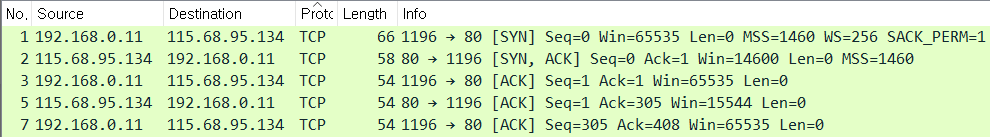
**인터넷 프로토콜 HW4**

12161756 윤성호

**1. 자신의 PC IP 주소, multinet.inha.ac.kr 서버의 IP 주소는?**

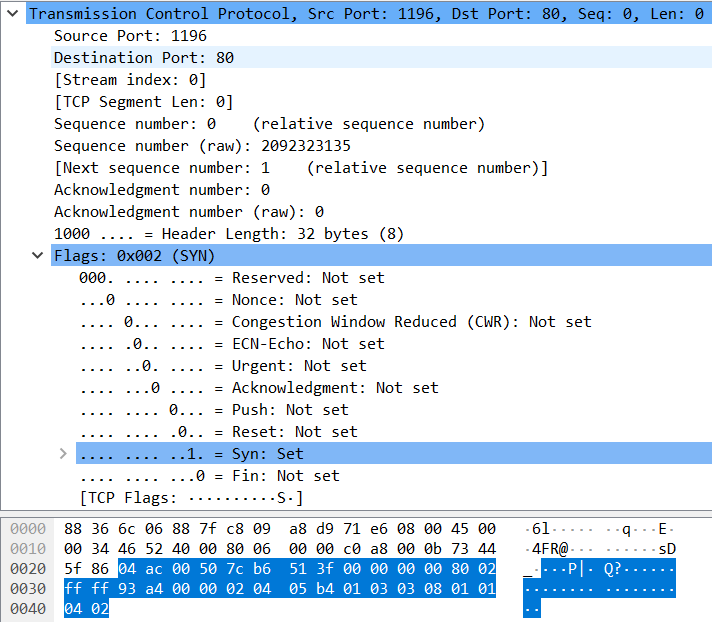


**PC IP : 192.168.0.11**

**multinet.inha.ac.kr : 115.68.95.134**

위 이미지는 Wireshark capture 시작 후 multinet.inha.ac.kr에 접속하면 가장 먼저 기록되는 TCP Segment 5개를 나타낸 것이다. multinet 서버 접속은 클라이언트인 내 PC가 multinet 서버에 SYN 비트가 활성화된 TCP segment를 보내면서 시작된다. 따라서 첫 번째 segment의 Source는 내 PC, Destination은 multinet 서버의 IP 주소이다.

**2. multinet.inha.ac.kr 서버에 접속하는 첫번째 TCP segment (SYN=1) 패킷의 내용을 분석하라**



**1) 내 PC의 source port : 1196**

Source port는 segment를 전송하는 응용 프로그램의 포트번호를 나타내며 TCP 헤더에서 16비트로 표현된다. 첫 번째 TCP segment(SYN=1)를 전송하는 쪽은 내 PC(클라이언트)이기 때문에 source port는 1196이다. 클라이언트의 포트 번호는 임의로 부여된 후 사용을 종료하면 회수된다.

**2) 서버의 HTTP port : 80**

Destination port는 segment를 수신하는 응용 프로그램의 포트번호를 나타내며 16비트로 표현된다. 현재 segment의 수신하는 쪽은 서버의 HTTP이므로 포트 번호는 80이다. 서버의 응용 프로그램은 well-known port number를 이용한다. 왜냐하면 현재 segment가 클라이언트에서 서버로 가는 최초 segment이므로, 만약 서버의 포트 번호가 임의로 부여된다면 클라이언트는 이를 알 방법이 없기 때문이다. 반대로 클라이언트의 포트 번호의 임의 부여가 가능한 이유는, 서버는 클라이언트의 포트번호가 포함된 최초 segment를 해당 클라이언트로부터 수신한 후 응답 segment를 보내기 때문이다.

**3) sequence 번호 : 2092323135**

sequence 번호는 데이터 스트림의 내부적인 순서를 구분하며 32비트로 표현된다. Wireshark에서는 이어지는 segment들의 SEQ와 ACK 변화를 사용자가 보기 쉽도록, 클라이언트와 서버의 첫 segment의 SEQ를 0으로 나타내고 있다. 실제 sequence 번호는 캡처 이미지 하단의 실제 헤더 값을 보면 0x7cb6513f(2092323135)인 것을 알 수 있다.

**4) 각 flag 비트 : Syn(1)**

먼저 각 flag의 의미부터 파악하면,

\* Reserved : 나중에 사용하려고 비워 둔 예약 공간이다. 3bits 공간에 모두 0으로 채워진다.

(1) Nonce : 은폐 보호를 위한 flag이다. RFC 3540에 의해 추가된 flag이다.

(2) CWR : 혼잡 윈도우 축소 플래그를 뜻하며 송신측 호스트에 의해 설정된다. 호스트가 ECE 플래그가 포함된 TCP 세그먼트를 수신했으며 혼잡 제어 메커니즘에 의해 응답했음을 알리는 역할을 한다. RFC 3168에 의해 추가된 flag이다.

(3) ECE-echo : SYN 플래그가 (1)로 설정된 경우, TCP 상대가 ECN이 가능함을 의미한다. SYN 플래그가 (0)인 경우, IP 헤더 셋에 Congestion Experienced 플래그가 설정된 패킷이 정상적인 전송 중에 수신되었다는 것을 의미한다. RFC 3168에 의해 추가된 flag이다.

(4) URG : 긴급 포인터가 있음을 의미한다.

(5) ACK : 확인응답번호가 유효함을 의미한다.

(6) PSH : 데이터를 가능한 빨리 응용계층으로 보내야 함을 의미한다.

(7) RST : 연결을 재설정할 때 이용한다.

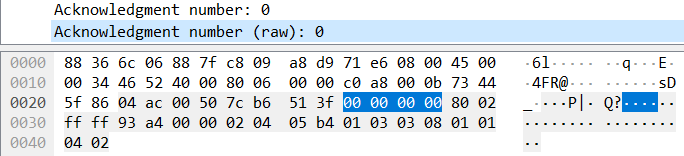
(8) SYN : 연결을 초기화하기 위해 순서번호를 동기화할 때 이용한다.

(9) FIN : 송신 측에서 데이터 전송을 종료할 때, 즉 연결을 해제할 때 이용한다.

현재 segment에서는 SYN을 제외하고 나머지는 0으로 비활성화 돼있다. 이를 통해서도 현재 segment는 최초 연결 설정 시 송신한 segment인 것을 알 수 있다.

**3. 첫번째 TCP 세그먼트에서**

**1) ACK 플래그는 0인데 실제 ACK 필드에 들어간 비트들은 무엇인가? 이경우 ACK 필드에 있는 값은 어떻게 처리되는가?**

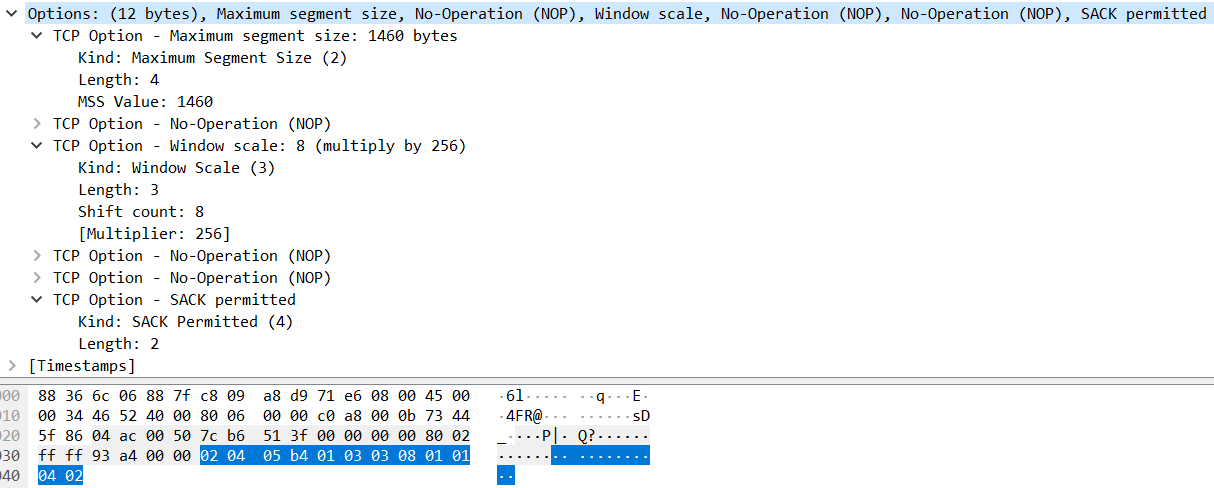


플래그도 0이고 ACK 필드의 실제 값도 0으로 채워져있다. ACK 플래그가 0이기 때문에 ACK 필드는 무시된다.

**2) Sequence 번호의 relative sequence number의 의미는?**

2번 문항에서 답했던 것처럼, 이어지는 segment들의 SEQ와 ACK 변화를 사용자가 보기 쉽도록 Sequence 번호를 0으로 시작하고 있다.

**3) Option들의 의미는?**



각 옵션들은 1byte Kind값으로 구분이 가능하다.

(1) Maximum segment size (MSS)

: 한 번에 전송할 수 있는 최대 세그먼트 크기를 알려준다. 위 캡처에서는 1460bytes를 한 번에 전송할 수 있음을 나타낸다. Kind 값이 2이며 옵션 길이는 4bytes이다.

(2) NOP

: TCP 옵션의 크기를 4byte의 배수로 맞추기 위해 이용된다. NOP의 크기는 1byte이다. NOP 이후 이어지는 옵션인 Window Scale 크기가 3bytes이기 때문에 Window Scale 앞에 한 개의 NOP를 두어 4byte를 맞추는 모습이다.

(3) Window Scale

: TCP 헤더의 윈도우 크기를 나타내기에 필드 값(최대 65,535bytes)이 부족할 때 이용하는 옵션으로, Window Scale값만큼 2의 거듭제곱하여 헤더에서 정의된 윈도우 크기에 곱하여 새로운 윈도우 크기를 정의한다. 여기서는 Window Scale 값이 8이므로 65,535bytes(헤더의 값) \* 2^8 = 16MB가 새로운 윈도우 크기가 된다. Kind 값은 3이며 옵션 길이는 3bytes이다.

(4) SACK permitted

: 선택 확인응답(SACK) 옵션을 허용함을 의미한다. SACK 옵션이 허용돼 있다면 Flow Control에서 Sequence 번호가 맞지 않는 데이터가 들어올 경우 따로 저장해 두었다가 duplicated ACK를 통해 알맞은 Sequence 번호를 가진 데이터를 받게 될 경우 이전에 저장한 잘못 전송된 데이터만 전송하게 된다. Kind 값은 4이며 옵션 길이는 2bytes이다. 따라서 4byte 배수를 맞추기 위해 앞에 2개의 NOP가 자리한 것을 확인할 수 있다.

**4. multinet 서버에 연결하기 위해 three-way handshake를 수행하는 것을 볼 수 있다. 세개의 TCP segment 내용을 화면 캡쳐하고 이를 상호 분석하고 분석한 내용을 중심으로 절차에 대한 간단한 그림을 그려라 (강의 노트와 유사하게).**

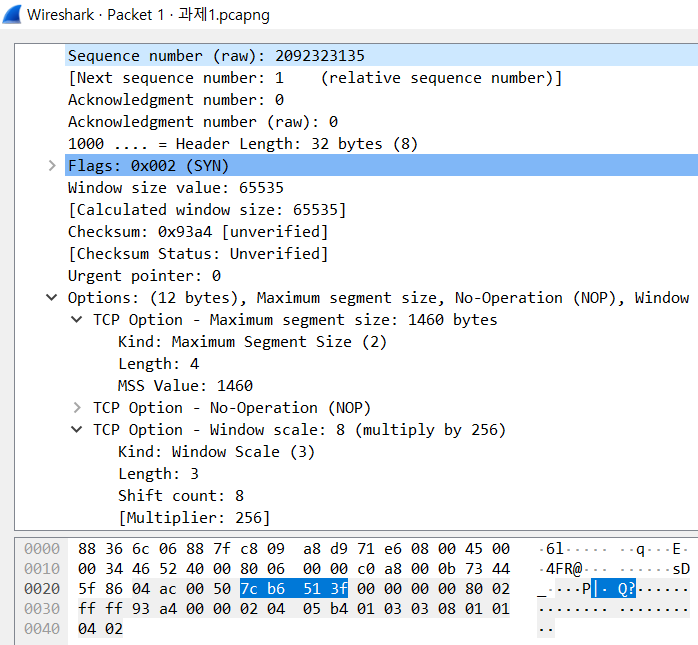


그림 1. 1st segment

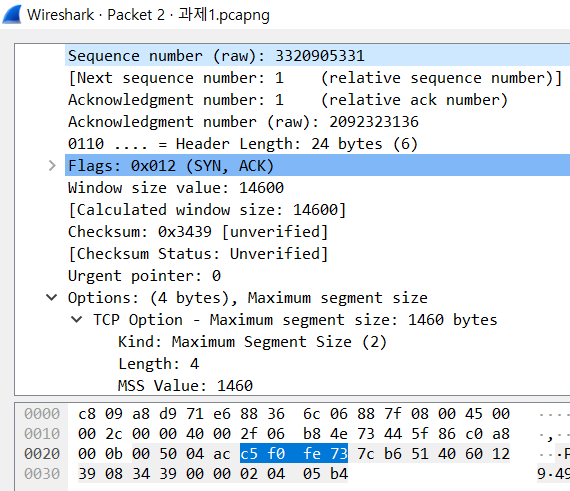


그림 2. 2nd segment

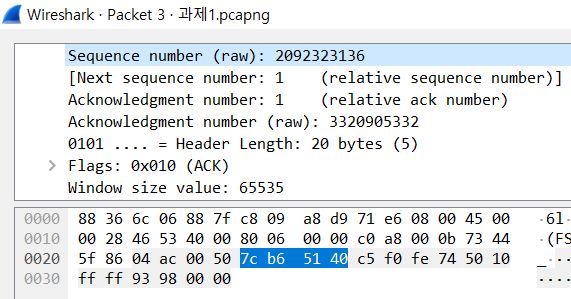


그림 3. 3rd segment

**1) 설정된 MSS**

첫 번째와 두 번째 segment에서 클라이언트와 서버는 MSS값을 1460bytes로 협의했다.

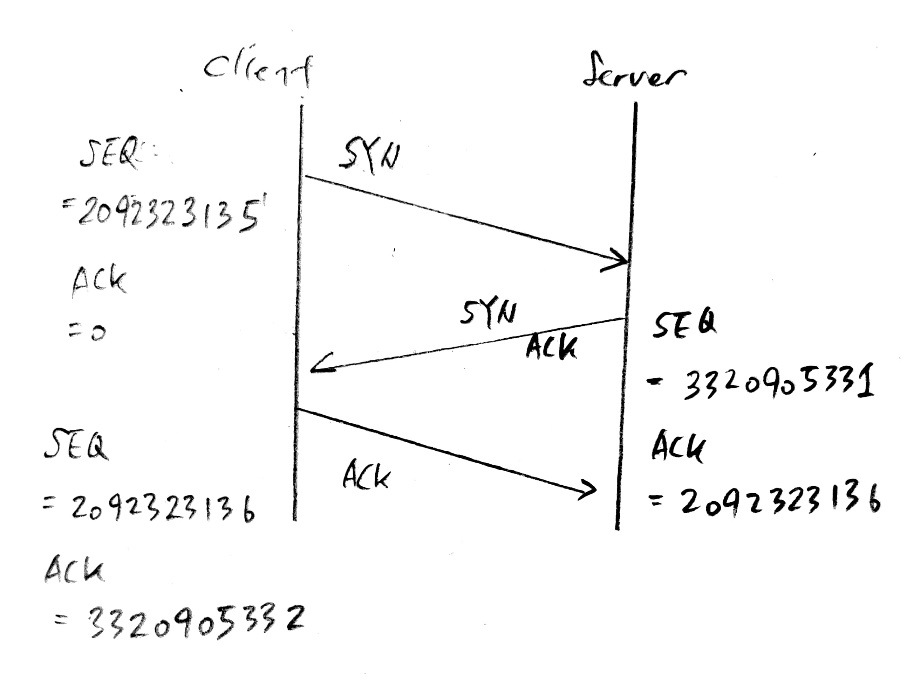
**2) 양측의 SEQ 시작 번호 (oX 값 직접 분석)**

클라이언트의 SEQ 시작 번호는 2092323135이고 서버 쪽은 3320905331이였다. 클라이언트로 부터 첫 번째 segment를 받은 서버는, ACK값으로 클라이언트의 SEQ+1=2092323136값을 보낸다. 두 번째 segment를 받은 클라이언트는 ACK값으로 서버의 SEQ+1=3320905332값을 보낸다.

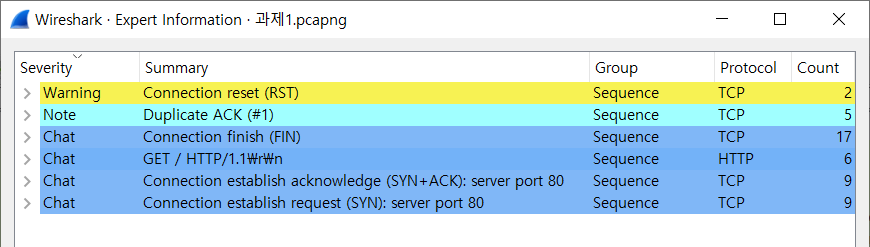
**3) Window Size 변화**

16MB -> 14600byte -> 65535byte로 변화하였다. 첫 번째 segment에만 Window scale 옵션이 있었다.

이를 그림으로 나타내면 아래와 같다.



**5. 재전송 (retransmission) 된 패킷이 있는가? 있다면 어떻게 알 수 있었는가?**



재전송 된 패킷이 없었다. Wireshark 메뉴의 Analyze > Expert Information에서 확인할 수 있다.

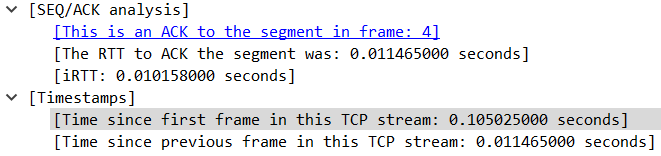
**6. 초기 3 way handshake를 제외하고 5개정도의 매 TCP segment 전송에 대해 언제 ACK가 수신되었는지 시간을 계산하고 5개의 RTT 시간을 구한다.**

**- 마지막 5개 이후 계산된 SRTT (a=0.8) 값은 = 0.01091523168s**

**- 5개 segment 각각의 TCP segment 크기는 = 20, 20, 1480, 20, 1480bytes**

**1) 5번 패킷**





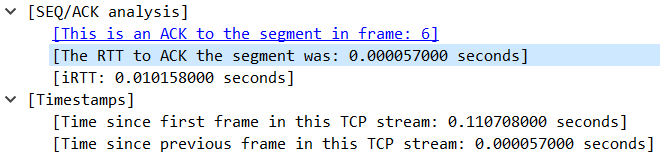
3 way handshake 이후 바로 이어지는 ACK 응답으로 RTT는 **0.011465s**이다.

첫 측정이기에 SRTT는 그대로 **0.011465s**가 된다.

TCP segment 크기는 기본 header만 있기에 **20bytes**이다.

**2) 7번 패킷**





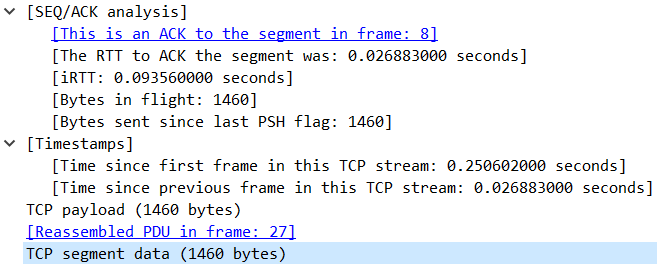
그 다음 ACK의 RTT는 **0.000057s**이다.

SRTT = 0.011465s \* 0.8 + 0.0000057s \* 0.2 = **0.00917314s**이다.

마찬가지로 기본 header만 있기에 **20bytes**이다.

**3) 9번 패킷**





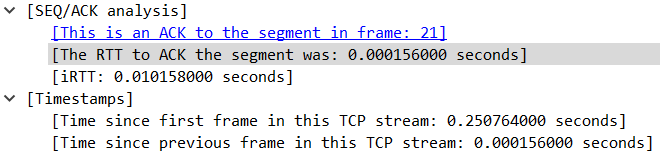
그 다음 ACK의 RTT는 **0.026883s**이다.

SRTT = 0.00917314s \* 0.8 + 0.026883s \* 0.2 = **0.012715112s**이다.

TCP 기본 header + TCP data = 1460 + 20 = **1480bytes**이다.

**4) 23번 패킷**





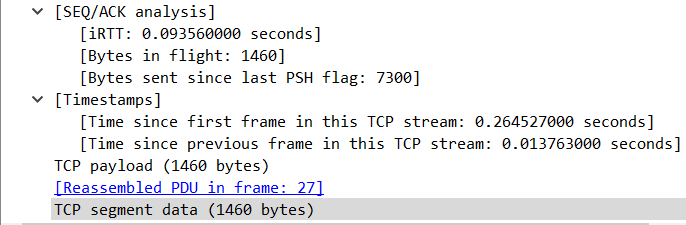
그 다음 ACK의 RTT는 **0.000156s**이다.

SRTT = 0.012715112s \* 0.8 + 0.000156s \* 0.2 = **0.0102032896s**이다.

TCP 기본 header만 있어 **20bytes**이다.

**5) 24번 패킷**





그 다음 ACK의 RTT는 **0.013763s**이다.

SRTT = 0.0102032896s \* 0.8 + 0.013763s \* 0.2 = **0.01091523168s**이다.

TCP 기본 header + TCP data = 20 + 1460 = **1480bytes**이다.

**- 5개 segment 중 수신자 (즉 multinet.inha.ac.kr 서버)의 최소 가용 버퍼 용량은 얼마 이었나**

(최소 가용 버퍼 용량의 의미를 잘 모르겠습니다.)

일단 Window size를 통해 자신이 수용할 수 있는 데이터 양을 전달해준다.

5개 segment 중 서버의 Window size가 포함된 패킷은 Source ip가 115.68.95.134인 5, 9, 24번 패킷이며 각각 **15544, 16616, 16616bytes**이다.