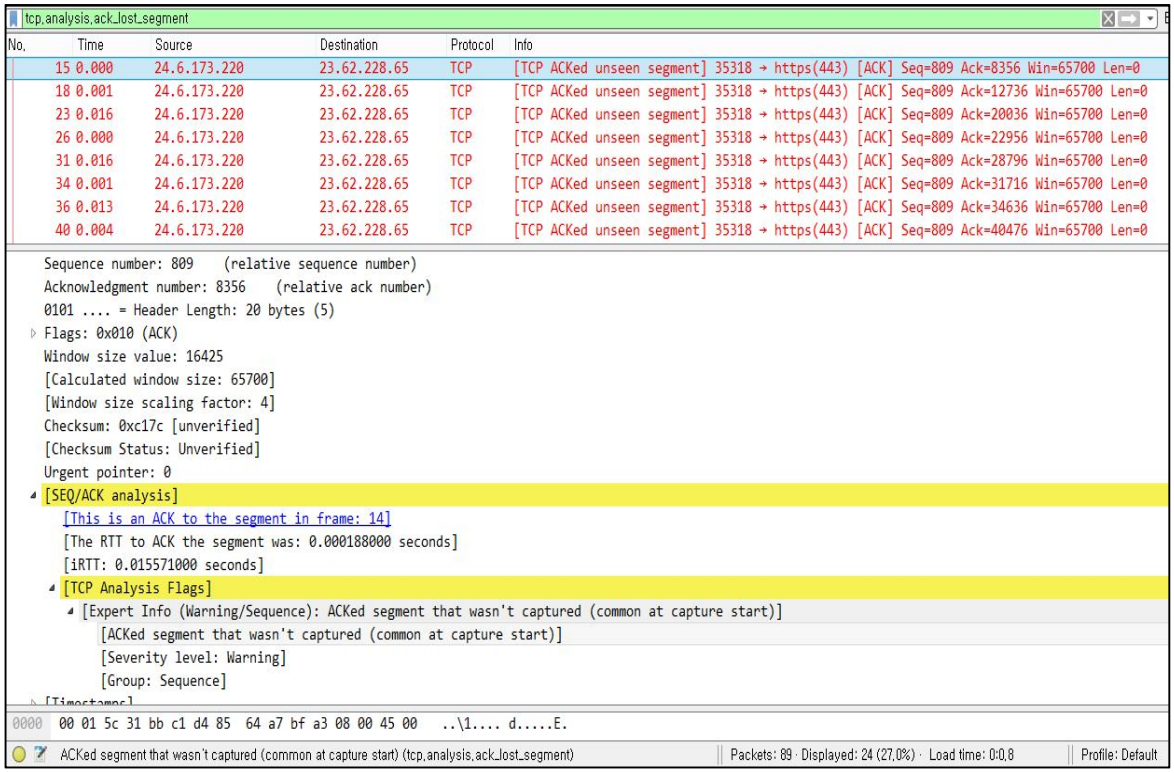
미도착 세그먼트 확인 응답

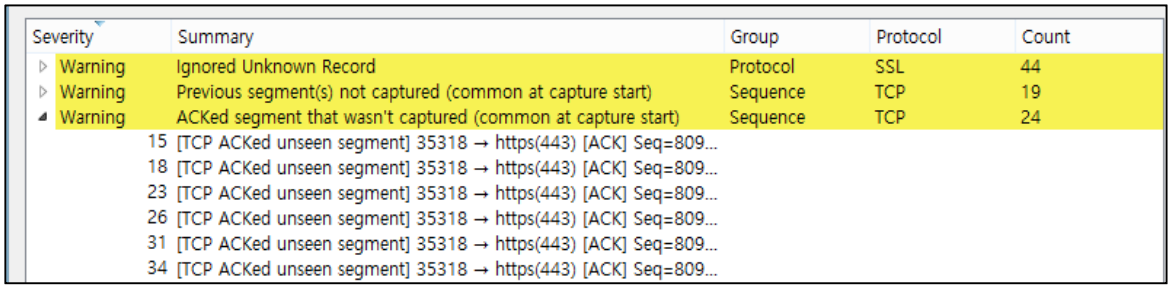
- 캡처 과정 중의 문제로 발생 가능
- 미도착 세그먼트 확인 응답은 비대칭 라우팅의 징후일 수도 있음
- 데이터가 네트워크의 한 경로를 따라 흐를 때 ACK는 다른 경로를 따라 전송될 가능성이 높음
- 와이어샤크가 두 번째 경로에서 캡처하고 있다면 데이터는 보이지 않고 ACK만 보게 됨
- 이 지점은 데이터 스트림의 일부분 밖에 볼 수 없기 때문에 트래픽을 캡처하기에 좋은 지점은 아님
- 미도착 세그먼트 확인 응답의 추적파일은 오탐이 많이 많기 때문에 분석하지 않는 것을 권장

[확인 방법]

① tcp.analysis.ack_lost_segment



② 전문가 정보(expert infos)로 미도착 세그먼트 확인 응답 알림 발견



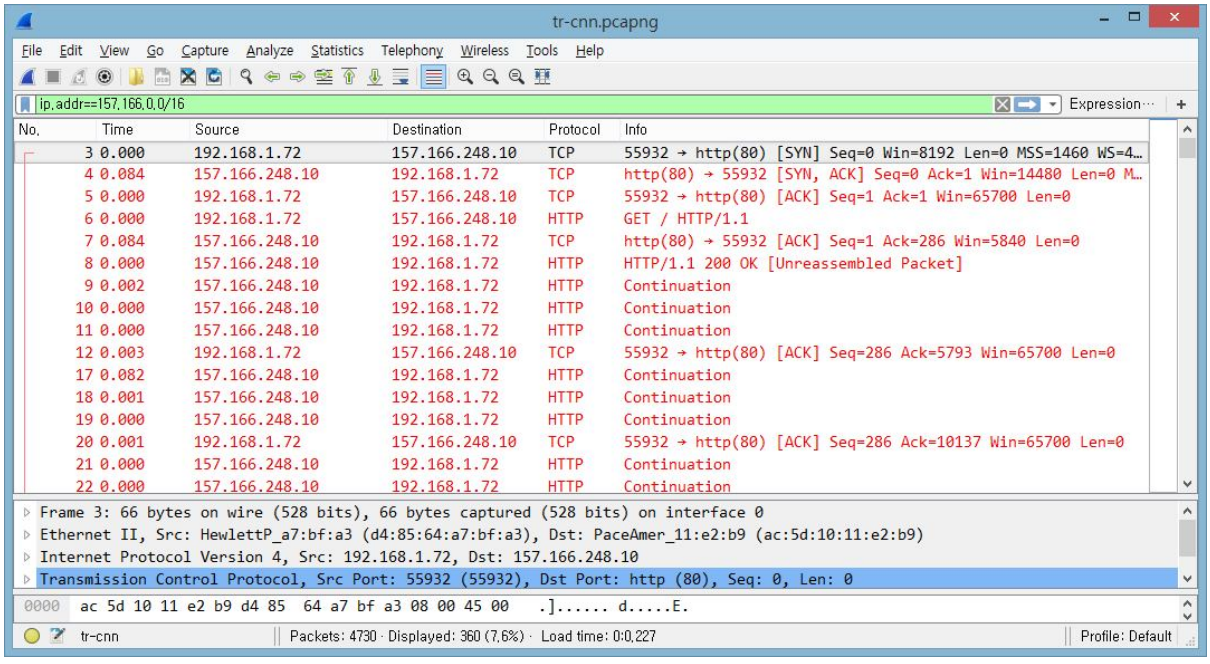
특정 호스트, 서브넷, 대화 필터링

- 특정한 클라이언트와 서버 사이의 트래픽을 조회 할 경우 호스트주소, 서브넷주소, 대화를 디스플레이 필터로 적용하여 검색
- Statistics > Resolved Addresses > 검색 사이트의 IP 주소또는 IP 대역 검사

① [특성호스트 IP 대역조회] Statistics > Resolved Addresses > cnn.com 조회

```
208.89.161.62    www.cnnimagesource.com
50.19.214.97     dualstack.log-334788911.us-east-1.elb.amazonaws.com
157.166.226.110  archives.cnn.com
157.166.224.32   svcs.cnn.com
66.155.9.238     lb.wordpress.com
76.74.254.120    lb.wordpress.com
176.32.100.5     s.amazon-adsystem.com
149.174.149.82   living.blogsmith.aol.com.aol.akadns.net
64.12.68.35      ads.adsonar.akadns.net
```

② [디스플레이 필터] ip.addr==157.166.0.0/16



③ [추적 파일 생성] File > Export Specified Packet > Save

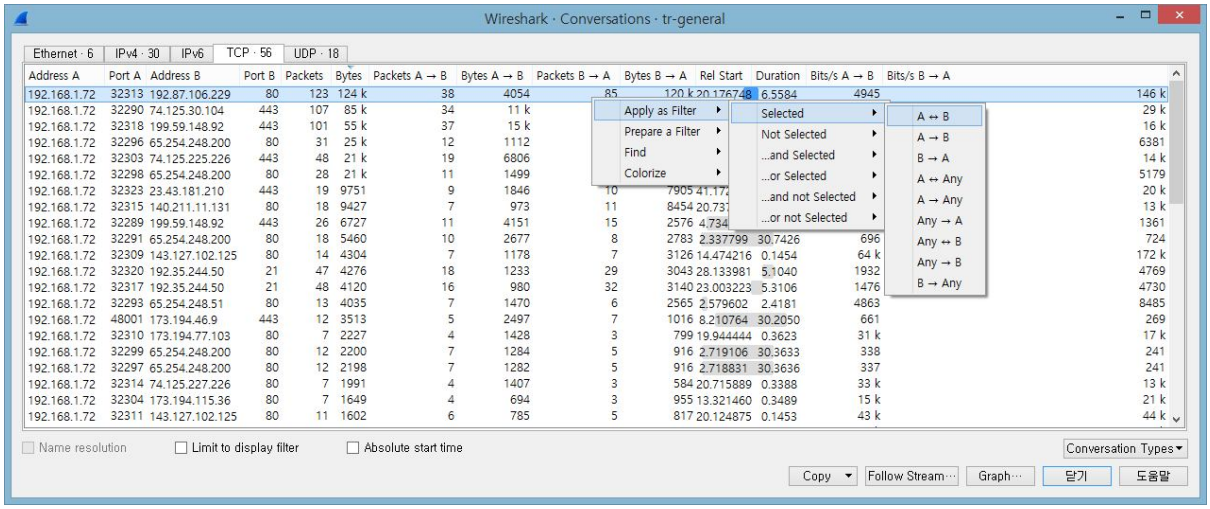
최다 대화자 조회

- 하드웨어주소, 네트워크주소, 포트번호로 가장 활발한 대화를 조회하기 위해 대화창 사용
- Statistics > Conversation 사용 > 송수신 바이트 수를 기준으로 최다 대화자 검색

① [종류별 대화 개수 확인] Statistics > Conversations

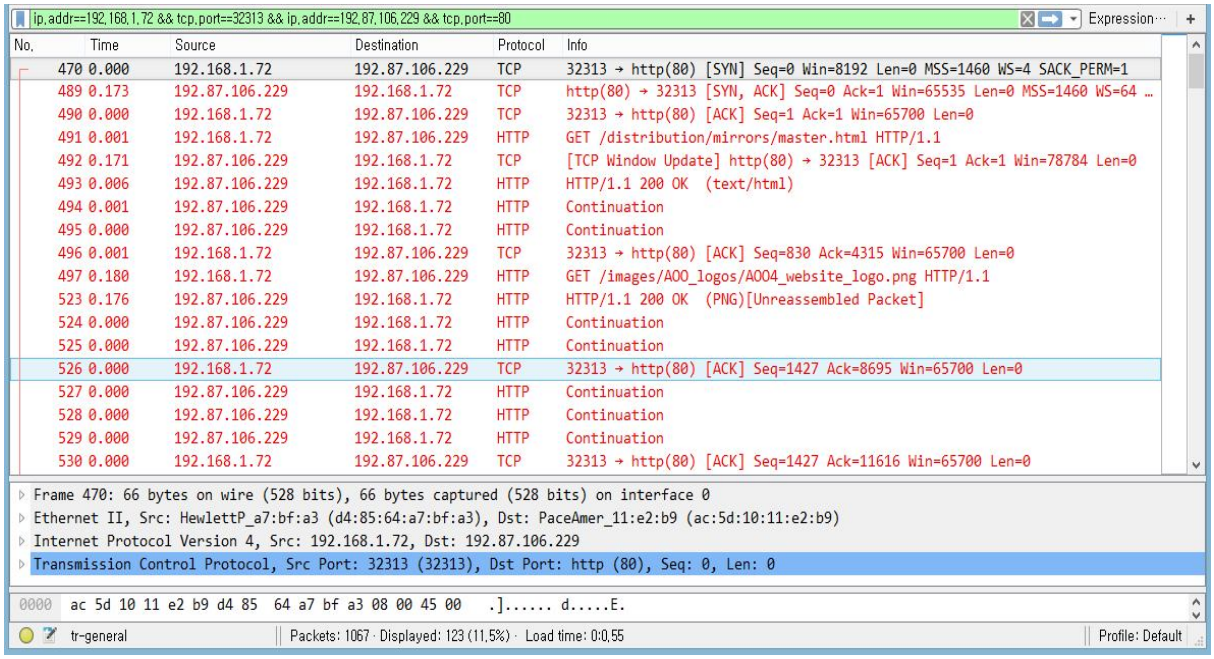
② TCP 탭 클릭 > Bytes 칼럼 제목필드 더블클릭 후 정렬

③ 송수신바이트가 가장 큰 레코드 선택 > Apply as Filter > Selected > A↔ B



④ [디스플레이필터 자동 생성]

ip.addr==192.168.1.72 && tcp.port==32313 && ip.addr==192.87.106.229 && tcp.port==80

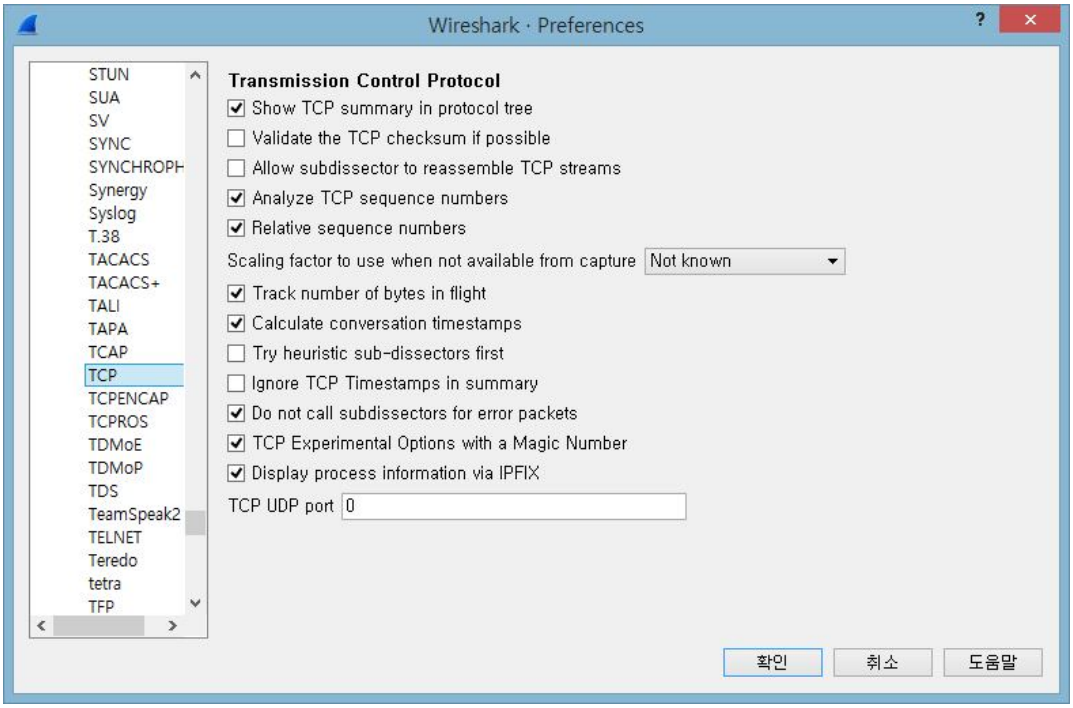


TCP 핸드셰이크를 이용한 왕복 시간 계산

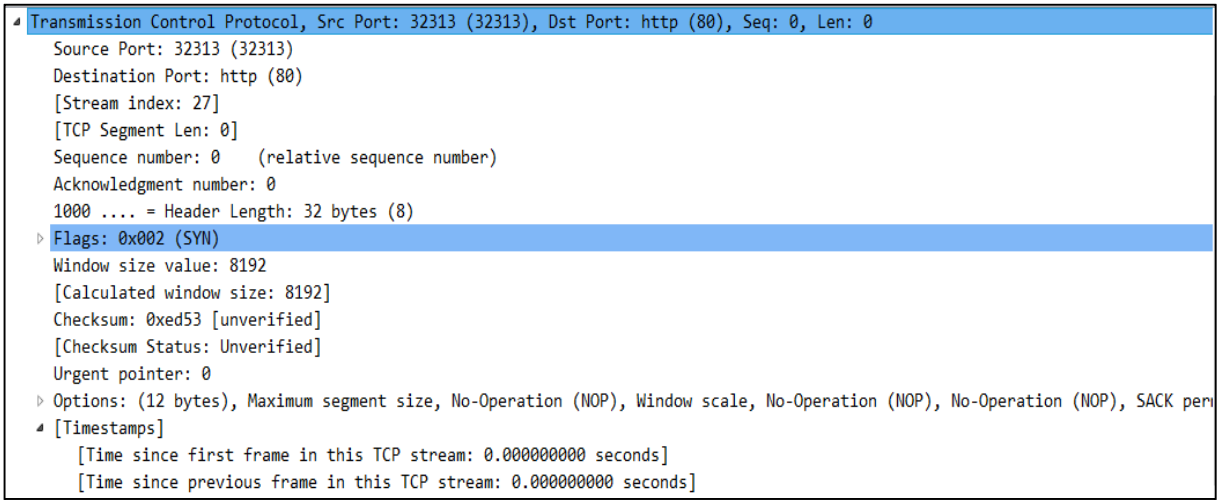
- 성능 문제의 원인이 경로 대기 시간인지 확인하는 것이 중요

(예) TCP 대화가 정상적으로 동작하고 왕복시간(RTT, Round Trip Time)만 큰 값일 때 파일 전송 프로세스가 느리게 보일 수도 있음

① [프로토콜 별 환경 설정] Edit > Preferences > Protocol > Calculate conversation timestamps



② [Packet detail] TCP 헤더 정보 필드 중 Timestamp 확인

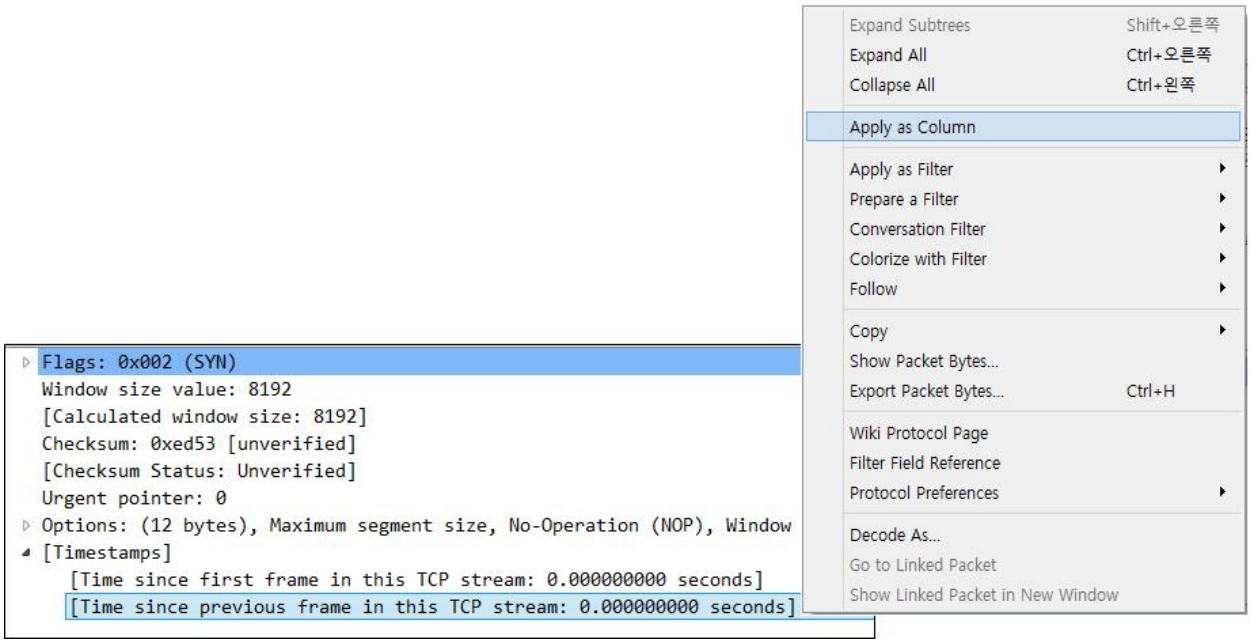


tcp.time_relative : 현재 TCP 스트림의 첫 프레임으로부터의 시간

tcp.time_delta : 현재 TCP 스트림의 이전 프레임으로부터의 시간

③ [TCP 시간차 칼럼 추가]

Time Since previous frame in this TCP Stream > 오른쪽 클릭 > Apply as Column 선택



④ [SYN과 SYN/ACK 패킷 필터] tcp.flags.syn ==1

핸드세이크 패킷	필터
1번과 2번	tcp.flags.syn==1
2번	tcp.flags.syn==1 && tcp.flags.ack==1
3번	(tcp.flags.syn==1 && tcp.flags.ack==1) || (tcp.seq==1 && tcp.ack==1)
1번과 3번	(tcp.flags.syn==1) || (tcp.seq==1 && tcp.ack ==1)

⑤ tcp.time_delta 칼럼 더블 클릭으로 시간 지연 정렬

tcp.flags.syn==1

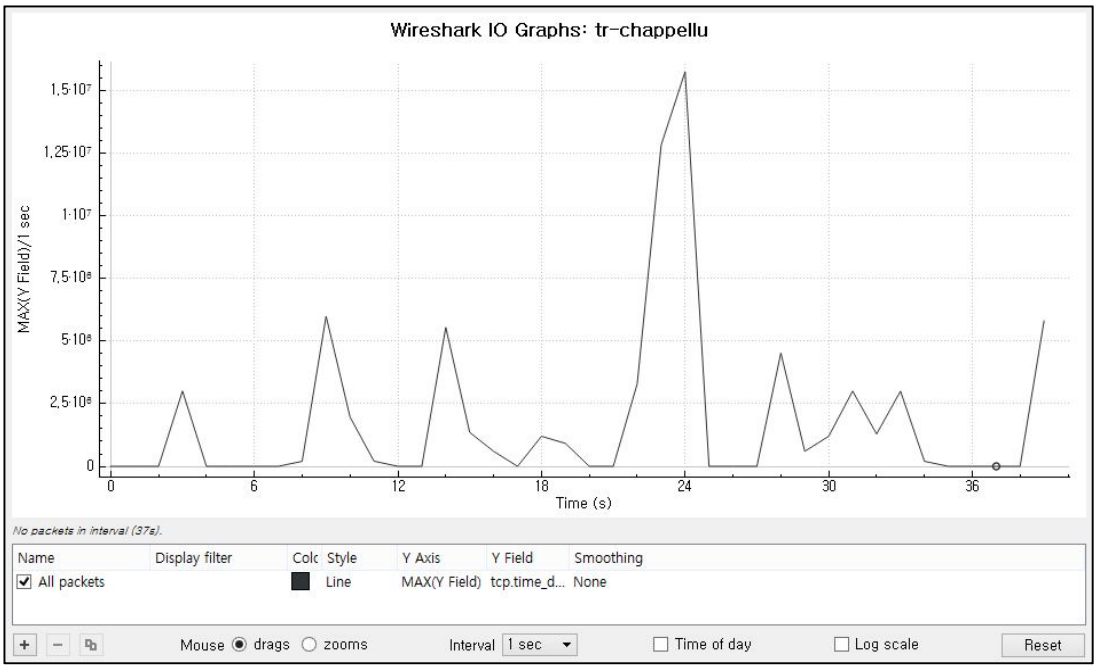
No.	Time	Source	Destination	Protocol	TCP Delta	Info
4730	6.002	63.166.98.136	192.168.1.72	TCP	6.002244000	[TCP Retransmission] http(80) → 56
4101	0.442	157.166.226.32	192.168.1.72	TCP	3.480712000	[TCP Retransmission] http(80) → 56
4594	0.339	63.166.98.136	192.168.1.72	TCP	0.339694000	[TCP Retransmission] http(80) → 56
2559	0.160	199.7.71.72	192.168.1.72	TCP	0.184279000	http(80) → 56021 [SYN, ACK] Seq=0
2568	0.021	199.7.71.72	192.168.1.72	TCP	0.181178000	http(80) → 56022 [SYN, ACK] Seq=0
945	0.078	46.51.168.42	192.168.1.72	TCP	0.176830000	http(80) → 55957 [SYN, ACK] Seq=0
1949	0.025	216.38.164.159	192.168.1.72	TCP	0.113300000	http(80) → 55993 [SYN, ACK] Seq=0
1420	0.007	216.38.164.159	192.168.1.72	TCP	0.111955000	http(80) → 55980 [SYN, ACK] Seq=0
2061	0.007	50.31.185.44	192.168.1.72	TCP	0.110960000	http(80) → 56004 [SYN, ACK] Seq=0

연결 설정 시 문제가 발생되는 것을 파악 할 수 있음

TCP 지연 그래프

• TCP 대화 지연을 찾을 때 tcp.time_delta 최대값을 그래프로 사용

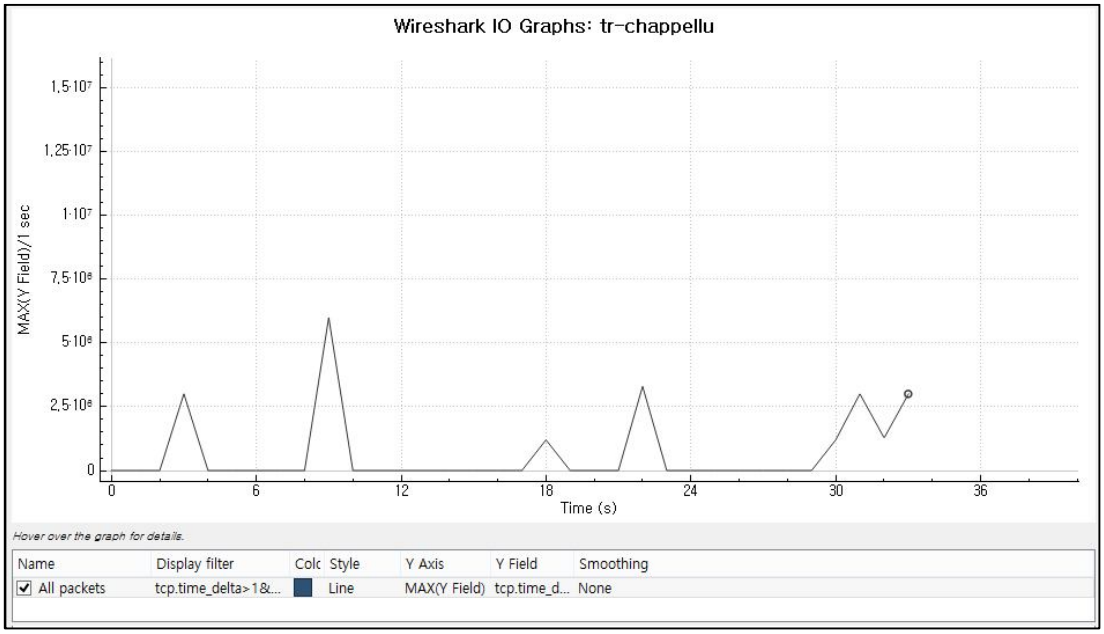
① Statistics > I/O Graph > Y 필드 축 지정(tcp.time_delta / MAX(delta))



⇨ 해당 패킷은 TCP FIN 패킷이므로 신경 쓸 필요가 없는 지연

② 허용 가능한 지연을 뷰에서 제거

tcp.time_delta>1&&tcp.flags.fin==0 && tcp.flags.reset==0 && tcp.flags.reset ==0&&!http.request.method=="GET"



➤**'정상' 또는 '허용 가능한 지연'**

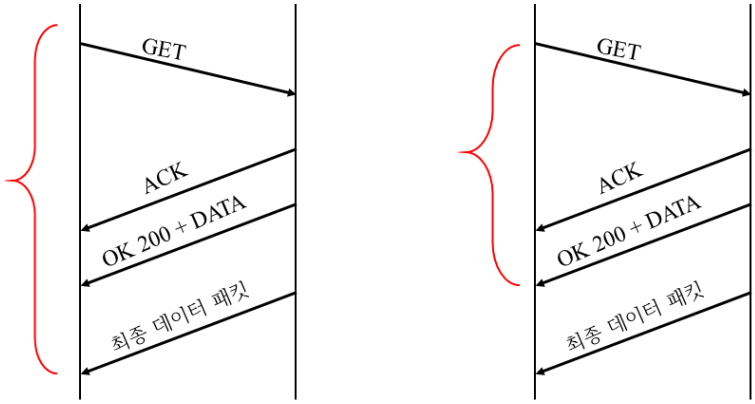
DNS 쿼리전 지연
TCP FIN 또는 RST 패킷 전 지연 - tcp.flags.fin == 0 && tcp.flags.reset == 0 - 애플리케이션은 미리 정해진 시간만큼 기다린 후 또는 작업 종료 후에 FIN 또는 RST 패킷을 보내 연결 종료 - 사용자는 연결이 종료되는 것을 알지 못함
클라이언트가 서버에 요청 보내기 전 지연
Keep-Alive E또는 제로 윈도우 prove전 지연
TCP FIN 또는 RST 앞에 나오는 TLS암호화된 경고 전 지연
주기적 연결 패킷 전 지연

➤ **조회 해야 할 지연**

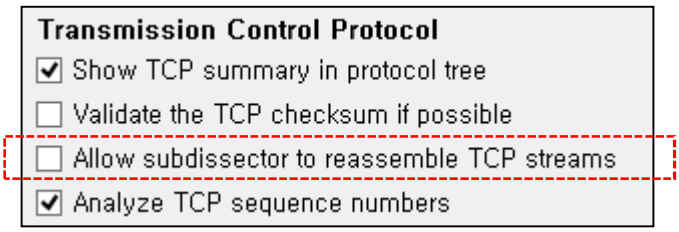
서버의 SYN/ACK 응답 전 지연
클라이언트의 3방향 TCP 핸드셰이크 종료 전 지연
서버 응답 전송 전 지연
데이터 스트림의 다음 패킷 전 지연
TCP 피어로부터 ACK 전 지연
윈도우 업데이트 전 지연

긴 HTTP 응답 시간 찾기

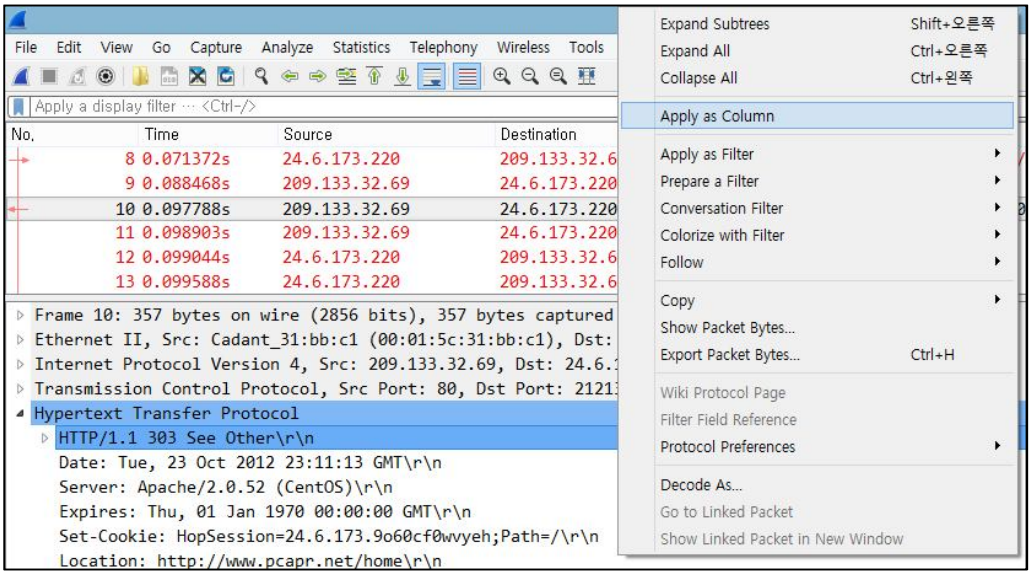
- 긴 HTTP 응답시간은 웹 서버에 연결 또는 서비스 요청이 폭주하거나 클라이언트 요청에 답하기 위해 다른 서버에 질의해야 할 때 생성
- HTTP 응답 시간을 측정 시 서비스 요청(HTTP GET 요청)과 응답(HTTP 200 OK) 사이의 시간차 조회
- HTTP 응답 시간 필드 : http.time
- 'Allow Sub-dissector to reassemble TCP stream' 활성 여부에 따라 http.time이 다름



(a) 활성화 시 (b) 비활성화 시



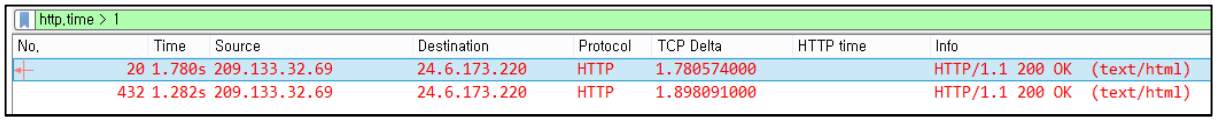
- HTTP 응답 시간 칼럼 추가



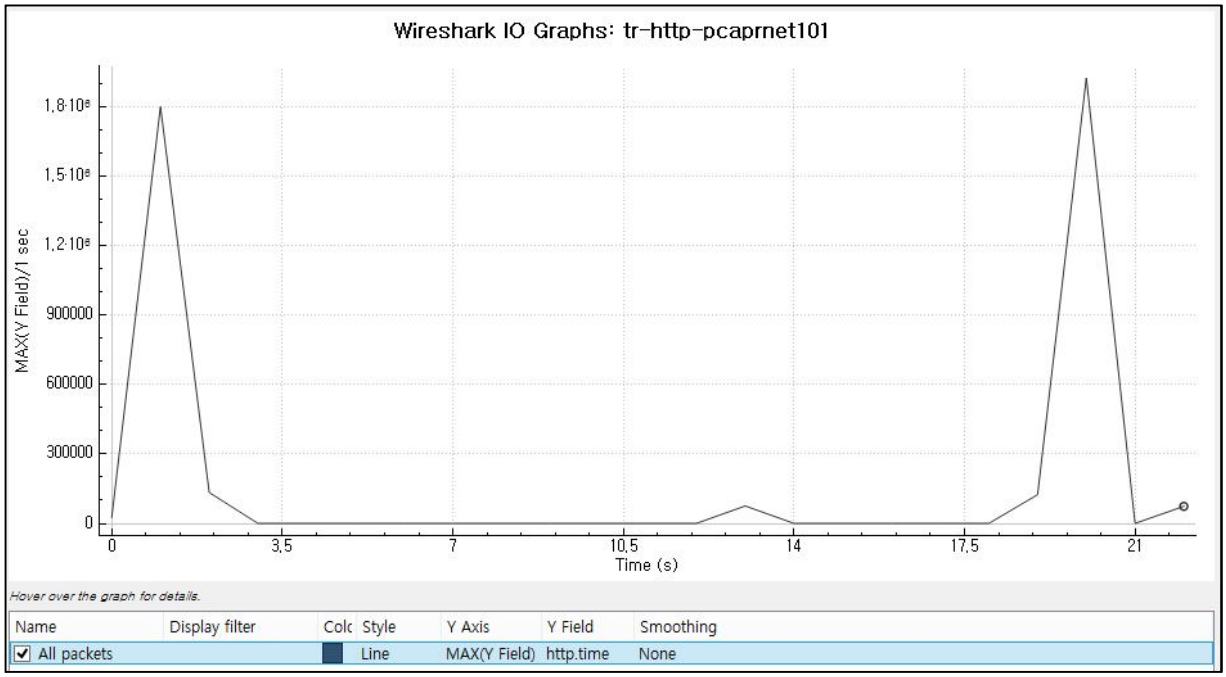
⇨ http.time 필드는 HTTP 응답 패킷에만 존재함

① 1초보다 긴 HTTP 응답 시간 탐지 필터

http.time > 1



② [HTTP 응답 시간 그래프 생성] Statistics > I/O Graph > Y Filed (http.time)



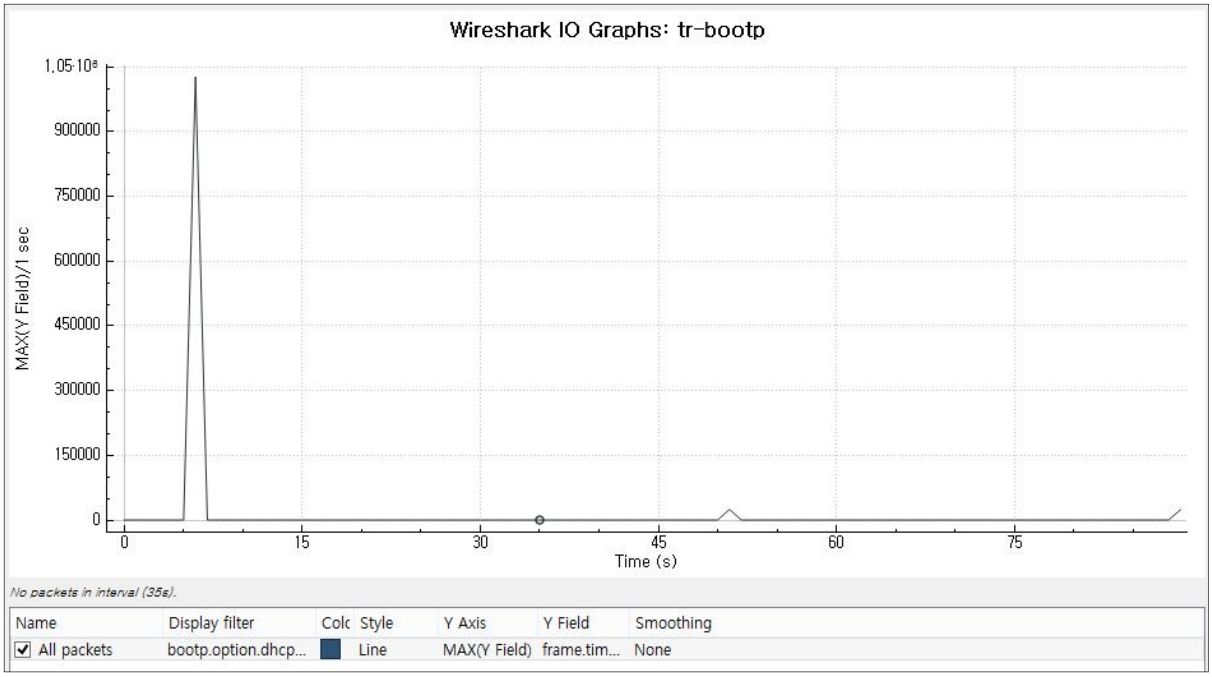
느린 DHCP 서버 응답 그래프
(UDP 기반 애플리케이션)

- 시간차(delta time) 기능이 없는 애플리케이션인 DHCP의 시간차 그래프 생성
- 시간차 기능을 가지고 있지 않은 애플리케이션의 느린 응답을 식별 시 사용

① [DHCP 응답 시간 그래프 생성] Statistics > I/O Graph > Y Filed (frame.time_delta)

② 느린 DHCP offer 패킷을 찾기 위한 필터 지정

bootp.option.dhcp == 2



한 지점에서 DHCP offer가 이전 프레임 다음 1초후에 도착

```
1 0.000s  192.168.1.72      192.168.1.254      DHCP     DHCP Release  - Transaction ID 0x2b5825c3
2 5.166s  0.0.0.0           255.255.255.255    DHCP     DHCP Discover - Transaction ID 0xa69b8b3f
3 1.027s  192.168.1.254     255.255.255.255    DHCP     DHCP Offer    - Transaction ID 0xa69b8b3f
4 0.001s  0.0.0.0           255.255.255.255    DHCP     DHCP Request  - Transaction ID 0xa69b8b3f
```

TCP 시간 차 그래픽

(TCP 기반 애플리케이션)

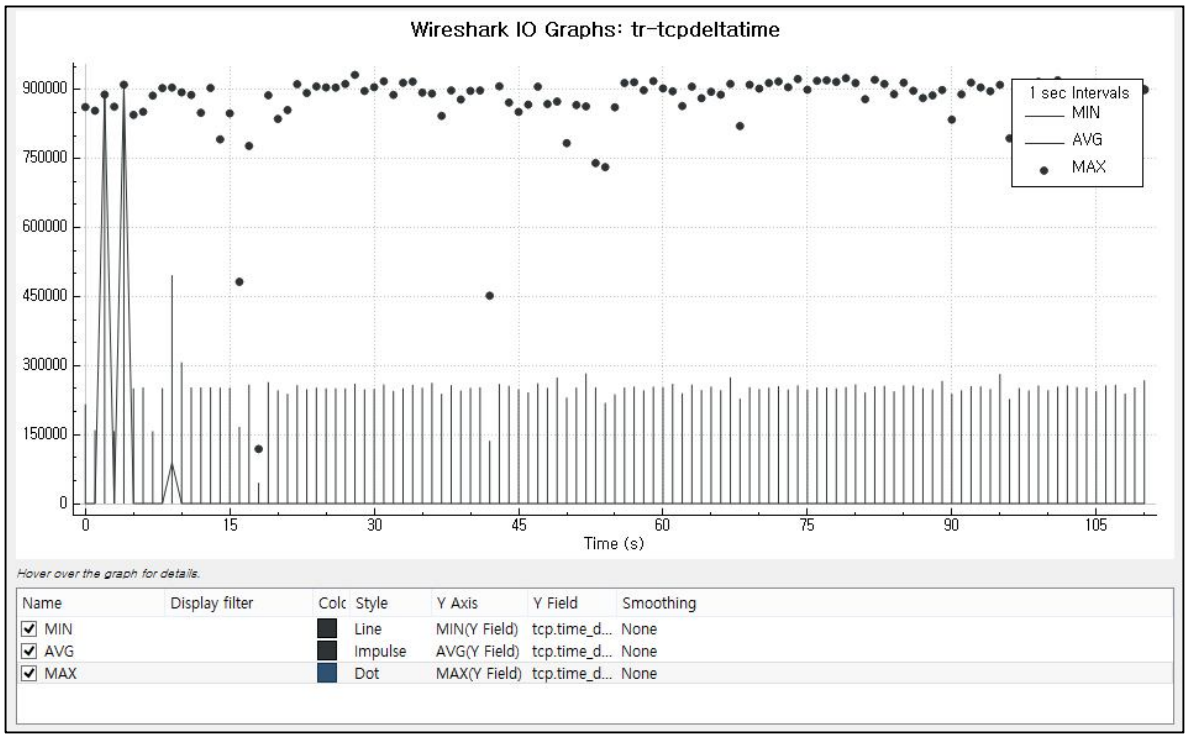
① [TCP 시간차그래프 생성] Statistics > I/O Graph > Y Filed (tcp.time_delta)

② 최소, 평균, 최대 응답 시간을 한 그래프에 생성하여 응답시간의 시간차 비교

tcp.time_delta / MIN / LINE

tcp.time_delta / AVG/ Impulse

tcp.time_delta / MAX / Dot



⇨ 추적 파일의 평균 응답시간이 300ms에 약간 못 미치고 앞쪽 부분에 큰 TCP 응답시간이 존재하는 것을 그래프로 확인 할 수 있음