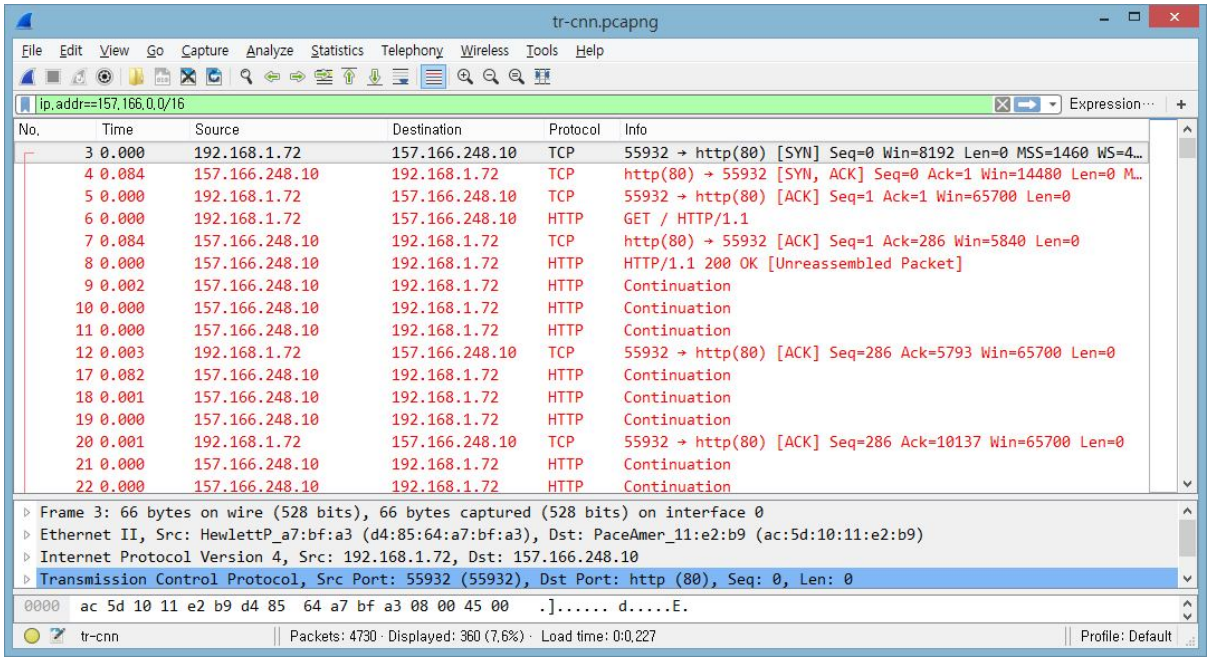
특정 호스트, 서브넷, 대화 필터링

- 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

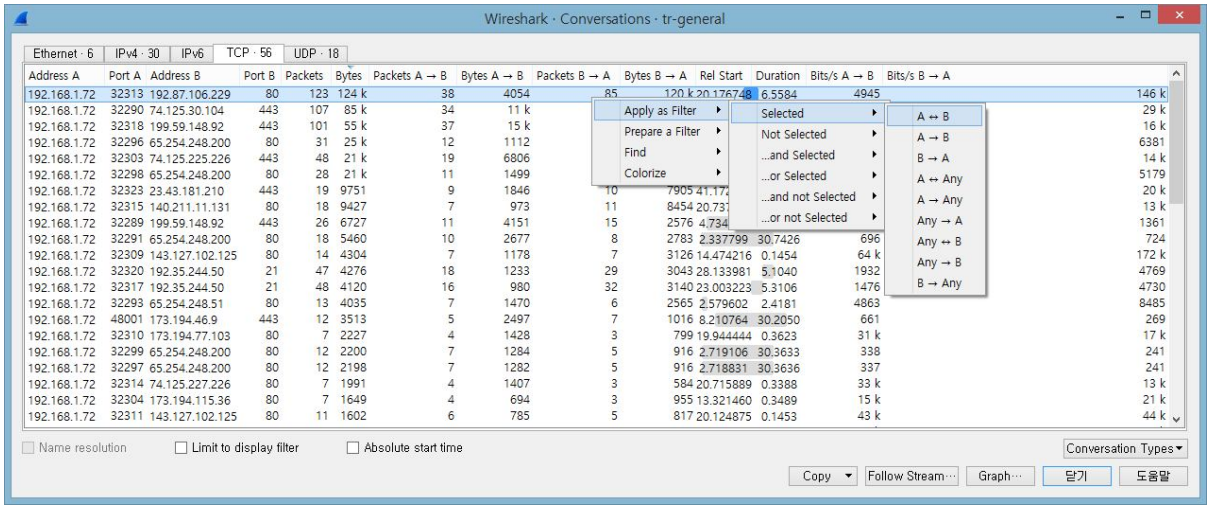
최다 대화자 조회 (tr.general.pcapng)

- 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

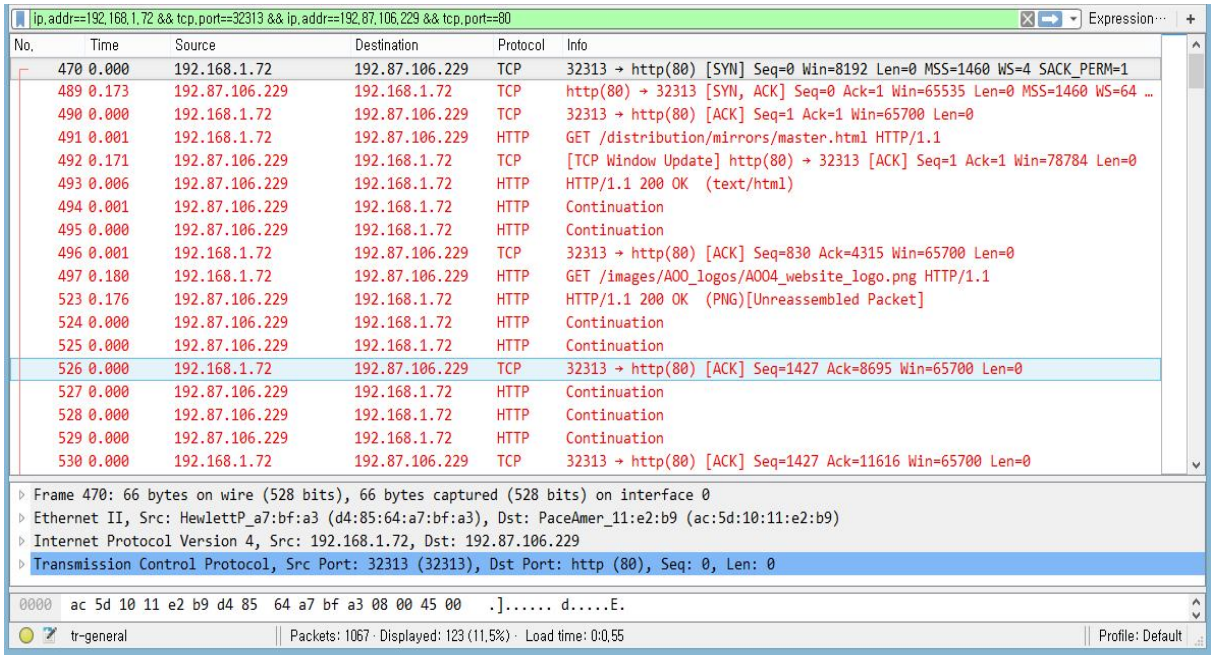
② TCP 탭 클릭 > Bytes 칼럼 제목필드 더블클릭 후 정렬

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

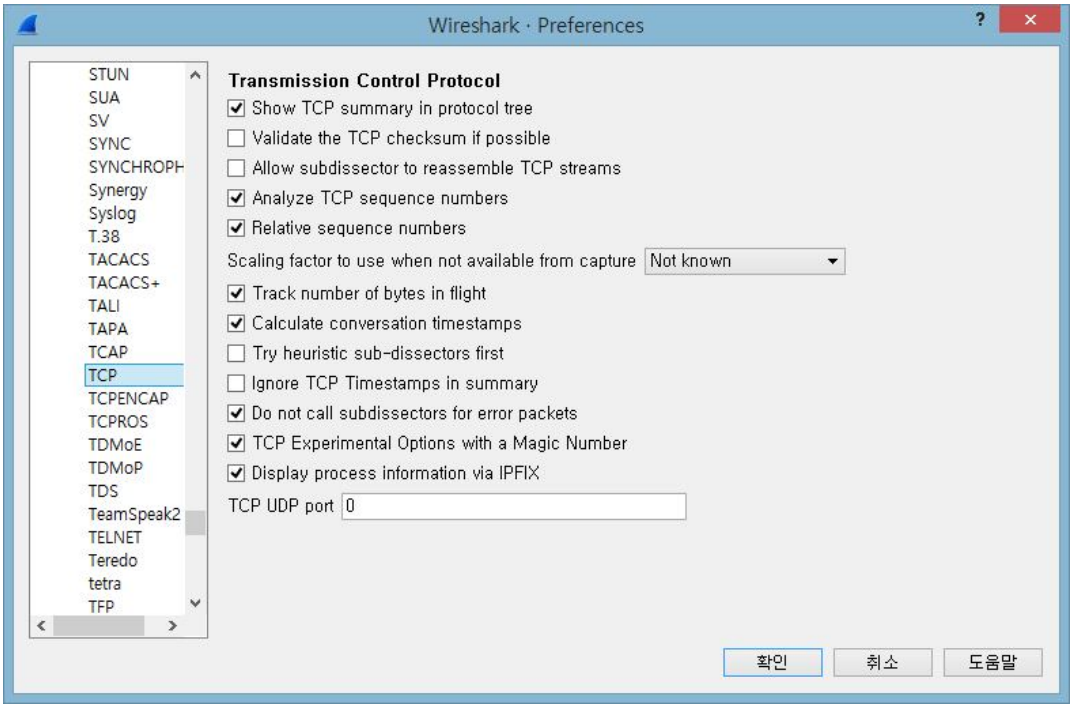


TCP 핸드셰이크를 이용한 왕복 시간 계산

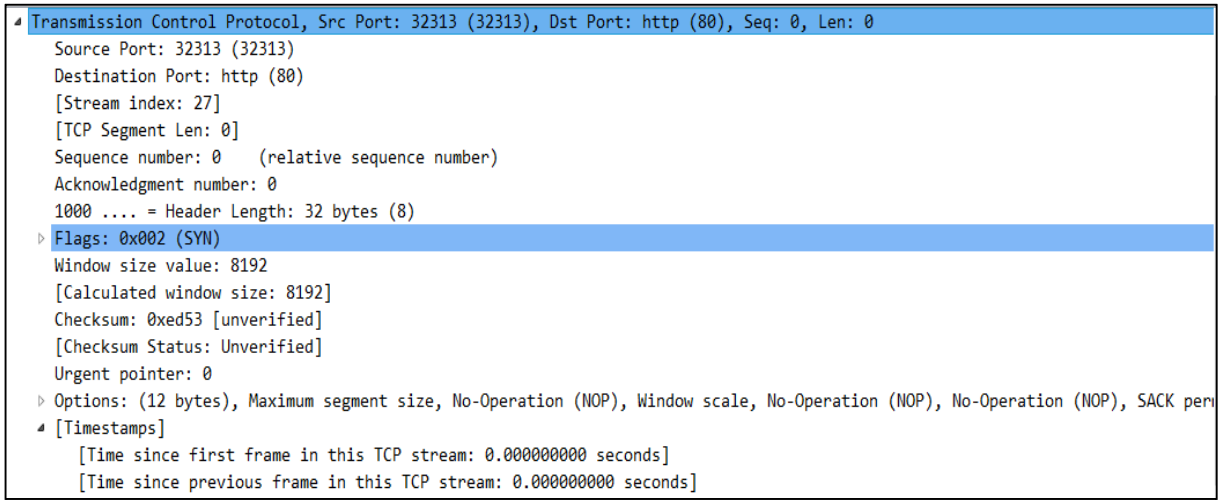
- 성능 문제의 원인이 경로 대기 시간인지 확인하는 것이 중요

(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

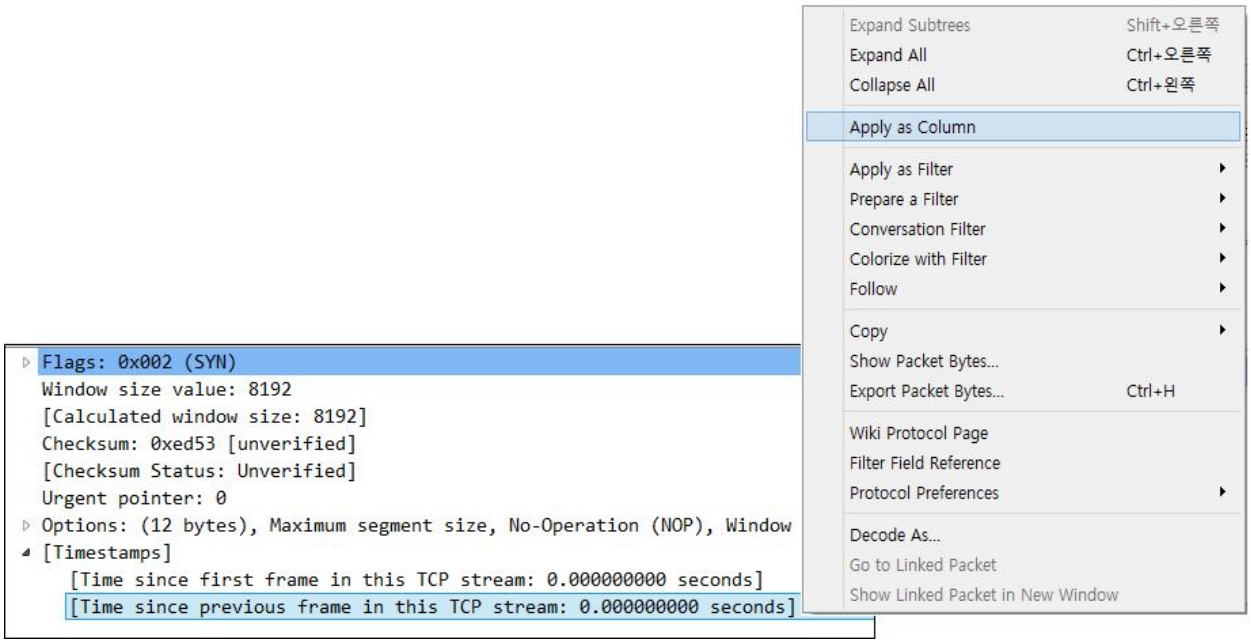


tcp.time_relative : 현재 TCP 스트림의 첫 프레임으로부터의 시간

tcp.time_delta : 현재 TCP 스트림의 이전 프레임으로부터의 시간

③ [TCP 시간차 칼럼 추가]

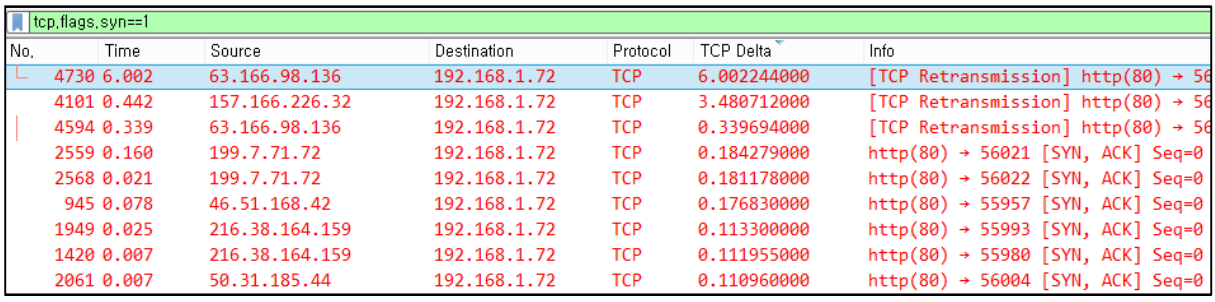
Time Since previous frame in this TCP Stream > 오른쪽 클릭 > Apply as Column 선택



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

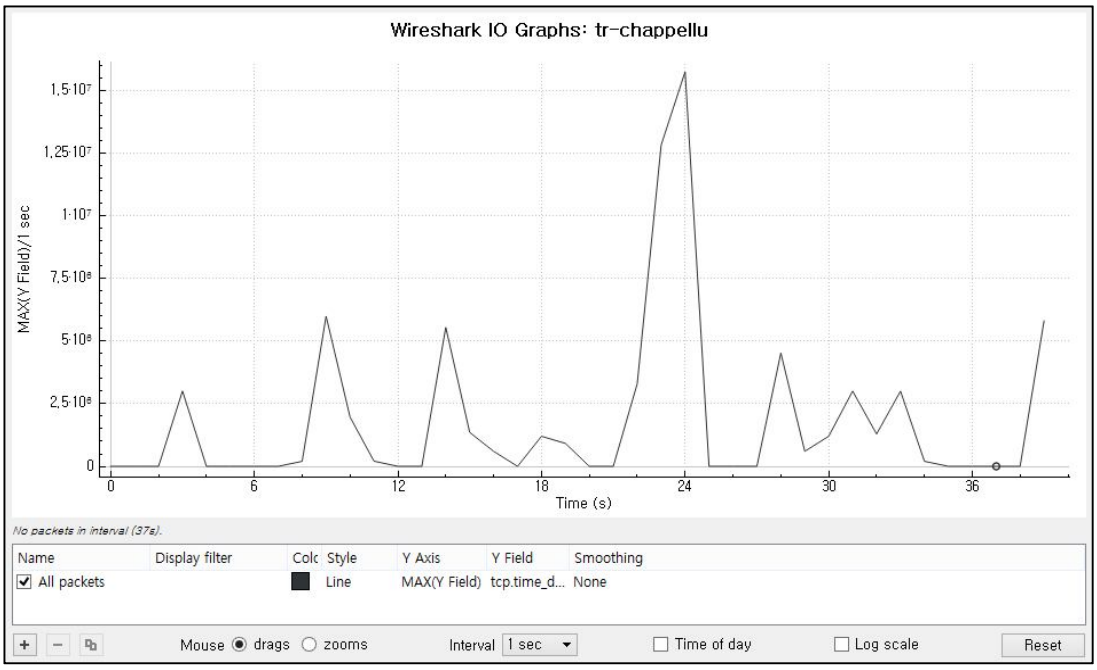


연결 설정 시 문제가 발생되는 것을 파악 할 수 있음

TCP 지연 그래프

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

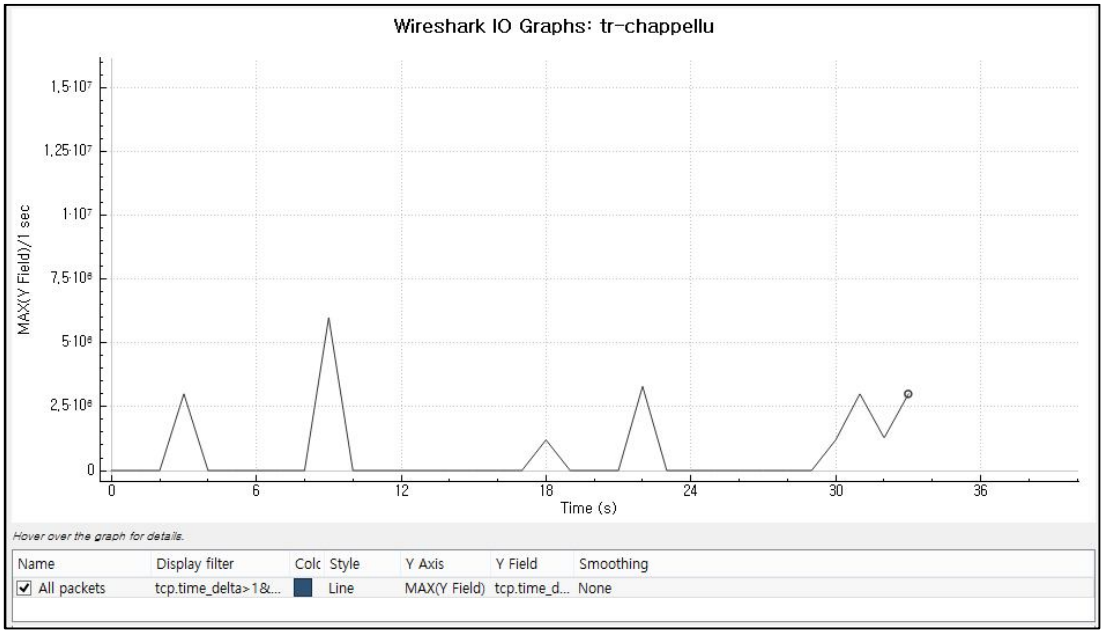
① Statistics > I/O Graph > Y 필드 축 지정(tcp.time_delta / MAX(delta))



⇨ 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

느린 DHCP 서버 응답 그래프

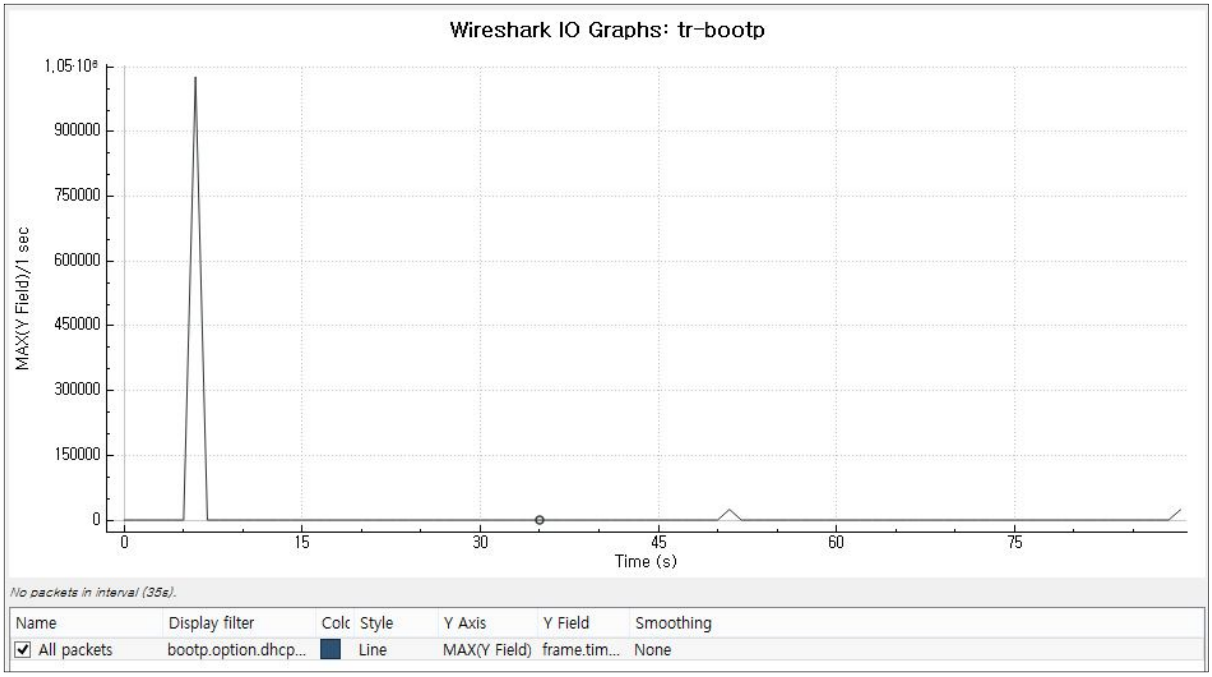
(UDP 기반 애플리케이션)

- 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)

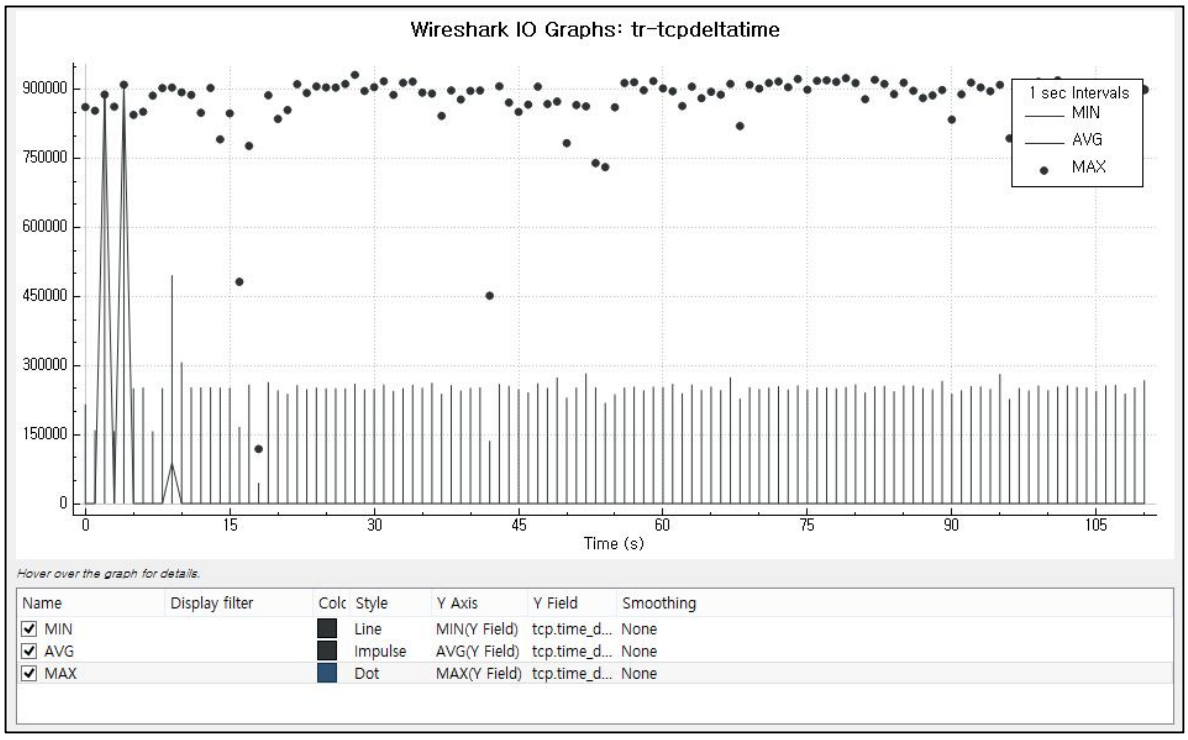
① [TCP 시간차그래프 생성] Statistics > I/O Graph > Y Filed (tcp.time_delta)

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

tcp.time_delta / MIN / LINE

tcp.time_delta / AVG/ Impulse

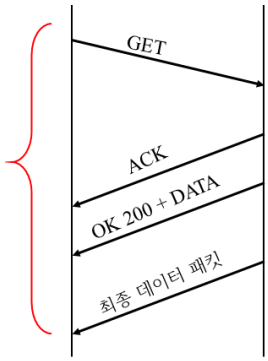
tcp.time_delta / MAX / Dot



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

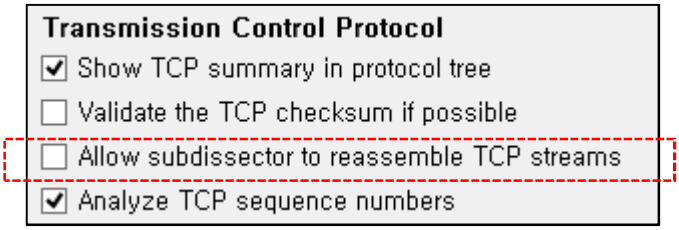
HTTP 응답 시간

- 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



Allow Sub-dissector to reassemble TCP stream' 활성화

- HTTP 요청으로부터 응답의 최종 데이터 패킷까지의 시간 측정
- 어떤 요소를 다운로드하기까지의 시간에 관심

Allow Sub-dissector to reassemble TCP stream' 바활성화

- 서버가 요청에 얼마나 빠른 응답이 가능하는지 확인 시 사용
- HTTP 요청으로부터 실제 응답 패킷까지의 시간을 측정

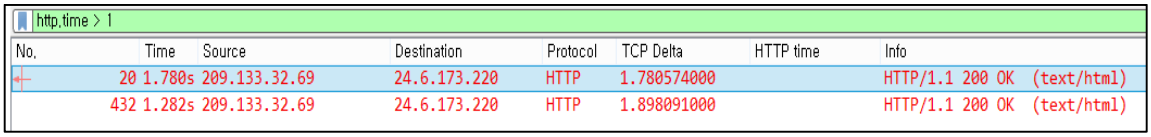
긴 HTTP 응답 시간 조회

- 추적 파일 tr-http-pcaprnet101 열기

❶ 'Allow Sub-dissector to reassemble TCP stream' 비활성

❷ 1초보다 긴 HTTP, 응답 시간 탐지

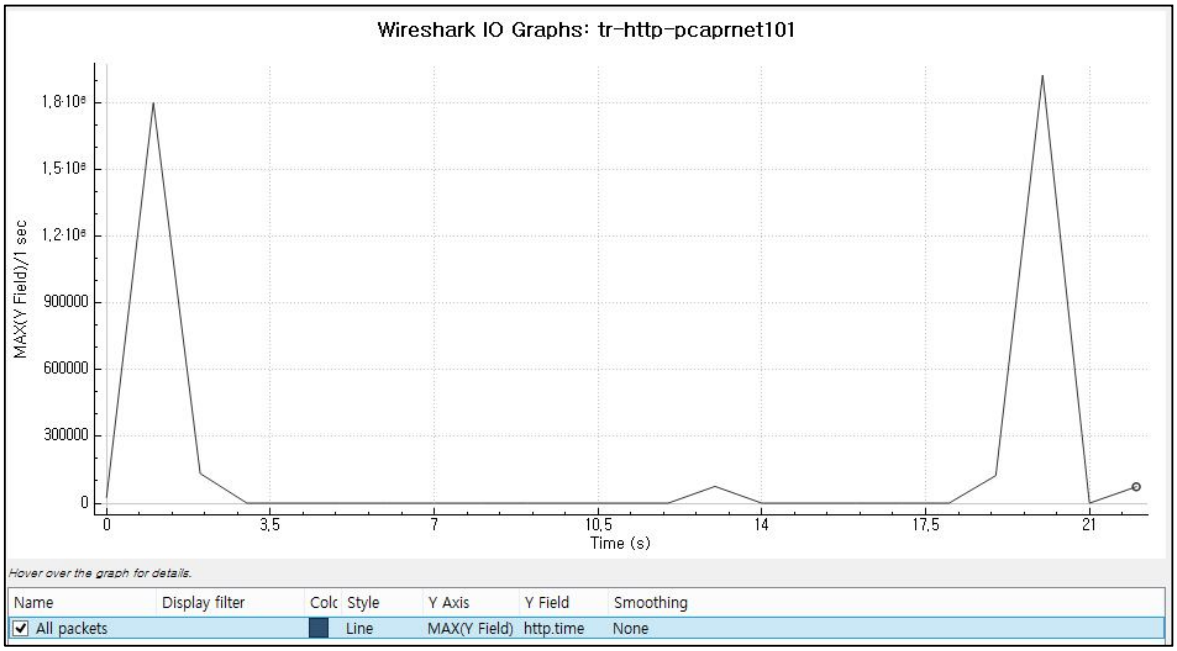
[디스플레이 필터] http.time > 1



❸ 필터해제

❹ HTTP 응답 시간 그래프 생성

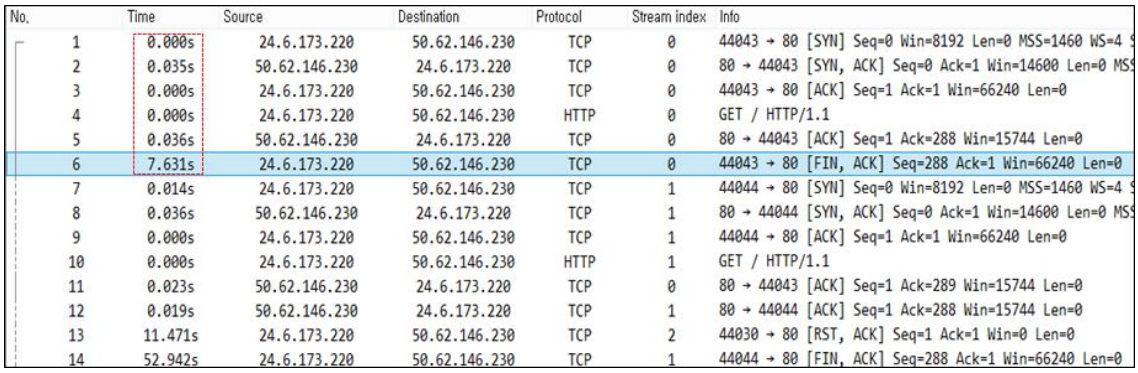
Statistics > I/O Graph > Y Filed (http.time)



⇨ 두 번의 긴 응답시간이 있었음을 확인

서비스 요청 무응답

- 추적 파일 tr-serverresponse 열기
  - 'Allow Sub-dissector to reassemble TCP stream' 비활성



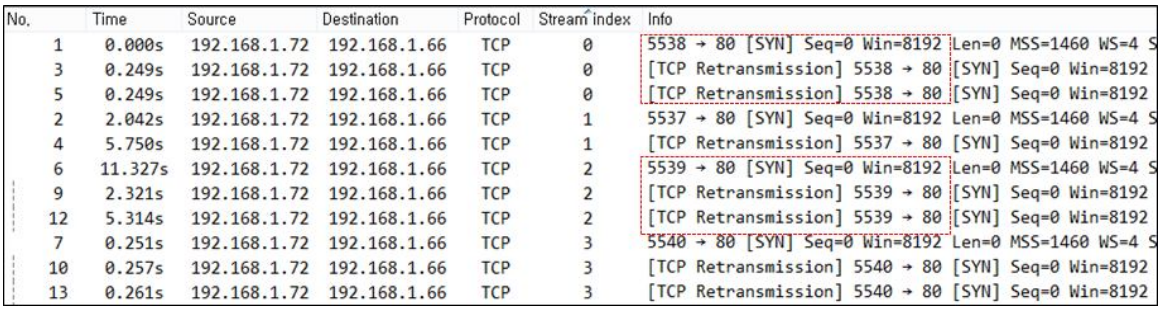
No.	Time	Source	Destination	Protocol	Stream index	Info
1	0.000s	24.6.173.220	50.62.146.230	TCP	0	44043 → 80 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=4 S
2	0.035s	50.62.146.230	24.6.173.220	TCP	0	80 → 44043 [SYN, ACK] Seq=0 Ack=1 Win=14600 Len=0 MS
3	0.000s	24.6.173.220	50.62.146.230	TCP	0	44043 → 80 [ACK] Seq=1 Ack=1 Win=66240 Len=0
4	0.000s	24.6.173.220	50.62.146.230	HTTP	0	GET / HTTP/1.1
5	0.036s	50.62.146.230	24.6.173.220	TCP	0	80 → 44043 [ACK] Seq=1 Ack=288 Win=15744 Len=0
6	7.631s	24.6.173.220	50.62.146.230	TCP	0	44043 → 80 [FIN, ACK] Seq=288 Ack=1 Win=66240 Len=0
7	0.014s	24.6.173.220	50.62.146.230	TCP	1	44044 → 80 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=4 S
8	0.036s	50.62.146.230	24.6.173.220	TCP	1	80 → 44044 [SYN, ACK] Seq=0 Ack=1 Win=14600 Len=0 MS
9	0.000s	24.6.173.220	50.62.146.230	TCP	1	44044 → 80 [ACK] Seq=1 Ack=1 Win=66240 Len=0
10	0.000s	24.6.173.220	50.62.146.230	HTTP	1	GET / HTTP/1.1
11	0.023s	50.62.146.230	24.6.173.220	TCP	0	80 → 44043 [ACK] Seq=1 Ack=289 Win=15744 Len=0
12	0.019s	50.62.146.230	24.6.173.220	TCP	1	80 → 44044 [ACK] Seq=1 Ack=288 Win=15744 Len=0
13	11.471s	24.6.173.220	50.62.146.230	TCP	2	44030 → 80 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
14	52.942s	24.6.173.220	50.62.146.230	TCP	1	44044 → 80 [FIN, ACK] Seq=288 Ack=1 Win=66240 Len=0

단계 1. 3 way handshake [1/2/3]
단계 2. 클라이언트가 서버에게 서비스 요청 (http request) [4]
단계 3. 클라이언트 요청의 수신 확인 ACK 송신 [5]
단계 4. 클라이언트 요청에 대한 서버의 처리 형태 정보 송신 없음
단계 5. 클라이언트는 대략 8초 정도 대기 후에 FIN/ACK를 송신하여 서버와의 연결 해제를 요청[6]
단계 6. 반복적으로 동일한 패턴 발생 [7,8,9, 10, 12]

- [결론] ① 네트워크 통신의 순간적인 결합이나 서버 실행 서비스의 일회성 문제가 아님
  ② TCP는 정상적으로 작동
  ③ 서버 쪽 애플리케이션의 작동 상태 파악 필요

TCP 연결 요청 무응답

- 추적 파일 tr-noserver 열기
  - Stream index 칼럼을 추가 후 정렬



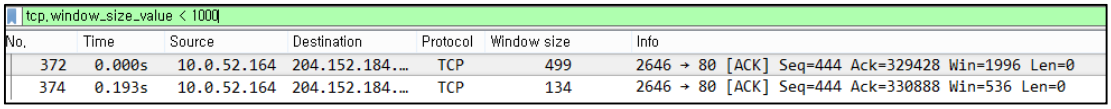
No.	Time	Source	Destination	Protocol	Stream index	Info
1	0.000s	192.168.1.72	192.168.1.66	TCP	0	5538 → 80 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=4 S
3	0.249s	192.168.1.72	192.168.1.66	TCP	0	[TCP Retransmission] 5538 → 80 [SYN] Seq=0 Win=8192
5	0.249s	192.168.1.72	192.168.1.66	TCP	0	[TCP Retransmission] 5538 → 80 [SYN] Seq=0 Win=8192
2	2.042s	192.168.1.72	192.168.1.66	TCP	1	5537 → 80 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=4 S
4	5.750s	192.168.1.72	192.168.1.66	TCP	1	[TCP Retransmission] 5537 → 80 [SYN] Seq=0 Win=8192
6	11.327s	192.168.1.72	192.168.1.66	TCP	2	5539 → 80 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=4 S
9	2.321s	192.168.1.72	192.168.1.66	TCP	2	[TCP Retransmission] 5539 → 80 [SYN] Seq=0 Win=8192
12	5.314s	192.168.1.72	192.168.1.66	TCP	2	[TCP Retransmission] 5539 → 80 [SYN] Seq=0 Win=8192
7	0.251s	192.168.1.72	192.168.1.66	TCP	3	5540 → 80 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=4 S
10	0.257s	192.168.1.72	192.168.1.66	TCP	3	[TCP Retransmission] 5540 → 80 [SYN] Seq=0 Win=8192
13	0.261s	192.168.1.72	192.168.1.66	TCP	3	[TCP Retransmission] 5540 → 80 [SYN] Seq=0 Win=8192

단계 1. 3 way handshake를 위한 SYN 패킷 전송 [1]
단계 2. 요청에 대한 SYN/ACK를 받지 못해 재전송 요청 [3,5]
단계 3. 클라이언트는 포트를 바꿔가면 TCP 연결 시도를 반복적으로 수행 [스트림 1, 2, 3]

- [결론] TCP 핸드셰이크가 성공한 뒤에도 서버 응답 문제를 겪을 수 있음

윈도우 제로 발생

- 버퍼 크기가 0으로 떨어지면(제로 윈도우 조건발생) 호스트는 더 이상 데이터를 받을 수 없다.
- 어떤 경우에는 window size 값이 작아서 TCP 피어로 데이터 송신이 중단되기도 한다.
- 추적 파일 tr-winsize 열기
  - 디스플레이 필터 tcp.window_size < 1000 (윈도우 크기가 1000보다 작은 패킷 조회)



tcp.window_size_value < 1000

No.	Time	Source	Destination	Protocol	Window size	Info
372	0.000s	10.0.52.164	204.152.184.…	TCP	499	2646 → 80 [ACK] Seq=444 Ack=329428 Win=1996 Len=0
374	0.193s	10.0.52.164	204.152.184.…	TCP	134	2646 → 80 [ACK] Seq=444 Ack=330888 Win=536 Len=0

⇨ TCP 피어가 536 byte 이상을 송신큐에 넣었을 때 데이터 전송이 멈춘다.

⇨ 해결 방법은 윈도우 업데이트 패킷을 기다리는 수밖에 없다.

윈도우 업데이트

- 윈도우 업데이트는 호스트의 애플리케이션이 수신버퍼에서 데이터를 가져갔음을 의미
- 호스트로부터 이전 패킷보다 더 커진 window size 값을 가진 ACK 패킷을 받으면 윈도우 업데이트로 표시
- 윈도우 제로 발생 시 유일한 해결 방안은 윈도우 업데이트이다.
- 윈도우 업데이트 디스플레이 필터값

  tcp.analysis.window_update

  tcp.analysis.window_full

- 추적 파일 tr-winzero-print 열기

No.	Time	Source	Destination	Protocol	winsize	Info
34	0.000s	192.168.1.111	192.168.1.101	TCP	6144	59319 → 9100 [ACK] Seq=37337 Ack=1 Win=6144 Len=1460
35	0.000s	192.168.1.111	192.168.1.101	TCP	6144	59319 → 9100 [ACK] Seq=38797 Ack=1 Win=6144 Len=1460
36	0.000s	192.168.1.111	192.168.1.101	TCP	6144	[TCP Window Full] 59319 → 9100 [ACK] Seq=40257 Ack=1
37	0.205s	192.168.1.101	192.168.1.111	TCP	0	[TCP ZeroWindow] 9100 → 59319 [ACK] Seq=1 Ack=41717 W
38	0.322s	192.168.1.111	192.168.1.101	TCP	6144	[TCP ZeroWindowProbe] 59319 → 9100 [ACK] Seq=41717 Ac
39	0.003s	192.168.1.101	192.168.1.111	TCP	0	[TCP ZeroWindowProbeAck] [TCP ZeroWindow] 9100 → 5931
40	0.011s	192.168.1.101	192.168.1.111	TCP	16384	[TCP Window Update] 9100 → 59319 [ACK] Seq=1 Ack=4171
41	0.000s	192.168.1.111	192.168.1.101	TCP	6144	59319 → 9100 [ACK] Seq=41717 Ack=1 Win=6144 Len=1460
42	0.000s	192.168.1.111	192.168.1.101	TCP	6144	59319 → 9100 [ACK] Seq=43177 Ack=1 Win=6144 Len=1460
43	0.000s	192.168.1.111	192.168.1.101	TCP	6144	59319 → 9100 [ACK] Seq=44637 Ack=1 Win=6144 Len=1460
44	0.000s	192.168.1.111	192.168.1.101	TCP	6144	59319 → 9100 [ACK] Seq=46097 Ack=1 Win=6144 Len=1460
45	0.000s	192.168.1.111	192.168.1.101	TCP	6144	59319 → 9100 [ACK] Seq=47557 Ack=1 Win=6144 Len=1460
46	0.000s	192.168.1.111	192.168.1.101	TCP	6144	59319 → 9100 [ACK] Seq=49017 Ack=1 Win=6144 Len=1460
47	0.000s	192.168.1.111	192.168.1.101	TCP	6144	59319 → 9100 [ACK] Seq=50477 Ack=1 Win=6144 Len=1460
48	0.000s	192.168.1.111	192.168.1.101	TCP	6144	59319 → 9100 [ACK] Seq=51937 Ack=1 Win=6144 Len=1460
49	0.000s	192.168.1.111	192.168.1.101	TCP	6144	59319 → 9100 [ACK] Seq=53397 Ack=1 Win=6144 Len=1460
50	0.000s	192.168.1.111	192.168.1.101	TCP	6144	59319 → 9100 [ACK] Seq=54857 Ack=1 Win=6144 Len=1460
51	0.000s	192.168.1.111	192.168.1.101	TCP	6144	59319 → 9100 [ACK] Seq=56317 Ack=1 Win=6144 Len=1460
52	0.195s	192.168.1.101	192.168.1.111	TCP	324	9100 → 59319 [ACK] Seq=1 Ack=57777 Win=324 Len=0
53	0.242s	192.168.1.101	192.168.1.111	TCP	16708	[TCP Window Update] 9100 → 59319 [ACK] Seq=1 Ack=5777
54	0.000s	192.168.1.111	192.168.1.101	TCP	6144	59319 → 9100 [ACK] Seq=57777 Ack=1 Win=6144 Len=1460

- tcp.analysis.window_update || tcp.analysis.window_full

tcp.analysis.window_full || tcp.analysis.window_update

No.	Time	Source	Destination	Protocol	winsize	Info
36	0.000s	192.168.1.111	192.168.1.101	TCP	6144	[TCP Window Full] 59319 → 9100 [ACK] Se
40	0.011s	192.168.1.101	192.168.1.111	TCP	16384	[TCP Window Update] 9100 → 59319 [ACK]
53	0.242s	192.168.1.101	192.168.1.111	TCP	16708	[TCP Window Update] 9100 → 59319 [ACK]
79	3.560s	192.168.1.101	192.168.1.111	TCP	17356	[TCP Window Update] 9100 → 59319 [ACK]
92	0.253s	192.168.1.101	192.168.1.111	TCP	17520	[TCP Window Update] 9100 → 59319 [ACK]
104	0.000s	192.168.1.111	192.168.1.101	TCP	6144	[TCP Window Full] 59319 → 9100 [ACK] Se
110	0.118s	192.168.1.101	192.168.1.111	TCP	16384	[TCP Window Update] 9100 → 59319 [ACK]
148	0.310s	192.168.1.101	192.168.1.111	TCP	17356	[TCP Window Update] 9100 → 59319 [ACK]
161	0.101s	192.168.1.101	192.168.1.111	TCP	17520	[TCP Window Update] 9100 → 59319 [ACK]
173	0.000s	192.168.1.111	192.168.1.101	TCP	6144	[TCP Window Full] 59319 → 9100 [ACK] Se

- 전문자정보(export infos)로 윈도우 업데이트 패킷 조회

Severity	Summary	Group	Protocol	Count
Warning	TCP window specified by the receiver is now comple...	Sequence	TCP	12
36	[TCP Window Full] 59319 → 9100 [ACK] Seq=4025...			
104	[TCP Window Full] 59319 → 9100 [ACK] Seq=1220...			
173	[TCP Window Full] 59319 → 9100 [ACK] Seq=2037...			
250	[TCP Window Full] 59319 → 9100 [ACK] Seq=2855...			
317	[TCP Window Full] 59319 → 9100 [ACK] Seq=3672...			
386	[TCP Window Full] 59319 → 9100 [ACK] Seq=4490...			
453	[TCP Window Full] 59319 → 9100 [ACK] Seq=5308...			
520	[TCP Window Full] 59319 → 9100 [ACK] Seq=6125...			
594	[TCP Window Full] 59319 → 9100 [ACK] Seq=6943...			
661	[TCP Window Full] 59319 → 9100 [ACK] Seq=7760...			
727	[TCP Window Full] 59319 → 9100 [ACK] Seq=8578...			
782	[TCP Window Full] 59319 → 9100 [ACK] Seq=9235...			
Warning	TCP Zero Window segment	Sequence	TCP	31
37	[TCP ZeroWindow] 9100 → 59319 [ACK] Seq=1 Ac...			
39	[TCP ZeroWindowProbeAck] [TCP ZeroWindow] 910...			
105	[TCP ZeroWindow] 9100 → 59319 [ACK] Seq=1 Ac...			
107	[TCP ZeroWindowProbeAck] [TCP ZeroWindow] 910...			
109	[TCP ZeroWindowProbeAck] [TCP ZeroWindow] 910...			
174	[TCP ZeroWindow] 9100 → 59319 [ACK] Seq=1 Ac...			
176	[TCP ZeroWindowProbeAck] [TCP ZeroWindow] 910...			
178	[TCP ZeroWindowProbeAck] [TCP ZeroWindow] 910...			
180	[TCP ZeroWindowProbeAck] [TCP ZeroWindow] 910...			
182	[TCP ZeroWindowProbeAck] [TCP ZeroWindow] 910...			
184	[TCP ZeroWindowProbeAck] [TCP ZeroWindow] 910...			
251	[TCP ZeroWindow] 9100 → 59319 [ACK] Seq=1 Ac...			

패킷 손실

• 패킷 손실은 스위치, 라우터, 방화벽 등 전달 결정을 하는 연결장치에서 발생한다.

• 스위치 또는 라우터가 오버헤드되어 패킷 전달률을 따라가지 못하거나 단순히 장치 불량일 수 있다.

• 패킷 손실 복구 방법 2가지

- 재전송 타이머(retransmission timer)를 보고 판단

  재전송 시간초과 (RTO, Retransmission Timeout) 타이머 값 이내에 데이터 패킷이 확인 응답을 받지 못한 것을 송신자가 알게 되면 패킷을 재전송

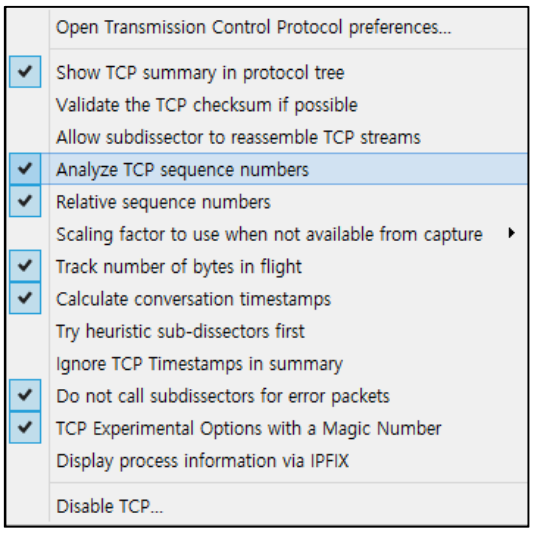
  재전송 타임 = { 이전 RTT * 0.9 + 현재 RTT *0.1 } *2

- 3개의 동일한 ACK가 연속적으로 들어오는 것을 보고 판단 (Duplicate ACK)

패킷 손실 위치 조회

• 추적 파일 tr-general101d 열기

• Analyze TCP sequence numbers 활성화



• Sequence Number, Next Sequence Number, Acknowledgement Number, Segment Length을 칼럼에 추가하여 번호 확인 상태 확인

No.	Time	Source	Destination	Protocol	SN	NextSN	AckN	Info
1	0.000s	10.9.9.9	10.10.10.10	TCP	1	1321	1	30000 → 1479 [ACK]
2	0.000s	10.10.10.10	10.9.9.9	TCP	1		1321	1479 → 30000 [ACK]
3	0.000s	10.9.9.9	10.10.10.10	TCP	1321	2641	1	30000 → 1479 [ACK]
4	0.000s	10.9.9.9	10.10.10.10	TCP	2641	3961	1	30000 → 1479 [ACK]
5	0.000s	10.10.10.10	10.9.9.9	TCP	1		3961	1479 → 30000 [ACK]
6	0.000s	10.9.9.9	10.10.10.10	TCP	3961	5281	1	30000 → 1479 [ACK]
7	0.000s	10.9.9.9	10.10.10.10	TCP	5281	6601	1	30000 → 1479 [ACK]
8	0.000s	10.10.10.10	10.9.9.9	TCP	1		6601	1479 → 30000 [ACK]
9	0.004s	10.9.9.9	10.10.10.10	TCP	6601	7921	1	30000 → 1479 [ACK]

No.	Time	Source	Destination	Protocol	SN	SegLen
1	0.000s	10.9.9.9	10.10.10.10	TCP	1	1320
2	0.000s	10.10.10.10	10.9.9.9	TCP	1	0
3	0.000s	10.9.9.9	10.10.10.10	TCP	1321	1320
4	0.000s	10.9.9.9	10.10.10.10	TCP	2641	1320
5	0.000s	10.10.10.10	10.9.9.9	TCP	1	0
6	0.000s	10.9.9.9	10.10.10.10	TCP	3961	1320
7	0.000s	10.9.9.9	10.10.10.10	TCP	5281	1320
8	0.000s	10.10.10.10	10.9.9.9	TCP	1	0
9	0.004s	10.9.9.9	10.10.10.10	TCP	6601	1320
10	0.000s	10.9.9.9	10.10.10.10	TCP	7921	1320

- 전문자 정보(export infos)를 이용해 패킷 손실 횟수 발견

Severity	Summary	Group	Protocol	Count
Warning	Previous segment not captured (common at capture start)	Sequence	TCP	5
4204	[TCP Previous segment not captured] 30000 → 1479 [ACK] Seq=369...			
10417	[TCP Previous segment not captured] 30000 → 1479 [ACK] Seq=917...			
11499	[TCP Previous segment not captured] 30000 → 1479 [ACK] Seq=990...			
15699	[TCP Previous segment not captured] 30000 → 1479 [ACK] Seq=141...			
32016	[TCP Previous segment not captured] 30000 → 1479 [ACK] Seq=275...			

⇨ 현재 5개의 패킷이 손실되었음을 의미

- 패킷 손실 지점 이후에 일어나는 현상을 관찰한다.

15843	0.000s	10.10.10.10	10.9.9.9	TCP	1479 → 30000 [ACK] Seq=1 Ack=13409881 Win=32768 Len=0 SL
15844	0.012s	10.9.9.9	10.10.10.10	TCP	30000 → 1479 [ACK] Seq=14254681 Ack=1 Win=46 Len=1320
15845	0.000s	10.10.10.10	10.9.9.9	TCP	[TCP Dup ACK 15843#1] 1479 → 30000 [ACK] Seq=1 Ack=1340
15846	0.000s	10.9.9.9	10.10.10.10	TCP	30000 → 1479 [ACK] Seq=14256001 Ack=1 Win=46 Len=1320
15847	0.000s	10.10.10.10	10.9.9.9	TCP	[TCP Dup ACK 15843#2] 1479 → 30000 [ACK] Seq=1 Ack=1340
15848	0.000s	10.9.9.9	10.10.10.10	TCP	30000 → 1479 [ACK] Seq=14257321 Ack=1 Win=46 Len=1320
15849	0.000s	10.10.10.10	10.9.9.9	TCP	[TCP Dup ACK 15843#3] 1479 → 30000 [ACK] Seq=1 Ack=1340
15850	0.000s	10.9.9.9	10.10.10.10	TCP	30000 → 1479 [ACK] Seq=14258641 Ack=1 Win=46 Len=1320
15851	0.000s	10.10.10.10	10.9.9.9	TCP	[TCP Dup ACK 15843#4] 1479 → 30000 [ACK] Seq=1 Ack=1340
15852	0.000s	10.9.9.9	10.10.10.10	TCP	30000 → 1479 [ACK] Seq=14259961 Ack=1 Win=46 Len=1320
15853	0.000s	10.10.10.10	10.9.9.9	TCP	[TCP Dup ACK 15843#5] 1479 → 30000 [ACK] Seq=1 Ack=1340

⇨ 패킷 손실 알림 이후 많은 중복 ACK [Duplicate ACK]를 송신하게 된다.

- 빠른 재전송(Fast Retransmission) 패킷 조회
  - 빠른 재전송은 빠른 복구 프로세서의 일부분이며 패킷 손실이 발생했다는 징후이다.
  - 패킷 빠른 재전송은 동일한 ACK 3개를 수신되면서 시작된다.
  - 전문가 정보에서 확인

Severity	Summary	Group	Protocol	Count
Note	This frame is a (suspected) fast retransmission	Sequence	TCP	2
12035	[TCP Fast Retransmission] 30000 → 1479 [ACK] Seq...			
12248	[TCP Fast Retransmission] 30000 → 1479 [ACK] Seq...			

작은 패킷 크기로 인한 지속적인 낮은 처리율 탐지

- 작은 패킷 크기로 파일을 전송 시 전송 지연을 유발한다.
- 작은 패킷 크기는 의도적으로 작은 크기의 데이터를 생성하는 애플리케이션 때문일 수도 있다.
- 또는 최대세그먼트크기(MSS, Maximum Segment Size) 설정에 의해 작은 패킷이 생성될 수 있다.
- 낮은 MSS는 클라이언트의 구성 오류 또는 VLAN 드라이버와 같은 다른 기능 때문일 수 있다.
- 추적파일 tr-throughput 열기
  - 호스트들 사이의 HTTPS 접속으로 구성
  - 암호화 해독할 키를 가지고 있지 않으므로 TCP 계층까지만 분석 할 수 있음
  - 크기가 작은 IP 패킷들이 많이 분포 되어 있는 것을 알 수 있음

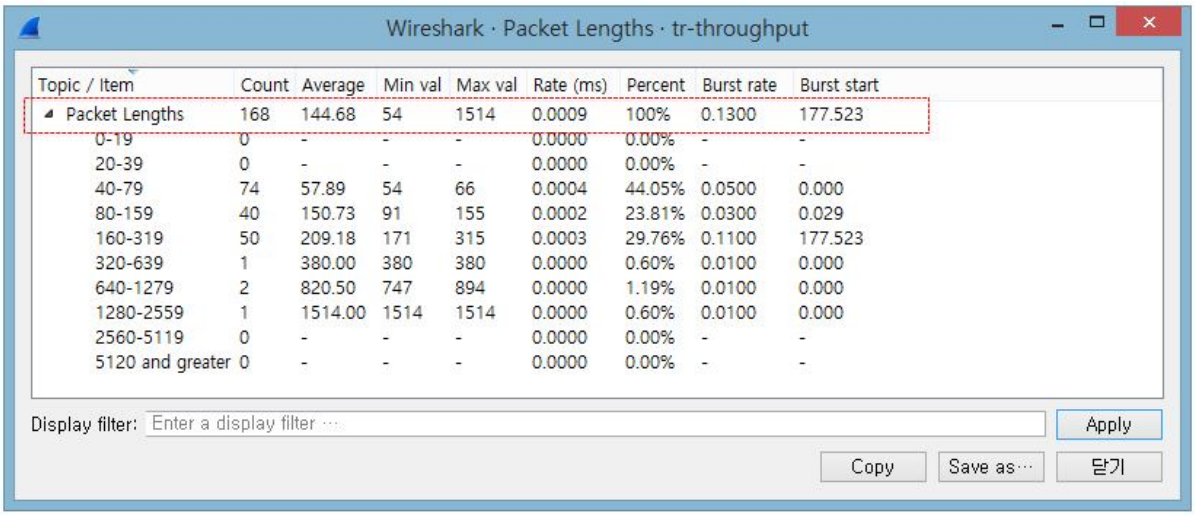
No.	Time	Source	Destination	Protocol	Total Length	Info
1	0.000s	24.6.173.220	216.115.209.…	TCP	52	44960 → 443 [SYN]
2	0.018s	216.115.209…	24.6.173.220	TCP	52	443 → 44960 [SYN,
3	0.000s	24.6.173.220	216.115.209.…	TCP	40	44960 → 443 [ACK]
4	0.009s	24.6.173.220	216.115.209.…	TLSv1	98	Client Hello
5	0.019s	216.115.209…	24.6.173.220	TCP	40	443 → 44960 [ACK]
6	0.002s	216.115.209…	24.6.173.220	TLSv1	1500	Server Hello
7	0.000s	216.115.209…	24.6.173.220	TLSv1	880	Ignored Unknown Re
8	0.000s	24.6.173.220	216.115.209.…	TCP	40	44960 → 443 [ACK]
9	0.001s	24.6.173.220	216.115.209.…	TLSv1	366	Client Key Exchang
10	0.032s	216.115.209…	24.6.173.220	TLSv1	274	New Session Ticket
11	0.000s	24.6.173.220	216.115.209.…	TLSv1	77	Application Data
12	0.020s	216.115.209…	24.6.173.220	TLSv1	77	Application Data
13	0.000s	24.6.173.220	216.115.209.…	TLSv1	173	Application Data
14	0.038s	216.115.209…	24.6.173.220	TLSv1	141	Application Data
15	0.002s	24.6.173.220	216.115.209.…	TLSv1	301	Application Data
16	0.021s	216.115.209…	24.6.173.220	TLSv1	733	Application Data
17	0.001s	24.6.173.220	216.115.209.…	TLSv1	301	Application Data
18	0.001s	216.115.209…	24.6.173.220	TLSv1	301	Application Data
19	0.000s	24.6.173.220	216.115.209.…	TLSv1	253	Application Data

  - MSS = 1,460 byte이므로 MSS 구성이 작은 패킷 크기의 원인은 아닌 것으로 추측

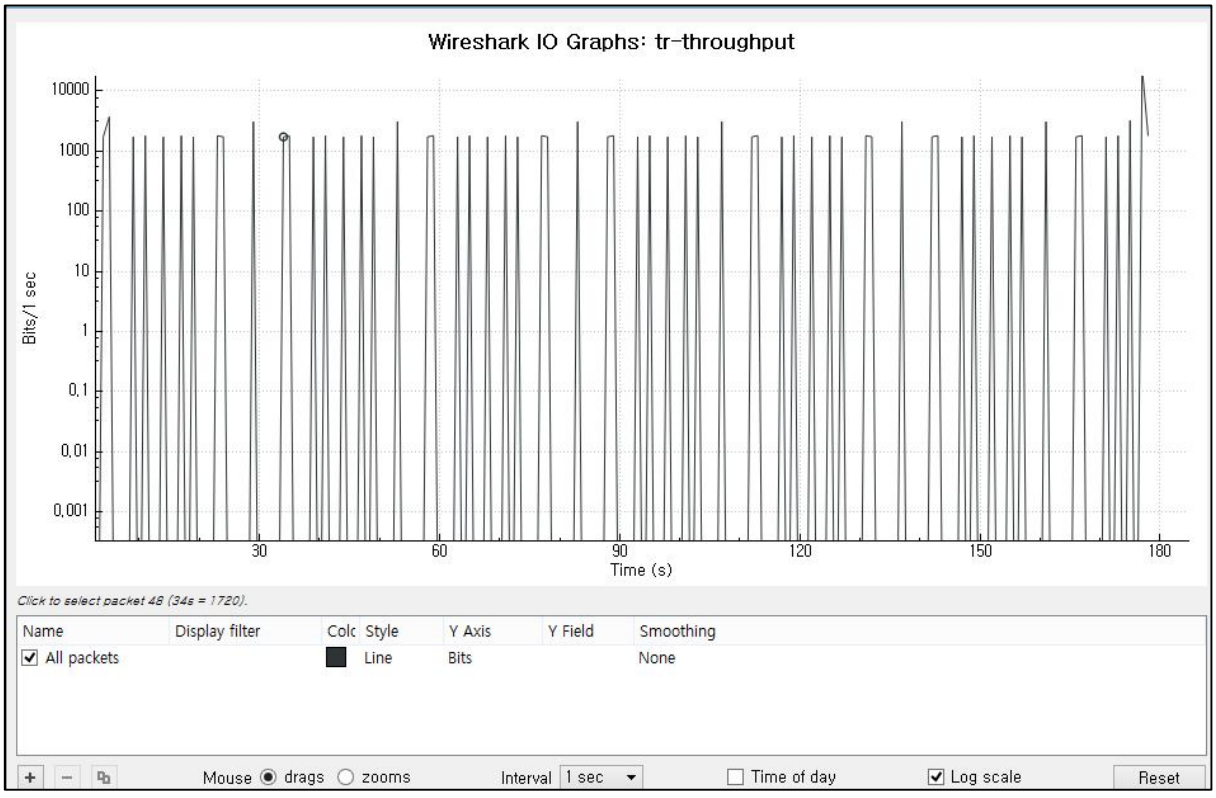
1	0.000s	24.6.173.220	216.115.209.…	TCP	52	44960 → 443 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=4 SACK_PERM=1
2	0.018s	216.115.209…	24.6.173.220	TCP	52	443 → 44960 [SYN, ACK] Seq=0 Ack=1 Win=5840 Len=0 MSS=1460 SACK_PERM=1 WS=128
3	0.000s	24.6.173.220	216.115.209.…	TCP	40	44960 → 443 [ACK] Seq=1 Ack=1 Win=65700 Len=0

- 추적파일의 평균 패킷 크기 확인

  Statistics > Packet Lengths



- 관련 그래프 생성 후 구체적인 상태 확인



⇨ 데이터 전송은 초당 1700~1900bit 정도의 값을 꾸준히 유지하고 있다.

⇨ TCP 연결에 의해 패킷 크기를 제약을 둔 것이 아니므로 해당 애플리케이션의 설정을 확인할 필요가 있다.

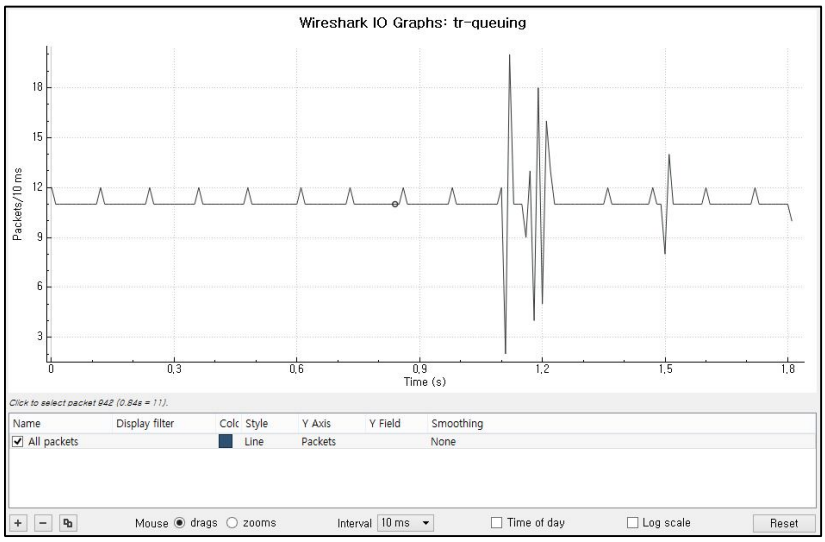
경로상의 큐 지연 식별

• 네트워크 장비들은 전달하는 패킷들을 큐에 넣어둠으로써 지연이 증가한다.

• 경로 상의 큐를 탐지하기 위해 트래픽 생성기를 사용한다.

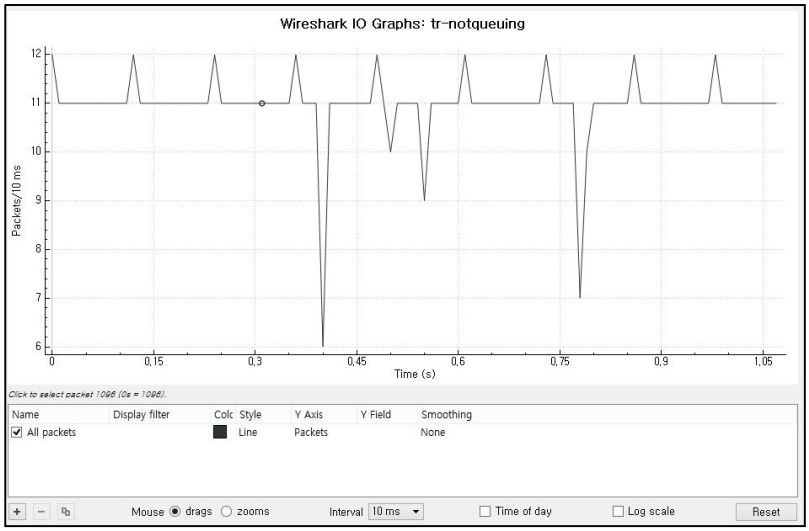
• 추적파일 tr-queuing 열기

- 초당 패킷 비율은 1,000이 넘기 때문에 다소 단조롭다.
- 0.01sec으로 변경하여 확인한다.



⇨ 트래픽이 일정한 비율로 전송되고 있으므로 트래픽률이 기준선을 설정할 수 있다. 처리율이 기준선 아래로 떨어진 다음 떨어진 만큼 기준선 위로 튀어 오르면 경로 위에 큐가 있다고 추론할 수 있다.

• 추적파일 tr-noqueuing 열기

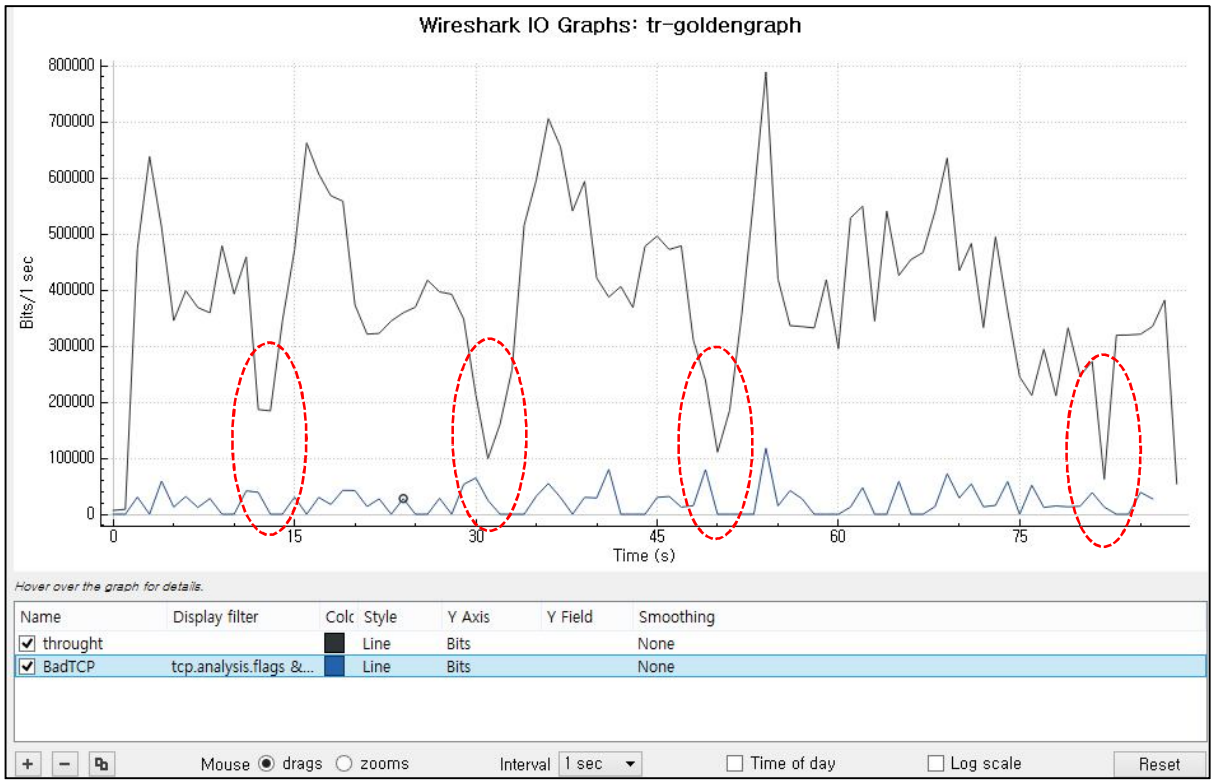


⇨ 그래프의 선이 기준선 아래로 떨어지지만 위로 튀어 오르지 않는 것을 알 수 있다. 이것은 패킷 손실을 의미한다 .UDP 전송 메커니즘은 재전송 기능이 없기 때문에 이 애플리케이션은 손실된 패킷을 탐지하거나 재전송 하지 않는다.

처리율 감소와 TCP 문제의 상관 관계 파악

- 추적파일 tr-goldengraph 열기
- 처리율 문제가 손실된 패킷이나 제로 윈도우 크기 등의 네트워크 문제와 관련이 있는지 확인 가능한 그래프이다.

- 클라이언트가 사이트에 접속하고 FTP로 파일을 다운로드할 때 캡쳐 한 것이다.



- 성능문제와 네트워크 문제의 연관 관계를 파악하기 위한 그래프이다.

  ⇨ 처리율이 거의 0까지 떨어지는 점이 4군데 존재한다. 이 추적 파일 전체에 걸쳐 Bad TCP 문제가 있지만 처리율이 떨어지는 지점에서는 bad tcp가 약간 증가하고 있다.