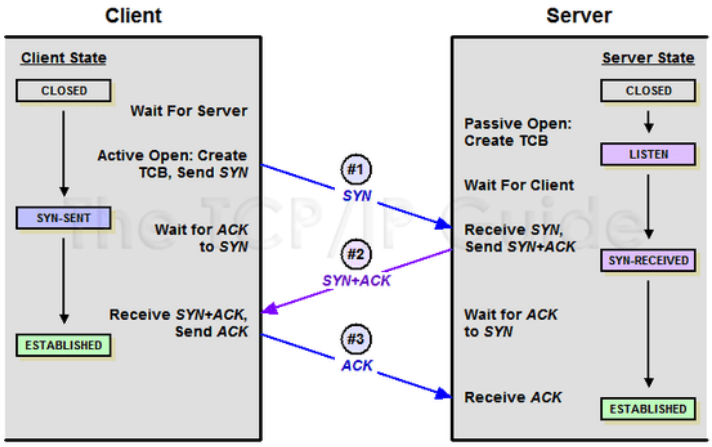
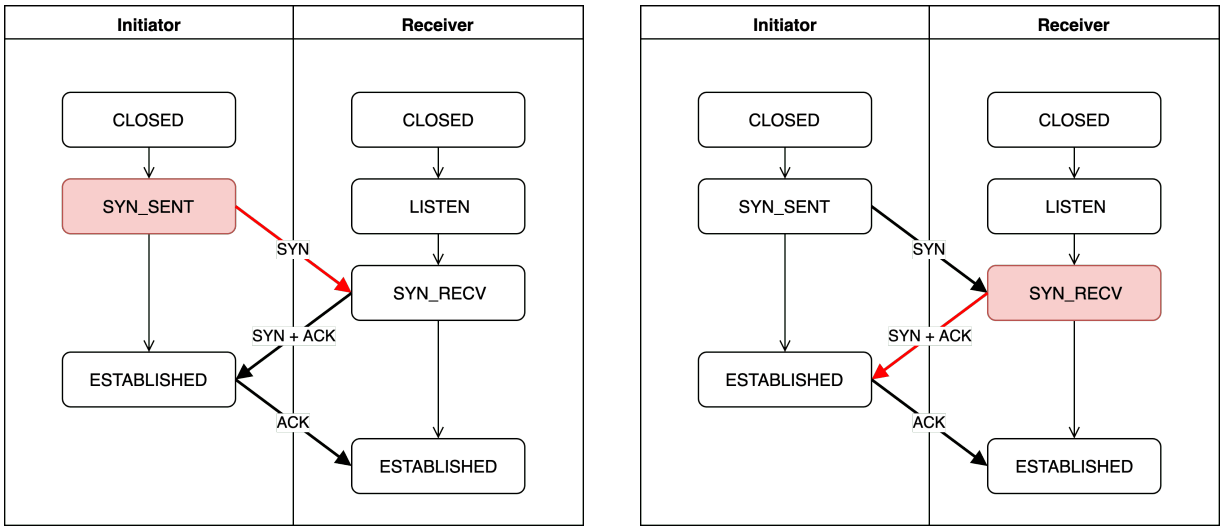


Figure 5: 3-way handshake diagram

3way Handshake에서 3 Way라는 말 그대로 총 3번의 통신 과정을 거친다. 이 과정을 거치면서 통신을 하는 양 종단은 내가 누구랑 통신하고 있는지, 내가 받아야할 데이터의 시퀀스 번호가 몇 번인지와 같은 정보를 주고 받으면서 연결 상태를 생성하게 된다.

[4]Protocol Data Unit
[5]Connection Oriented



(a) The diagram of SYN sent using SYN segment

(b) The diagram of SYN Received using SYNSCK segment

Figure 6: 3-way Handshake Process

SYN_sent : SYN segment

요청자가 수신자에게 연결 요청을 하면서 랜덤한 숫자인 시퀀스 번호를 생성해서 SYN segment에 담아 보낸 상태이다. 이제 요청자와 수신자는 이 시퀀스 번호를 사용하여 계속 새로운 값을 만들고 서로 확인하며 연결 상태와 패킷의 순서를 확인하게 된다.

SYN_recv : SYNACK segment

SYN_RECV는 요청자가 보낸 SYN 패킷을 수신자가 제대로 받은 상태를 의미한다.이후 수신자는 제대로 된 시퀀스 번호를 받았다는 확인의 의미인 '승인번호 Acknowledgement' 값을 만들어서 다시 요청자에게 돌려줘야한다. 이때 승인 번호는 처음 요청자가 보낸 시퀀스 번호 + 1이 된다.

1.c window의 의미

TCP의 기능중 전송되는 데이터의 완전성 보장이외에도 혼잡제어 기능을 수행한다. 네트워크의 특성상 어느 경로에서 지연이 발생하는지 파악하기 어렵고, 지연이 반복되면 재전송등 비효율이 발생하고 지연이 중첩될 수 있다. 따라서 네트워크의 혼잡상태를 감지해서, 자연을 회피하기 위해 송신측의 윈도우 크기를 조절함으로서 데이터 전송량을 강제로 줄이는 혼잡제어 기능을 수행한다.

이때 TCP가 인지하는 parametersms 송신 측은 자신의 최종 윈도우 크기를 정할 때 수신 측이 보내준 윈도우 크기인 수신자 **윈도우(RWND)**, 그리고 자신이 네트워크의 상황을 고려해서 정한 윈도우 크기인 **혼잡윈도우(CWND)** 중에서 더 작은 값을 사용한다.

즉 이때 protocol이 설정해주는 송신측이 가지는 혼잡윈도우[6]의 크기를 window라고 한다.

[6]혼잡 윈도우(Congestion Window, CWND)

2 UDP protocol

2.a UDP packet의 의미

UDP User Datagram Protocol이라는 이름에서도 알 수 있듯이 데이터그램 방식을 사용하는 프로토콜이기 때문에 애초에 각각의 패킷 간의 순서가 존재하지 않는 독립적인 패킷을 사용한다.

또한 데이터그램 방식은 패킷의 목적지만 정해져있다면 중간 경로는 어딜 타든 신경쓰지 않기 때문에 종단 간의 연결 설정 또한 하지 않는다.

이에따라 단순히 datagram을 일방적으로 보내기 때문에 protocol의 기능을 담은 header의 field가 tcp와 비교해보았을 때 단순하다.

UDP Header

Offsets	Octet	0								1								2								3							
Octet	Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0	0	Source port																Destination port															
4	32	Length																Checksum															

Figure 7: UDP Header Diagram

Source Port / Destination Port

Invitor 와 reciver 각각의 port 정보.

Total Length

헤더와 데이터를 합한 사용자 데이터그램의 전체 길이의 정보. Packet 이 아닌 Datagram을 전송하는 UDP의 경우 데이터그램의 헤더인 8바이트부터 65507바이트 사이의 값을 가진다.

Checksum

TCP의 checksum field와 동일하다.헤더와 데이터를 모두 포함한 사용자 데이터그램 전체에 대해 오류를 탐지한다.

3 IP protocol

IP 프로토콜은 OSI 참조 모델의 제 3계층인 네트워크 계층에서 정의된 패킷 (또는 IP 데이터그램)을 출발지에서 목적지까지 전달하는 기능을 담당한다. 이를 위해 최선형(Best Effort)서비스를 이용한다. 최선형 서비스는 패킷을 목적지까지 확실하게 전달하는 것을 보장하는 것이 아닌, 전송하는데 최선을 다하는 방식이기 때문에 전송 도중에 패킷이 손상될 수 있고, 패킷들이 순차적으로 도착하지 않을 수가 있다. 따라서 IP 프로토콜 고유의 최선형 서비스 특성의 단점을 극복하기 위해서는 상위 계층의 TCP와 같은 신뢰성 있는 프로토콜의 도움을 받아야 한다.

3.a IP diagram의 구성

Version

IP가 어떤 버전을 사용하는지를 나타낸다. 현재는 IPv4를 사용하고 있다. 하지만 버전 4의 주소 고갈 문제와 보안 이슈로 인해 IPv6가 등장하게 되었다. v6는 패킷 처리에 대한 오버헤드를 줄이기 위해 v4의 헤더를 대폭 간소화하였으며 128비트의 주소 공간을 가지고 있어 무한대에 가까운 주소를 할당할 수 있고 기존의 v4와 달리 주소의 클래스를 나누지 않고 단순히 유니캐스트(unicast, anycastm multicast)의 주소 형태로 나누었다. v4에는 보안 항목이 없어 보완하기 위해 IPSec와 같은 프로토콜을 사용하였는데 v6는 프로토콜 내에 보안 관련 기능을 가지고 있다.

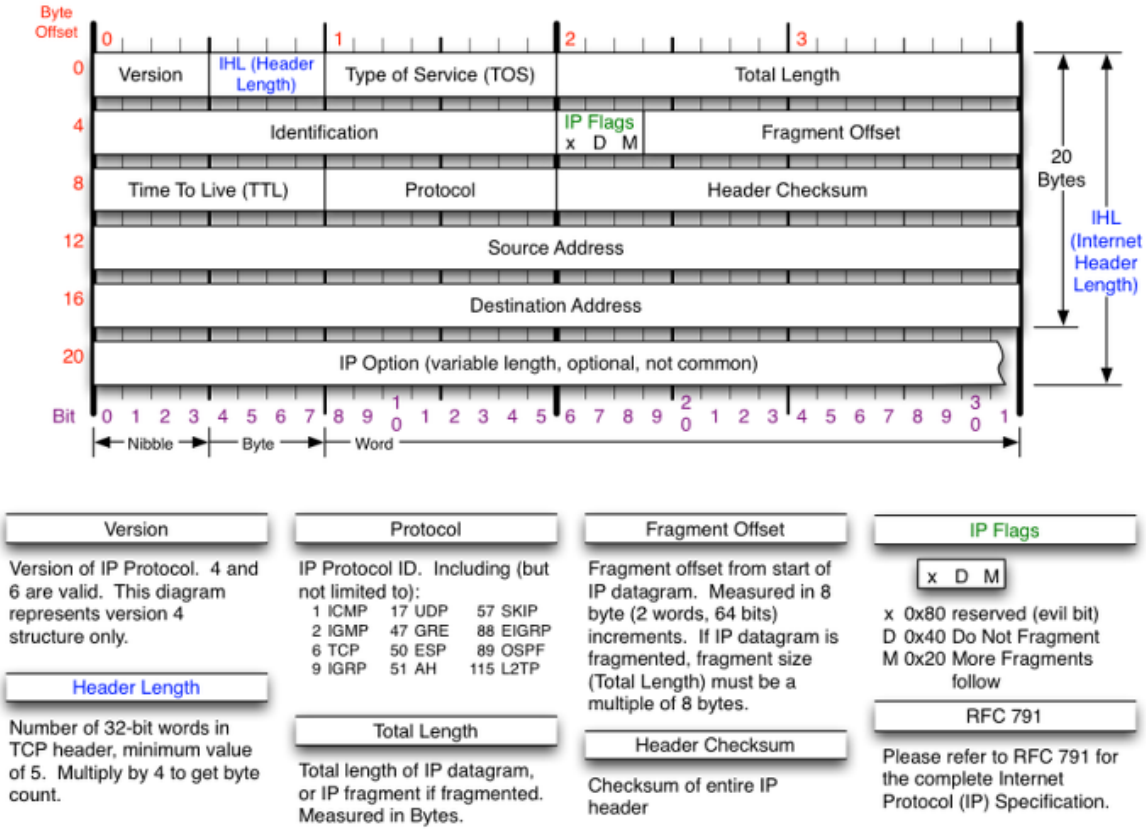


Figure 8: IP Header Diagram

TOS (Type of Service)

ToS는 우선쉰위를 나타내는 3비트의 Precedence 및 4비트의 서비스 유형 지정 비트, 그리고 사용되지 않은 1비트이다. ToS 필드를 사용하여 라우터나 호스트 등의 장치에서 패킷 처리에 대한 우선순위(QoS)를 설정할 수 있는데 현재는 DSCP(Differentiated Serviecs Code Point) 필드로 정의되어 있다.

Total Length

ToS는 우선쉰위를 나타내는 3비트의 Precedence 및 4비트의 서비스 유형 지정 비트, 그리고 사용되지 않은 1비트이다. ToS 필드를 사용하여 라우터나 호스트 등의 장치에서 패킷 처리에 대한 우선순위(QoS)를 설정할 수 있는데 현재는 DSCP(Differentiated Serviecs Code Point) 필드로 정의되어 있다.

Identification

생성되는 각각의 패킷마다 부여된느 고유의 번호이다. 패킷은 제 2계층 프로토콜의 최대 전송 단위(MTU) 값에 따라 여러 개의 프로그먼트(Fragment)로 분할되어 처리되는데, 분할되어 온 fragment들을 원래의 패킷으로 재조립할 때 이 식별자 값을 기준으로 한다.

Flags

IP 패킷의 분할[7] 가능 여부와 마지막 fragment인지 아닌지를 알리기 위해 사용되는 필드이다.

Fragment Offset (분할위치)

하나의 패킷이 여러 개의 프래그먼트로 분할될 때, 각각의 프래그먼트 내의 페이 로드가 원래의 패킷 내의 페이로드를 기준으로 어떤 위치에 있는지를 명시하는 필드이다. 따라서 분할된 프레그먼트들은 이 분할 위치 값을 이용하여 원래의 패킷으로 재조립되게 된다.

[7] fragmentation

TTL (Time to Live)

패킷의 루핑 현상으로 인한 문제를 해결하기 위해 사용하는 필드로 패킷의 수명을 나타낸다. 예를 들어 TTL값은 라우터를 1개 지날 때마다 2씩 감소되어 이 값이 0 이 되면 해당 패킷은 네트워크에서 폐기된다.

Protocol

패킷의 캡슐화되어 있는 상위 계층 PDU가 어떠한 프로토콜을 사용하는지를 명시하는 필드이다. 예를 들어 캡슐화된 PDU가 TCP 세그먼트일 경우 이 필드는 6의 값을, UDP 세그먼트일 경우에는 17의 값을 갖는다.

Head Checksum

P 헤더의 오류 검사를 위한 필드이다. TCP와 UDP를 포함하여 IP 데이터그램으로 캡슐화되는 프로토콜은 대부분 헤더 및 데이터를 포함하는 체크섬 필드를 가지고 있기 때문에 IP 데이터그램의 체크섬 필드는 단순히 IP 헤더에 대한 오류 검사만을 수행한다.

Source IP Address

32비트 길이의 출발지 장치의 ip 주소이다. ip 주소는 네트워크 필드 및 호스트 필드의 길이에 따라 클래스 a, 클래스 b, 클래스 c 주소로 나뉜다. 이 외에도 클래스 D 및 클래스 E 주소가 있지만, 이는 각각 멀티캐스트 주소와 예약된 주소로서 네트워킹 장치나 호스트에 할당하는 주소는 아니다.

Destination IP Address

32비트 길이의 목적지 장치의 IP 주소이다.

Options

패킷의 전송 경로를 포함한 IP 프로토콜의 동작 옵셥을 정의하는 필드이다.

3.b TTL의 의미

IP header field의 설명과 동일하게 , 네트워크의 환경에 따라 부정확한 라우팅 테이블의 결합으로 패킷이 네트워크에서 끝없이 순환할 수 있다. 일정시간이 지나면 패킷을 소거시키기 위해서 각 라우터가 패킷을 받을때 TTL 필드값을 감소시키는 방법으로 이루어진다. 즉 TTL 값은 패킷이 버려지기 전에 허용되는 라우터의 개수로 이해할 수 있다.

3.c fragment의 의미

패킷의 크기가 MTU[8]을 초과하면 한번에 전송할 수 없기 때문에, 패킷을 MTU 보다 작은 조각으로 분할하는 것을 fragmentation이라고 하고, 분할된 조각을 fragment 라고 한다.

[8]Maximum Transmission Unit