Nexus 9000(NX-OS)에서 TCP 성능 문제 해결

목차

소개

이 문서에서는 TCP 기본 사항, Wireshark 심층 패킷 분석, 엔드 투 엔드 성능을 최적화하는 실용적인 문제 해결에 대해 설명합니다.

사전 요구 사항

요구 사항

다음 주제에 대한 지식을 보유하고 있으면 유용합니다.

- IP/TCP

사용되는 구성 요소

이 문서의 정보는 다음 소프트웨어 및 하드웨어 버전을 기반으로 합니다.

- Cisco NX-OS 10.6(X)을 통해 Cisco Nexus 9000 클라우드 확장

참고: 타사 소프트웨어 또는 하드웨어의 구성 및 상호운용성에 대한 모든 질문은 Cisco 지원 범위에 포함되지 않습니다. 타사 툴을 사용하는 것은 Cisco 장비를 사용한 구성 및 운영을 입증하는 데 있어 최선의 노력입니다.

이 문서의 정보는 특정 랩 환경의 디바이스를 토대로 작성되었습니다. 이 문서에 사용된 모든 디바이스는 초기화된(기본) 컨피그레이션으로 시작되었습니다. 현재 네트워크가 작동 중인 경우 모든 명령의 잠재적인 영향을 미리 숙지하시기 바랍니다.

배경 정보

TCP란?

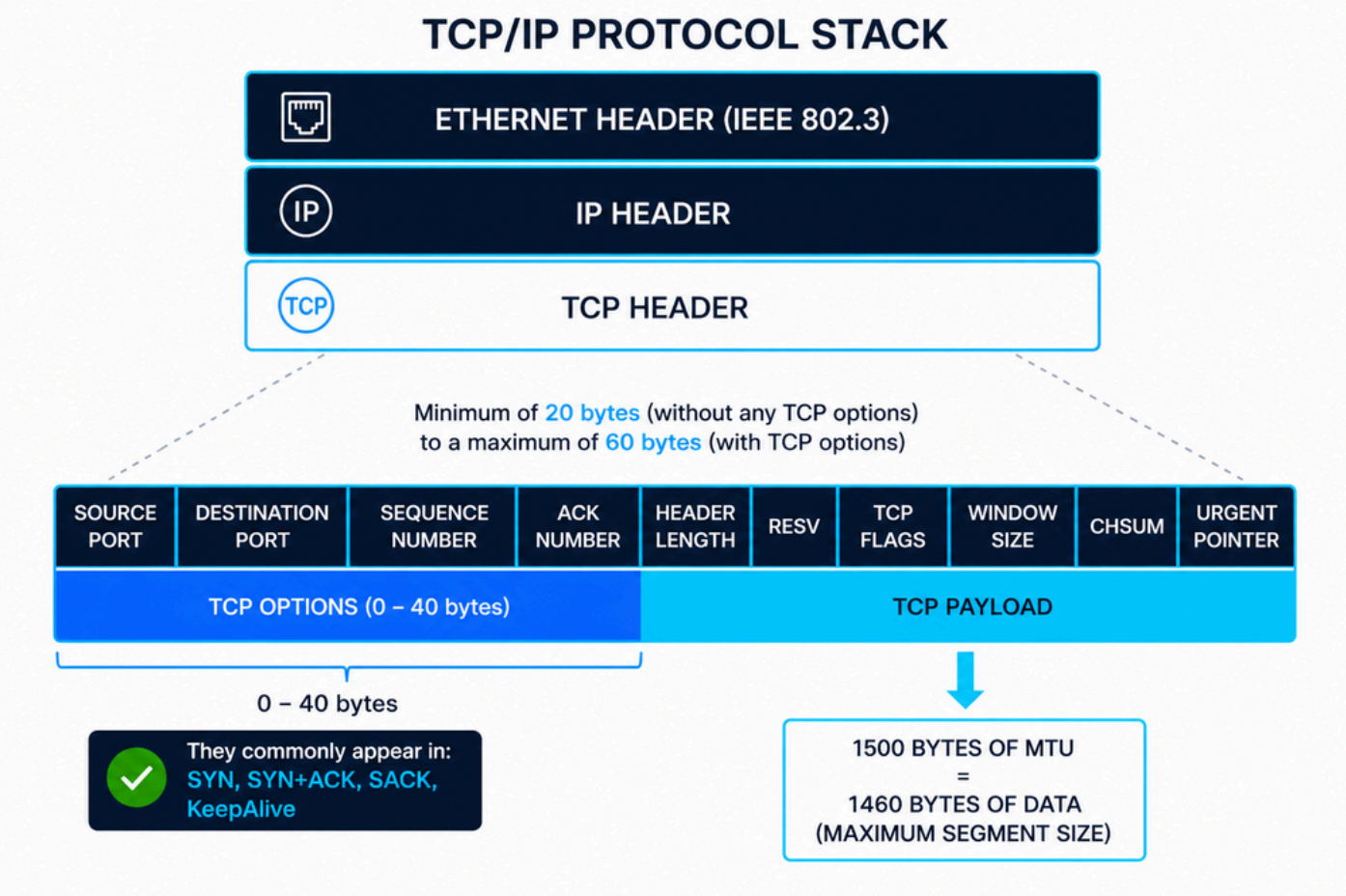
TCP(Transmission Control Protocol)는 OSI 모델의 레이어 4에서 작동하는 기본 전송 레이어 프로토콜로서, IP 네트워크를 통해 통신하는 애플리케이션 간에 바이트 스트림을 안정적이고 순서가 지정되었으며 오류가 확인된 방식으로 전달합니다.

세 가지 주요 이점

1. 신뢰성: TCP는 연결 지향적이며 수신기의 승인을 요구하여 전달을 보장합니다. 전송 중에 패킷이 손실되거나 손상된 경우, TCP는 대상에 도달할 수 있도록 데이터를 자동으로 다시 전송합니다.
2. 주문 배송: 네트워크 패킷이 순서를 벗어나서 도착할 수 있기 때문에 TCP는 각 세그먼트에 시퀀스 번호를 할당합니다. 이를 통해 수신 시스템은 원래 전송했던 정확한 순서대로 데이터를 재조합할 수 있습니다.
3. 흐름 및 혼잡 제어: TCP는 수신기 처리 용량 및 현재 네트워크 상태에 맞게 데이터 전송 속도를 동적으로 관리하여 버퍼 오버플로 또는 네트워크 혼잡으로 인한 데이터 손실을 방지합니다.

TCP/IP 캡슐화 개요

다이어그램은 TCP 세그먼트(레이어 4)가 IP 패킷(레이어 3) 내에 캡슐화된 다음 IEEE 802.3에 의해 정의된 이더넷 프레임(레이어 2) 내에 캡슐화되는 TCP/IP 스택을 나타냅니다. 이 계층화된 접근 방식은 모듈형 통신을 보장하며, 각 레이어는 고유의 제어 정보(헤더)를 추가하여 전달, 라우팅 및 데이터 무결성을 보장합니다.



이더넷 헤더(IEEE 802.3)

이더넷 헤더는 일반적으로 14바이트이며 다음으로 구성됩니다.

- 대상 MAC 주소(6바이트)
- 소스 MAC 주소(6바이트)
- 이더 타입/길이(2바이트)

또한 이더넷 프레임에는 레이어 2에서 오류 탐지를 위한 4바이트 FCS(Frame Check Sequence) 트레일러가 포함되어 있습니다. IEEE 802.3은 프레이밍, 최소/최대 프레임 크기, TCP와 같은 상위 레이어 프로토콜에 직접적인 영향을 주는 물리적 전달 제약 조건을 정의합니다.

IP 헤더(IPv4)

IPv4 헤더의 최소 크기는 20바이트이며, 옵션으로 최대 60바이트까지 확장 가능합니다. 주요 필드는 다음과 같습니다.

- 소스 및 대상 IP 주소

- TTL(Time To Live)
- 프로토콜(TCP를 페이로드로 식별)

IP 레이어는 네트워크에서 논리적 주소 지정 및 라우팅을 담당하지만, 신뢰성을 보장하지는 않습니다.

TCP 헤더 구조

TCP 헤더의 범위는 옵션에 따라 20~60바이트입니다. 주요 필드는 다음과 같습니다.

- 소스/대상 포트
- 시퀀스 번호
- 승인 번호
- 플래그(SYN, ACK, FIN, RST 등)
- 창 크기
- 체크섬

TCP는 IP 통신에 신뢰할 수 있는 전달, 적절한 시퀀싱 및 흐름 제어를 추가합니다.

TCP 옵션(일반 10)

TCP 옵션은 기본 프로토콜을 확장합니다. 가장 일반적인 내용은 다음과 같습니다.

1. MSS(Maximum Segment Size) - 호스트가 허용할 수 있는 최대 TCP 페이로드를 정의합니다.
2. Window Scale(창 크기) - 수신 창을 65,535바이트 이상으로 확장합니다.
3. Selective Acknowledgment Permitted (SACK Permitted) - 선택적 확인 응답 기능을 활성화합니다.
4. SACK(Selective Acknowledgment) - 전체 재전송을 방지하기 위해 수신된 데이터 블록을 지정합니다.
5. 타임스탬프 - RTT 계산 및 PAWS(Protection Against Wrapped Sequence Number)에 사용됩니다.
6. NOP(No-Operation) - 옵션 정렬을 위한 패딩
7. EOL(End of Option List) - TCP 옵션의 끝을 표시합니다.
8. TCP TFO(Fast Open) - 초기 핸드셰이크 중에 데이터 교환을 허용합니다.
9. Multipath TCP(MPTCP) - 단일 TCP 세션에 대해 여러 네트워크 경로를 활성화합니다.
10. User Timeout Option (UTO)(사용자 시간 초과 옵션(UTO)) - 전송된 데이터가 승인되지 않은 상태로 유지되는 기간을 제어합니다.

TCP 시퀀스 및 승인 동작(SYN/FIN 포함)

페이로드가 없는 경우에도 SYN 및 FIN 플래그는 각각 하나의 시퀀스 번호를 사용합니다. TCP는 바이트 지향 시퀀싱 모델을 사용하여 작동하며, 여기서 전송되는 모든 바이트 및 특정 제어 플래그는 시퀀스 공간을 진행합니다. 이 동작은 패킷 캡처의 정확한 TCP 분석과 시퀀싱 또는 승인 불일치를 진단하는 데 필수적입니다.

ACK = 시퀀스 + 페이로드 길이 + (SYN ? 1: 0) + (FIN ? 1: 0)

여기서 각 항목은 다음을 나타냅니다.

- 시퀀스 = 초기 시퀀스 번호
- 페이로드 길이 = 데이터 크기(바이트)
- SYN? 1: 0 = SYN 플래그가 설정되어 있으면 1을 추가하고 그렇지 않으면 0을 추가합니다.
- FIN? 1: 0 = FIN 플래그가 설정되어 있으면 1을 추가하고 그렇지 않으면 0을 추가합니다.
- ACK = 다음 예상 바이트

예 1: 데이터가 있는 SYN(TCP Fast Open)

- 순번 = 1000
- SYN = 1
- 페이로드 길이 = 200바이트

ACK 계산:

- ACK = 1000 + 200 + 1 + 0 = 1201

이는 TCP 핸드셰이크 중에 데이터가 전송되는 시나리오를 반영합니다. 페이로드와 SYN 플래그는 모두 시퀀스 공간을 소비합니다.

예 2: 데이터가 있는 FIN(연결 종료)

- 순번 = 3000
- FIN = 1
- 페이로드 길이 = 150바이트

ACK 계산:

- ACK = 3000 + 150 + 0 + 1 = 3151

이는 TCP가 연결 해제 중에 데이터를 포함할 수 있으며, 페이로드와 FIN 플래그 모두 시퀀스 번호를 증가시킨다는 것을 보여줍니다.

MSS 및 MTU와의 관계

MSS(Maximum Segment Size)는 TCP가 세그먼트에서 전송할 수 있는 최대 페이로드를 정의합니다.

- 일반적인 이더넷 MTU = 1500바이트
- MSS = MTU − IP 헤더 − TCP 헤더
- 표준 MSS = 1,460바이트(1,500−20−20)

TCP 옵션이 있으면 MSS가 그에 따라 감소합니다. MSS는 TCP 3-way 핸드셰이크 중에 협상되며 IP 레이어에서 프래그먼트화를 방지합니다.

TCP 3방향 핸드셰이크에서 MSS 협상이 작동하는 방식

TCP 3방향 핸드셰이크 중에 SYN 패킷의 MSS 옵션을 사용하여 MSS(Maximum Segment Size)를 교환합니다.

- 호스트 A → 호스트 B(SYN): MSS를 광고합니다(예: 1460).
- 호스트 B → 호스트 A(SYN-ACK): MSS를 광고합니다(예: 1380).

양측은 효과적으로 말하고 있습니다.

허용되는 최대 TCP 페이로드입니다.

주요 규칙: MSS는 방향

MSS는 하나의 합의된 가치로 협상되지 않습니다.

대신:

- 각 호스트는 다른 쪽에서 광고한 MSS를 사용합니다.
- 이렇게 하면 두 개의 독립적인 제한이 생성됩니다. 한 방향당 하나씩.

따라서

- A는 B의 MSS를 사용하여 데이터를 전송합니다.
- B는 A의 MSS를 사용하여 데이터를 전송합니다.

소스가 목적지 MSS보다 더 많은 TCP 페이로드를 전송할 수 있습니까?

올바르게 작동하는 TCP 스택에서: 아니요.

- 발신자는 수신자가 광고한 MSS를 존중해야 합니다.
- 더 큰 세그먼트를 보내는 것은 위험할 수 있습니다.
  - IP 프래그먼트화(MTU가 초과된 경우)
  - 패킷 삭제(조각화가 차단되었거나 지원되지 않는 경우)
- 그 결과 다음과 같은 결과를 낳게 됩니다.
  - 재전송
  - 성능 저하
  - PMTUD(Path MTU Discovery) 블랙홀과 같은 문제

문제 해결을 위한 실용적인 통찰력

- TCP 3방향 핸드셰이크(SYN/SYN-ACK 패킷)에서 항상 MSS 값을 확인합니다.
- 다음 원인으로 인한 불일치 확인:
  - 터널(VXLAN, GRE, IPsec)
  - MSS(MSS 클램핑) 수정 방화벽
- Cisco NX-OS와 같은 플랫폼에서는 캡슐화된 경로 간의 프래그먼트화를 방지하기 위해 MSS 조정이 자주 사용됩니다

창 크기(흐름 제어)

Window Size(윈도우 크기)는 수신자가 승인 없이 허용할 수 있는 데이터의 양을 정의합니다.

정의:

- 버퍼 오버플로를 방지하기 위한 흐름 제어 메커니즘.

목적:

- 발신자가 수신자를 압도하지 않도록 합니다.

얻을 수 있는 위치:

- 패킷 캡처(예: Wireshark)에 표시됩니다.
- OS TCP 스택 컨피그레이션 및 버퍼 크기에서 파생됩니다.

공급업체/OS 가변성:

- 다양한 구현(Linux, Windows, Cisco NX-OS)에서 동적 확장 및 버퍼 조정을 사용하므로 윈도우 크기가 다양합니다.

제로 윈도우 조건:

- Window Size = 0이면 수신기 버퍼가 꽉 찼습니다.
- 발신자는 전송을 일시 중지하고 주기적인 프로브를 전송합니다.

가변 Windows 메커니즘

- 속도 기반 흐름 제어
  - 발신자에게 고정 데이터 전송률을 할당하고 데이터가 해당 할당량을 초과하지 않도록 합니다.
  - 스트리밍 애플리케이션에 이상적입니다.
  - 브로드캐스트 및 멀티캐스트 전달
- 창 기반 흐름 제어
  - 창 크기는 시간에 따라 달라집니다.
  - 수신기는 허용된 윈도우를 발신자 윈도우 업데이트로 시그널링함으로써 흐름 제어를 달성한다.

문제 해결 용도:

- 소형 또는 제로 윈도우→ 수신기 측 병목 현상(CPU, 메모리, 애플리케이션).
- 네트워크 문제(레이턴시, 혼잡)→ 처리량이 적지만 큰 윈도우가 있습니다.
- 창 동작을 분석하는 것은 TCP 세션의 성능 문제를 진단하는 데 매우 중요합니다.

Cisco Nexus 9000(NX-OS)의 TCP 데이터 플레인 문제 해결

이 섹션에서는 NX-OS를 실행하는 Cisco Nexus 스위치가 TCP 트래픽 포워딩에 영향을 미치거나 성능 문제를 일으키는지 여부를 진단하기 위한 실용적인 방법론에 대해 설명합니다. 그 접근은 가상의 시나리오를 통해 제시된다.

TCP 지연 시간 또는 성능 저하가 관찰될 경우, 처음에는 네트워크가 문제를 일으키고 있다고 생각하는 것이 일반적입니다. 그러나 이 가정은 데이터 기반 분석을 통해 검증해야 합니다. TCP 트러블슈팅에 대한 신뢰할 수 있는 방법은 패킷 캡처이며, 이상적으로 수행됩니다.

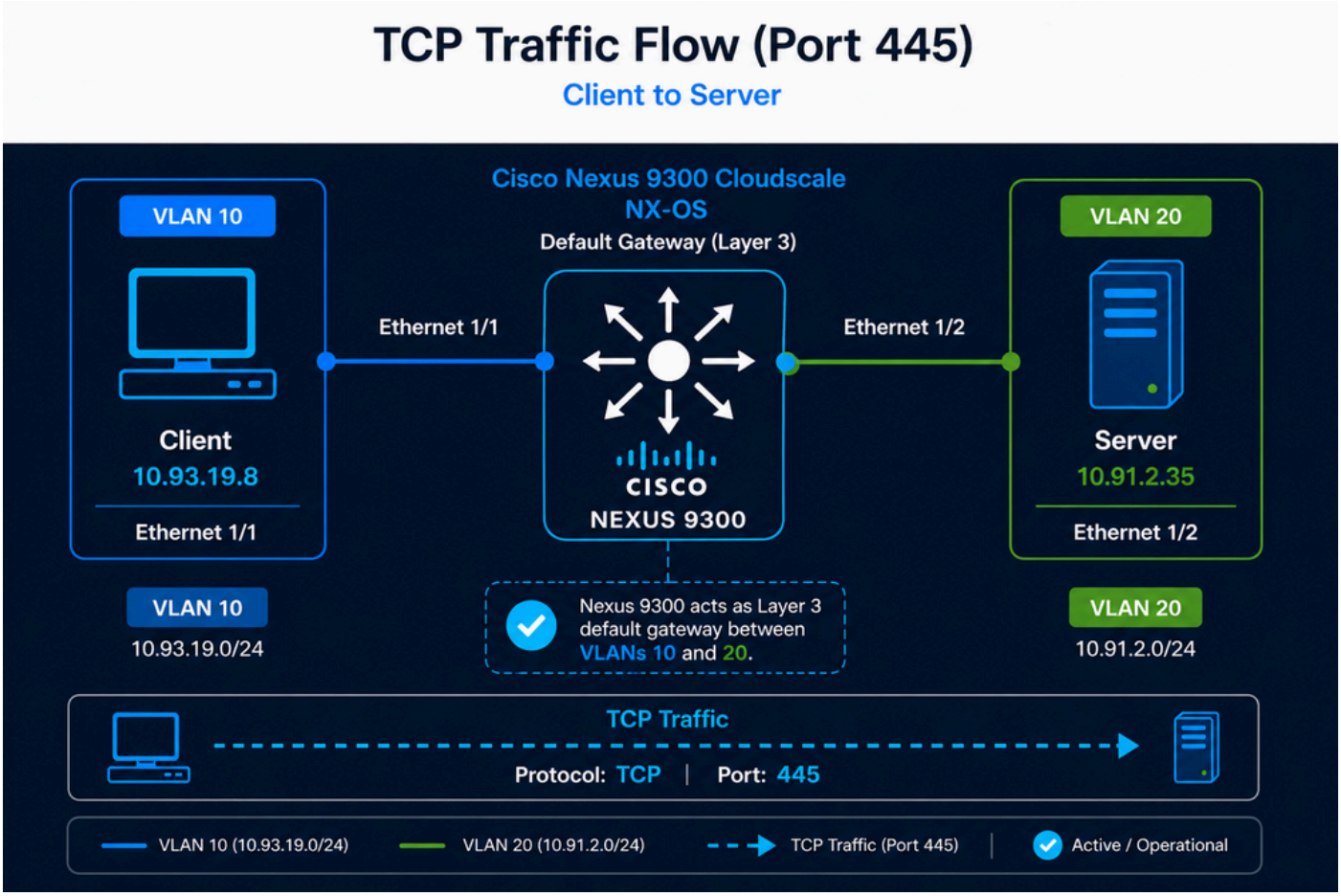
- 소스와 대상에서 동시에
- 트래픽 시작 전

이렇게 하면 MSS, Window Scale 및 SACK와 같은 중요한 매개 변수가 협상되고 세션 후반부에서 반복되지 않는 TCP 3방향 핸드셰이크에 대한 가시성이 확보됩니다. 동시 캡처가 불가능한 경우 단일 캡처를 사용하여 분석을 진행할 수 있지만 결론은 제한됩니다.

시나리오 정의

한 사용자가 이전에 약 9시간 만에 완료된 약 7.5TB의 애플리케이션 데이터 세트에 대한 백업 프로세스가 이제 거의 21시간이 걸린다는 사실을 확인했습니다. 클라이언트와 서버 간의 TCP 세션이 여전히 성공적으로 설정되지만 백업 기간이 크게 증가하면 처리량 또는 전반적인 TCP 성능이 저하될 수 있습니다. Nexus 스위치는 경로에 있는 유일한 네트워크 디바이스이며 레이어 3 게이트웨이 기능도 제공하므로 네트워크 관리자는 Nexus 스위치가 문제의 원인이라고 의심합니다.

- 클라이언트: 10.93.19.8(VLAN 10)
- 서버: 10.91.2.35(VLAN 20)
- 기본 게이트웨이 역할을 하는 Nexus 9300
- TCP 포트 445



초기 검증(연결 가능성)

- 이 명령은 DF(Don't Fragment) 비트가 설정된 ICMP 패킷을 전송하여 소스와 대상 간의 경로 MTU(PMTU)를 검증하는 데 사용됩니다. 조각화 없이 네트워크를 이동할 수 있는 최대 패킷 크기를 결정하는 데 도움이 됩니다. 이 프로세스는 소스와 대상 모두에서 수행해야 합니다.
- 소스와 대상 모두에서 항상 물리적 인터페이스의 MTU를 확인합니다.
- 이 시나리오에서는 MTU 1500이 식별된 소스 호스트에만 액세스할 수 있습니다.

```
Linux: ping -c 10 -I 10.93.19.8 -s 1472 -M do 10.91.2.35
```

- -c 10 → 10개의 ICMP 에코 요청을 보냅니다.
- -I 192.168.10.10 → 이 특정 소스 IP/인터페이스를 사용합니다.
- -s 1472 → ICMP 페이로드 크기를 1472바이트로 설정합니다.
- -M do → DF(Don't Fragment) 비트를 설정합니다
- 192.168.20.20 → 대상 IP

```
Windows: ping -n 10 -l 1472 -f 10.91.2.35
```

- -n 10 → 10개의 ICMP 에코 요청을 보냅니다.
- -l 1472 → ICMP 페이로드 크기를 1472바이트로 설정합니다.
- -f → DF(Don't Fragment) 플래그를 설정합니다
- 192.168.20.20 → 대상 IP

왜 1472바이트일까요?

- ICMP 페이로드 = 1472바이트
- IP 헤더 = 20바이트
- ICMP 헤더 = 8바이트
- 총 패킷 크기: 1472 + 20 + 8 = 1500바이트(표준 MTU)
- 이 테스트에서는 경로가 단편화 없이 1500바이트 MTU를 지원하는지 여부를 테스트합니다. 1500바이트의 ICMP 페이로드를 전송하려고 하면 총 패킷 크기가 IP 및 ICMP 헤더를 추가한 후 표준 MTU를 초과하므로 ping이 실패할 수 있습니다.

결론 내릴 수 있는 것

- Ping이 성공하면(패킷 손실 없음) 경로는 1500바이트 이상의 MTU를 지원하며 단편화가 필요

하지 않습니다.
    - TCP 분석으로 → ICMP 결과 정리
    - 간헐적인 ping 성공→ 패킷 손실, 일시적인 정체, 속도 제한 또는 포워딩 문제가 발생할 수 있습니다. tcp가 효율적으로 작동하려면 손실이 없는 경로가 필요하므로 패킷 손실 분석으로 진행합니다.
- Ping이 실패하고 "Fragmentation needed(조각화가 필요함)" 오류가 발생하거나 시간이 초과되면 경로에 MTU가 1500바이트보다 낮은 링크가 있으며, DF 비트 때문에 패킷을 전달할 수 없으며, 이는 경로 MTU 문제를 나타냅니다.

트러블슈팅에 이 방법을 사용하는 방법

- 페이로드 크기를 점진적으로 줄여(예: 1472→1400→1300) 성공한 최대 크기를 식별합니다.
- 식별되면 MTU = 페이로드 + 28바이트(IP + ICMP 헤더)라는 공식을 사용하여 MTU를 계산합니다.

TCP와의 실질적 관련성

- MTU가 예상보다 작으면 TCP 세그먼트가 조각화되거나 삭제될 수 있습니다.
- 이는 재전송, 레이턴시 증가, 처리량 감소로 이어지며 애플리케이션 성능에 직접적인 영향을 미칩니다.

트래픽 경로 식별(인터페이스)

Cisco Nexus 9000 스위치에서 TCP 성능의 문제를 효과적으로 해결하려면 소스와 대상 간에 어떤 인터페이스에서 트래픽을 수신하고 포워딩하는지 확인해야 합니다.

단순 토폴로지에서는 물리적 연결에서 직접 유추할 수 있습니다. 예를 들어 클라이언트가 Ethernet1/1에 연결되고 서버가 Ethernet1/2에 연결된 경우 트래픽 경로가 간단합니다. 그러나 여러 활성 인터페이스, 포트 채널 또는 vPC 컨피그레이션이 있는 실제 환경에서는 이러한 식별이 항상 간단한 것은 아닙니다.

그러한 경우 ASIC(데이터 플레인 하드웨어) 레벨에서 가시성을 제공하는 ELAM(Embedded Logic Analyzer Module)을 사용하는 것이 좋습니다.

ELAM에서는 포워딩 파이프라인에서 처리되는 패킷을 캡처하고 다음과 같은 중요한 정보를 표시할 수 있습니다.

- 인그레스 인터페이스
- 이그레스 인터페이스
- 전달 결정(L2/L3 조회 결과)

이 방법은 실제 하드웨어 포워딩 경로를 반영하므로 컨트롤 플레인 툴에 의존하는 것보다 훨씬 정확합니다.

ELAM은 한 번에 하나의 패킷만 캡처하므로 원하는 트래픽(예: 소스 IP, 목적지 IP, TCP 포트)과 일치하도록 필터링 기준을 정확하게 정의해야 합니다. 필터가 너무 광범위하면 의도한 TCP 흐름 대신 ICMP 또는 UDP와 같은 관련 없는 트래픽을 캡처할 위험이 있습니다.

또한 두 트래픽 방향 모두에 대해 이 프로세스를 반복해야 합니다.

- 소스 → 대상
- 대상 → 원본

vPC 또는 ECMP를 사용하는 환경에서는 여러 경로에서 트래픽을 로드 밸런싱할 수 있습니다. 따라서 전달 및 반환 트래픽이 서로 다른 스위치 또는 인터페이스를 통과할 수 있습니다. 이러한 시나리오에서는 완벽한 가시성을 보장하기 위해 각 관련 Nexus 스위치에서 ELAM을 실행해야 합니다.

인그레스(ingress) 및 이그레스(egress) 인터페이스를 정확하게 식별하여 문제 해결 범위를 크게 줄임으로써 정확한 포워딩 경로에 따라 인터페이스 카운터, QoS 정책, MTU 설정 및 잠재적 정체 지점을 집중적으로 검증할 수 있습니다.

ELAM 컨피그레이션(Nexus 9300 클라우드 확장)

이 예에서는 소스 IP 10.93.19.8, 목적지 IP 10.91.2.35 및 TCP 목적지 포트 445의 트래픽을 필터링합니다.

ELAM 설정

<#root>

```
switch#
```

debug platform internal tah elam

```
switch(TAH-elam)#
```

trigger init

```
Slot 1: param values: start asic 0, start slice 0, lu-a2d 1, in-select 6, out-select 0
switch(TAH-elam-insel6)#
```

set outer ipv4 src_ip 10.93.19.8

```
switch(TAH-elam-insel6)#
set outer ipv4 dst_ip 10.91.2.35

switch(TAH-elam-insel6)#
set outer l4 l4-type 0

switch(TAH-elam-insel6)#
set outer l4 dst-port 445

switch(TAH-elam-insel6)#
start
```

트래픽을 생성한 후 결과를 검색합니다.

<#root>

```
switch(TAH-elam-insel6)#
report
```

역방향 트래픽 캡처(전체 가시성을 위해 필수)

반환 경로를 검증하려면 소스 및 대상 IP 주소를 스와핑하여 컨피그레이션을 반복합니다.

<#root>

```
switch#
debug platform internal tah elam

switch(TAH-elam)# trigger init
Slot 1: param values: start asic 0, start slice 0, lu-a2d 1, in-select 6, out-select 0
switch(TAH-elam-insel6)#
set outer ipv4 dst_ip 10.93.19.8
```

```
switch(TAH-elam-insel6)#

set outer ipv4 src_ip 10.91.2.35


switch(TAH-elam-insel6)#

set outer l4 l4-type 0


switch(TAH-elam-insel6)#

set outer l4 dst-port 445


switch(TAH-elam-insel6)#

start
```

운영 참고 사항

- ELAM은 하나의 패킷만 캡처하므로 캡처를 시작할 때 트래픽이 활발하게 흐르는지 확인합니다.
- 관련 없는 트래픽을 캡처하지 않으려면 필터가 정확해야 합니다.
- vPC 환경에서는 트래픽이 각 방향으로 다르게 해시될 수 있으므로 두 스위치에서 ELAM을 실행합니다.
- 출력에는 인그레스 인터페이스, 이그레스 인터페이스, 전달 결정이 하드웨어에서 표시되므로 데이터 평면에 대한 신뢰할 수 있는 가시성이 제공됩니다.

참조

Cisco Nexus 9000 Cloud Scale ASIC ELAM 가이드

인터페이스 레벨 검증

인터페이스 레벨 검증은 Nexus 스위치가 TCP 트래픽에 영향을 주는 어떤 제약 조건이나 이상 징후도 일으키지 않도록 합니다. 초점은 컨피그레이션, 운영 상태 및 하드웨어 카운터가 고성능 데이터 플레인 포워딩에 필요한 예상 동작과 일치하는지 확인하는 것입니다.

구성 검증

- 인터페이스에 적용되는 제한 ACL이 없는지 확인합니다.

<#root>

switch#

show running-config interface ethernet1/1-2 | include access-group

- 의도하지 않은 QoS 정책이 트래픽에 영향을 미치지 않는지 확인합니다(대기, 폴리싱 및 셰이핑을 비롯한 인터페이스 레벨 및 글로벌 QoS).

<#root>

switch#

show running-config interface ethernet1/1-2 | include service-policy

switch#

show policy-map interface ethernet1/1-2

<#root>

switch#

show policy-map

<#root>

switch#

show class-map

<#root>

switch#

```
show class-map type network-qos
```

<#root>

```
switch#
```

```
show policy-map type network-qos
```

<#root>

```
switch#
```

```
show policy-map system type network-qos
```

<#root>

```
switch#
```

```
show queuing interface ethernet1/1-2
```

<#root>

```
switch#
```

```
show policy-map type queuing
```

- VLAN 멤버십, STP 상태, IP 주소 지정 등 레이어 2 또는 레이어 3 컨피그레이션(스위치포트 대 라우티드 인터페이스)을 확인합니다.

<#root>

```
switch#
```

```
show running-config interface ethernet1/1-2
```

<#root>

switch#

show interface ethernet1/1-2 switchport

<#root>

switch#

show spanning-tree interface ethernet1/1-2

<#root>

switch#

show ip interface ethernet1/1-2

운영 상태 검증

- MTU 일관성을 확인하고 예상 컨피그레이션(예: 1500 또는 9000바이트)과 일치하는지 확인합니다.

<#root>

switch#

show interface ethernet1/1-2 | include MTU

- 인터페이스 속도 및 이중 설정 확인:

<#root>

```
switch#
 show interface ethernet1/1-2 | include speed|duplex
```

- 인터페이스 안정성 검증(플래핑 또는 빈번한 링크 전환 없음):

```
<#root>
switch#
 show interface ethernet1/1-2 | include rate|flap
```

오류 카운터 유효성 검사

- 테스트 전에 카운터를 지웁니다.

```
<#root>
switch#
 clear counters interface all
```

- 오류 카운터 모니터링(0이 아닌 값만):

```
<#root>
switch#
 show interface counters errors non-zero | include Port|Eth1/1|Eth1/2
```

테스트 후 검증

- TCP 트래픽 테스트를 다시 실행하고 카운터를 다시 관찰합니다.

```
<#root>
switch#
 show interface counters errors non-zero | include Port|Eth1/1|Eth1/2
```

- 카운터는 증가하면 안 됩니다. 이러한 증가는 물리적 링크 오류, CRC/FCS 오류 또는 버퍼 오버런/드롭과 같은 잠재적인 레이어 1 또는 하드웨어 관련 문제를 나타냅니다.

라우팅 및 ARP 안정성

라우팅 및 ARP 안정성을 보장하는 것은 Nexus 스위치가 일관된 레이어 3 연결성을 가지고 있고 TCP 성능에 영향을 줄 수 있는 간헐적인 해결 문제가 발생하지 않는지 확인하는 데 매우 중요합니다. 라우팅 엔트리 또는 ARP 확인이 불안정하면 패킷 손실, 레이턴시 증가 또는 트래픽 블랙홀링을 초래할 수 있습니다.

검증 기준

- 소스 및 목적지에 대한 라우팅 엔트리가 있고 안정적이며 자주 변경되지 않아야 합니다.
- ARP 항목은 확인되어야 하며 지속적으로 새로 고치거나 누락되지 않아야 합니다.

<#root>

switch#

show ip route 10.93.19.8

<#root>

switch#

show ip route 10.91.2.35

<#root>

switch#

show ip arp detail | include 10.93.19.8

<#root>

switch#

show ip arp detail | include 10.91.2.35

트래픽이 CPU에 할당되지 않았는지 확인

Cisco Nexus 9000 스위치에서 포워딩은 하드웨어(ASIC)에서 수행되며, CPU는 일반적인 데이터 플레인 작업에 관여하지 않습니다. 따라서 컨트롤 플레인에서 호스트 간 TCP 트래픽을 관찰하는 것은 비정상적이며 예외 또는 잘못된 컨피그레이션으로 인해 패킷이 펀트되고 있음을 나타냅니다. CPU에서 트래픽을 처리해야 하는 경우, 트래픽이 컨트롤 플레인 정책(Control Plane Policing)의 대상이 되며, 트래픽이 허용된 컨트롤 플레인 속도를 초과하면 드롭이 관찰될 수 있습니다.

검증 방법

- Ethanalyzer를 사용하여 컨트롤 플레인에 도달하는 트래픽을 캡처합니다.

<#root>

```
switch#

ethanalyzer local interface inband display-filter "ip.addr==10.93.19.8 and ip.addr==10.91.2.35" limit-ca
```

예상 동작

- CPU에서 호스트 간 TCP 데이터 플레인 트래픽을 관찰할 수 없습니다.

예기치 않은 동작

- 플로우와 일치하는 패킷이 표시될 경우, 다음 원인으로 인해 트래픽이 펀팅됩니다.
  - 예외적인 패킷 처리(TTL 만료, ACL 로깅, 리디렉션)
  - 컨피그레이션 오류 또는 지원되지 않는 기능
  - 잘못된 하드웨어 프로그래밍

패킷 전달 레이턴시 결정

Nexus 9000 스위치의 패킷 포워딩 레이턴시는 패킷 크기, 포워딩 모드 및 활성화된 기능에 따라 달라집니다. Cisco 사양은 일반적으로 컷스루 전달 시 64바이트 패킷의 레이턴시를 참조합니다.

```
+----------------------+----------------------+--------------------------+------------------------------
| Switch Model         | ASIC / Architecture  | Ports (example config)   | Typical Forwarding Latency
|                      |                      |                          | (64B packet)
```

Nexus 93180YC-EX	Cloud Scale (EX)	48x25G + 6x100G	~1.0 - 1.2 microseconds
Nexus 93180YC-FX	Cloud Scale (FX)	48x25G + 6x100G	~0.9 - 1.0 microseconds
Nexus 93180YC-FX2	Cloud Scale (FX2)	48x25G + 6x100G	~0.8 - 0.9 microseconds
Nexus 9364C	Cloud Scale	64x100G	~1.0 microsecond
Nexus 9336C-FX2	Cloud Scale (FX2)	36x100G	~0.8 microseconds
Nexus 93240YC-FX2	Cloud Scale (FX2)	48x25G + 12x100G	~0.8 - 0.9 microseconds
Nexus 92300YC	Broadcom Trident II	48x10/25G + 6x40/100G	~2 - 3 microseconds
Nexus 92160YC-X	Broadcom Tomahawk	48x25G + 6x100G	~2 microseconds

- 컷스루 전달(Nexus 9000의 기본값):
  - 전체 패킷을 수신하기 전에 전달을 시작합니다.
  - 지연 시간(마이크로초 미만~1 마이크로초) 최소화
- 저장 후 전달:
  - 포워딩 전에 전체 패킷을 수신해야 합니다.
  - 패킷 크기에 비례하는 레이턴시를 추가합니다.

추가 기능으로 점증적 레이턴시가 발생할 수 있습니다.

- VXLAN 캡슐화/역캡슐화
- ACL 조회(TCAM 처리)
- QoS 분류 및 큐잉
- 텔레메트리(NetFlow, ERSPAN, sFlow)
- 혼잡 중 버퍼링

그러나:

- 이러한 작업은 하드웨어 파이프라인에서 수행됩니다.

레이턴시가 눈에 띄게 증가하는 유일한 현실적인 시나리오는 혼잡입니다.

- 패킷은 이그레스 대기열에서 버퍼링됩니다.
- 지연은 다음에 따라 달라집니다.
  - 대기열 깊이
  - 인터페이스 사용률
  - QoS 정책

이러한 경우에도

- 레이턴시는 대개 마이크로초 ~ 수백 마이크로초의 낮은 범위입니다.
- 밀리초로 지속되는 지연은 다음을 의미합니다.
  - 극심한 정체

- 초과 가입
- 잘못 구성된 QoS 또는 버퍼링

SPAN to CPU(데이터 플레인의 패킷 캡처)

이렇게 하면 데이터 플레인 트래픽을 컨트롤 플레인으로 미러링하여 패킷을 캡처하고 .pcaping 파일로 내보낼 수 있으므로 Wireshark에서 자세히 분석할 수 있습니다.

설정

```
monitor session 1
 source interface ethernet1/1 both
 source interface ethernet1/2 both
 destination interface sup-eth0
 no shut
```

캡처 실행

<#root>

```
switch#
```

```
ethanalyzer local interface inband mirror capture-filter "tcp port 445" limit-capture 0 write bootflash:
```

기술 고려 사항

- CPU로 미러링되는 트래픽은 CoPP(Control Plane Policing)의 적용을 받습니다.
- 트래픽이 CoPP를 초과하는 경우:
  - 패킷은 제어 평면에서만 삭제할 수 있습니다.
  - 그러면 분석 중에 오탐이 생성됩니다.
- SPAN to CPU는 낮음에서 보통 트래픽 시나리오에 사용하는 것이 좋습니다.
- 처리량이 많은 환경의 경우:
  - 로컬 SPAN 사용(외부 분석기)
  - 원격 캡처에 ERSPAN 사용

방법	장점	제한
스팬	정확성, 캡슐화 안 함	물리적 연결이 필요합니다.

방법	장점	제한
에르스판	원격 캡처 기능	네트워크 혼잡에 취약합니다.

컨트롤 플레인 속도 제한 검증

SPAN-to-CPU 캡처를 안정적으로 유지하려면 컨트롤 플레인이 속도 제한으로 인해 미러링된 패킷을 삭제하지 않는지 검증해야 합니다.

유효성 검사 명령

```
switch(config)# show hardware rate-limiter | i Allowed|span

Allowed, Dropped & Total: aggregated bytes since last clear counters

R-L Class     Config Allowed Dropped Total

span          50           0       0     0 <<<

span-egress   disabled     0       0     0
```

검증 방법론

- ~3초 간격으로 명령을 실행합니다.
- SPAN 관련 삭제 카운터를 확인합니다.

해석

- SPAN 행에 대한 삭제 카운터가 증가하지 않으면 신뢰할 수 있는 캡처가 있음을 나타냅니다.
- 드롭 카운터가 증가하면 컨트롤 플레인에서 패킷이 손실되어 캡처를 신뢰할 수 없게 됩니다.

드롭이 관찰되면 캡처 방법을 SPAN 또는 ERSPAN으로 변경해야 합니다.

TCP 이전의 ICMP 기반 검증

ICMP 테스트는 복잡한 TCP 분석을 수행하기 전에 데이터 플레인 무결성의 베이스라인 검증을 제공합니다. ICMP는 스테이트리스(stateless)이며 더 간단하기 때문에 패킷 손실, 중복 또는 경로 불일치를 신속하게 탐지할 수 있습니다.

SPAN 캡처의 예상 동작

- 각 ICMP 패킷은 두 번 나타날 수 있습니다.
  - 인그레스에 한 번
  - 한 번 이그레스
- 표준 ping의 경우:
  - 에코 요청 → 패킷 2개
  - 에코 응답 → 패킷 2개

그러면 데이터 플레인에서 패킷 손실이 발생하지 않고 올바르게 포워딩할 수 있습니다.

비정상적인 행동

- 중복 또는 비대칭 패킷 수가 누락되면 잠재적인 패킷 손실 또는 캡처 제한이 있음을 나타냅니다.
- 간헐적인 시간 초과는 레이어 1 문제, 혼잡 또는 업스트림 문제를 나타냅니다.

ICMP 트래픽이 손실 없이 지속적으로 전달되는 경우, TCP 트래픽도 레이어 2/3에서 올바르게 전달될 가능성이 높습니다.

패킷 캡처를 사용하여 Nexus 스위치 포워딩 레이턴시 결정

SPAN-CPU(또는 SPAN/ERSPAN)를 사용하여 트래픽을 캡처할 경우 각 패킷을 두 번 관찰할 수 있습니다. 인그레스 및 이그레스 모두에 대해 두 인스턴스 간의 시간 차이를 계산하여 Nexus 스위치에 의해 도입된 포워딩 레이턴시를 추정하는 데 이 중복 기능을 사용할 수 있습니다.

실제로 이 레이턴시는 이전에 캡처된 ICMP 트래픽을 사용하여 중복된 에코 요청과 에코 응답 패킷 간의 시간 델타를 비교하여 측정할 수 있습니다. 이는 스위치 포워딩 성능을 위한 간단하고 효과적인 베이스라인을 제공합니다. 심층적인 분석이 필요한 경우, 플로우를 캡처하고 복제된 TCP 패킷 간의 시간 차이를 측정하여 동일한 방법을 TCP 트래픽에 적용할 수 있습니다.

방법론

- 패킷 및 중복(동일한 시퀀스 번호)을 식별합니다.
- 인그레스 복사본과 이그레스 복사본 간의 시간 델타를 측정합니다.
- 이 델타는 스위치 포워딩 레이턴시에 대한 상한 추정치를 나타내며, 여기에는 미러링 및 타임스탬프 오버헤드가 포함될 수 있습니다.

Wireshark 컨피그레이션

- 시간 델타 표시 사용:

```
View > Time Display Format > Seconds Since Previous Displayed Packet
```

- 시간 델타에 대한 사용자 지정 열을 추가합니다.

```
Right-click on "Time Delta from Previous Displayed Packet" → Apply as Column
```

- 관련 트래픽 필터링(예):

```
ip.addr==10.93.19.8 and ip.addr==10.91.2.35 and tcp
```

- 시퀀스 번호 또는 TCP 스트림별로 패킷을 정렬합니다.

```
Right-click packet → Follow → TCP Stream
```

해석

- 복제된 패킷 간의 시간 델타는 마이크로초 범위에 있을 수 있습니다.
  - 이 경우 Nexus 스위치는 패킷 포워딩에 레이턴시를 도입하지 않습니다.
- 일관된 낮은 델타에 따라 하드웨어 기반 포워딩 성능이 확인됩니다.
- 델타가 높거나 일치하지 않으면 다음을 나타낼 수 있습니다.
  - 혼잡 또는 버퍼링

참조

- Cisco Nexus 9000 Series 데이터 시트
- Cisco Nexus 9000 Series 스위치 설계 가이드
- Intelligent Buffer Management on Cisco Nexus 9000 Series Switches 백서

소스 호스트 패킷 캡처로부터의 TCP 트래픽 분석

이 섹션에서는 앞서 설명한 가설 사례를 통해 프로파일 컨피그레이션을 포함하여 Wireshark에서 TCP 패킷 캡처를 분석하는 자세한 방법론을 제공합니다. 공개된 이미지는 와이어샤크사의 직접 촬영한 것이다. 다시 한 번 상기시켜 드리자면 시나리오는 다음과 같습니다.

한 사용자가 이전에 약 9시간 만에 완료되었던 약 6.5TB의 애플리케이션 데이터 세트에 대한 백업 프로세스가 이제 거의 21시간이 걸리고 있음을 확인했습니다. 액세스 가능한 유일한 네트워크 디바이스는 소스 서버(10.93.19.8)에 연결된 Cisco Nexus 9300 스위치입니다. 스위치 인터페이스에 구성된 MTU는 9000바이트(점보 프레임)이지만 서버의 MTU는 알 수 없습니다. 소스 서버의 패킷 캡처를 사용할 수 있으며, 모든 이전 Nexus 검증 단계가 이미 완료되었지만 이상 징후는 감지되지 않았습니다.

주요 관점 및 제약 조건

- Nexus 스위치는 제외되었습니다.
  - 패킷 삭제 없음
  - 인터페이스 오류 없음
  - QoS 또는 ACL에 영향 없음
  - 하드웨어 포워딩 확인됨
- 인터페이스 구성:
  - 액세스 포트
  - MTU: 9000바이트
- 사용 가능한 데이터:
  - 소스에서 패킷 캡처
  - 엔드 투 엔드 MTU 지식
    - 1,472바이트의 데이터가 포함된 1500바이트 패킷을 사용하여 단편화 없이 ping을 성공적으로 완료했습니다.
- 누락된 데이터:
  - 대상 가시성
  - 대상 서버에서 사용할 수 있는 패킷 캡처가 없습니다.

Wireshark에서는 수행하려는 특정 분석 유형에 맞게 맞춤화된 사용자 지정 프로필을 생성할 수 있습니다.

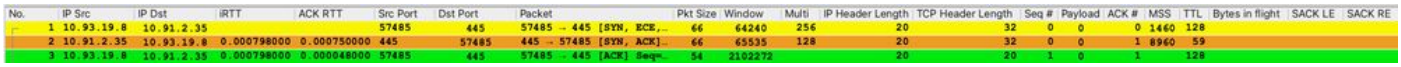
열 설명

- tcp.analysis.initial_rtt(iRTT): TCP 3-way 핸드셰이크를 기반으로 초기 왕복 시간을 추정합니다.
- tcp.analysis.ack_rtt(ACK RTT): TCP 세그먼트와 해당 승인 사이의 시간을 측정합니다.
- tcp.window_size(창): 스케일링이 적용되기 전에 수신기의 알려진 TCP 윈도우 크기를 나타냅니다.
- tcp.options.wscale.multiplier(다중): 유효 수신 창을 계산하는 데 사용되는 창 배율 계수를 나타냅니다.
- tcp.seq(시퀀스 번호): TCP 세그먼트의 첫 번째 바이트의 시퀀스 번호를 표시합니다.

- tcp.len(페이로드): 해당 세그먼트에 대한 TCP 페이로드의 크기를 바이트 단위로 표시합니다.
- tcp.ack(ACK#): 발신자의 다음 예상 바이트(누적 승인)를 나타냅니다.
- tcp.options.mss_val(MSS): TCP 핸드셰이크 중에 광고되는 최대 세그먼트 크기를 표시합니다.
- ip.ttl(TTL): 홉 수 및 라우팅 동작을 식별하는 데 유용한 TTL(Time To Live) 값을 표시합니다.
- tcp.analysis.bytes_in_flight(전송 중인 바이트): 현재 전송 중인 승인되지 않은 데이터의 양을 나타냅니다.

TCP 3-Way 핸드셰이크 분석

TCP 3방향 핸드셰이크는 세션 동작을 정의하는 MSS, Window Scale 및 SACK과 같은 중요한 매개변수를 포함하므로 반드시 캡처해야 합니다.
이 정보가 없으면 TCP 분석이 완료되지 않으며, 성능 또는 근본 원인에 대한 잘못된 결론을 도출할 수 있습니다.



No.	IP Src	IP Dst	iRTT	ACK RTT	Src Port	Dst Port	Packet	Pkt Size	Window	Multi	IP Header Length	TCP Header Length	Seq #	Payload	ACK #	MSS	TTL	Bytes in flight	SACK LE	SACK RE
1	10.93.19.8	10.91.2.35			57485	445	57485 → 445 [SYN, ECE,..	66	64240	256	20	32	0	0	0	1460	128			
2	10.91.2.35	10.93.19.8	0.000798000	0.000750000	445	57485	445 → 57485 [SYN, ACK]..	66	65535	128	20	32	0	0	1	8960	59			
3	10.93.19.8	10.91.2.35	0.000798000	0.000048000	57485	445	57485 → 445 [ACK] Seq=..	54	2102272		20	20	1	0	1		128			

트래픽 식별

패킷 캡처에서

- 소스 IP 주소: 10.93.19.8
- 대상 IP 주소: 10.91.2.35

초기 iRTT(Round-Trip Time) 분석

초기 RTT(iRTT)는 다음과 같이 계산됩니다.

- iRTT = 798마이크로초

이 값은 다음에서 파생됩니다.

- 패킷 2(SYN-ACK) ACK RTT: 대상이 SYN에 응답하는 데 걸리는 시간→ 750µs입니다.
- 패킷 3(ACK) ACK RTT: 48µs → 소스에서 SYN-ACK를 승인하는 시간.

대다수의 지연 시간(~94%)은 전달 경로(클라이언트 → 서버 → 클라이언트)에 있는 반면, 소스의 응답 시간은 최소화되어 클라이언트에서 CPU 또는 애플리케이션 지연이 없음을 나타냅니다.

TCP 포트 식별

- 대상 TCP 포트: 445

포트 445는 파일 공유, 네트워크 드라이브 및 Windows 인증 서비스에 일반적으로 사용되는 Microsoft SMB(Server Message Block)에 해당합니다. 이 프로토콜은 레이턴시와 처리량에 모두 민감하므로 TCP 효율성과 네트워크 안정성에 크게 의존합니다.

TCP 창 크기 분석

- 소스 창(배율 조정): 64,240바이트
- 대상 창: 65,535바이트

TCP 창은 확인 응답을 대기하기 전에 전송할 수 있는 데이터의 양을 나타냅니다. 이 경우 소스가 목적지보다 약간 더 제한적입니다. 이러한 값은 최신 환경에서 상대적으로 작으며 특히 RTT가 증가하면 처리량을 제한할 수 있습니다.

이론상의 최대 처리량은 다음을 사용하여 추정할 수 있습니다.

처리량 = TCP 윈도우 크기/RTT

관찰된 값 대체:

- TCP 창 크기 = 64,240바이트
- RTT = 798마이크로초 = 0.000798초

처리량 ≈ 64,240/0.000798 ≈ 80.5MB/s(~644Mbps)

이는 다음을 가정한 상한 처리량을 나타냅니다.

- 패킷 손실 없음
- 재전송 없음
- 이상적인 네트워크 조건

처리량, 전송 시간 및 필수 조건 분석

현재 처리량이 644Mbps일 때 6.5TB 파일을 전송하는 데 약 23.5시간이 걸리며, 이는 관찰된 성능 저하와 일치합니다. 9시간 전송 윈도우를 달성하려면 처리량이 약 1.68Gbps로 증가해야 하므로 더

큰 TCP 윈도우(~2.7x 증가)나 상당히 낮은 RTT(~291µs)가 필요합니다.

현재 조건(64KB Window 및 ~798µs RTT)에서는 TCP 처리량이 대역폭 지연 제품에 의해 제한되므로 9시간 목표에 도달할 수 없습니다. 윈도우 크기를 늘리거나 레이턴시를 줄이지 않으면 프로토콜에서 더 높은 가용 대역폭을 사용할 수 없으므로 대상을 달성할 수 없습니다.

시나리오	처리량	예상 전송 시간(6.5TB)	필수 TCP 창	필수 RTT
현재 상태	644Mbps(~80.5MB/s)	~23.5시간	64KB	798마이크로초
대상(9시간)	~1,683Mbps(~210MB/s)	9시간	~172KB	~291마이크로초

이는 이전에 작동하여 네트워크, 애플리케이션, 소스 또는 목적지에 변화가 발생했음을 나타냅니다. 이 초기 분석만으로도 이미 다음과 같은 중요한 결론을 내릴 수 있습니다. 현재 TCP 윈도우 크기 및 RTT 조건에서는 9시간 목표를 달성할 수 없습니다.

이 표에서는 RTT 및 TCP 윈도우 크기가 증가 또는 감소할 때 처리량이 어떻게 달라지는지 비교합니다.

처리량에 대한 RTT 영향(고정 윈도우 크기 = 64,240바이트)

RTT	처리량(MB/초)	처리량(Mbps)
200마이크로초(0.0002초)	~321MB/초	~2,568Mbps
798마이크로초(0.000798마이크로초)	~80.5MB/초	~644Mbps
2ms(0.002초)	~32.1MB/초	~257Mbps
10ms(0.01초)	~6.4MB/초	~51Mbps

TCP 윈도우 크기 영향(고정 RTT = 798µs)

TCP 창 크기	처리량(MB/초)	처리량(Mbps)
16KB(16,384B)	~20.5MB/초	~164Mbps
64KB(64,240B)	~80.5MB/초	~644Mbps
256KB(262,144B)	~328MB/초	~2,624Mbps

TCP 창 크기	처리량(MB/초)	처리량(Mbps)
1MB(1,048,576B)	~1,314MB/초	~10.5Gbps

기술 해석

- 처리량은 RTT→ 반비례하며 레이턴시가 높을수록 성능이 저하됩니다.
- 처리량은 TCP 윈도우 크기와 정비례→ 비례하며, 윈도우가 클수록 용량이 증가합니다.
- 작은 윈도우 크기는 저지연 환경에서도 처리량을 극도로 제한합니다.
- 고속 네트워크(10G+)에서 대역폭을 완전히 활용하려면 윈도우 확장이 필요합니다.

이는 RTT와 TCP 윈도우 크기가 모두 TCP 성능의 중요한 요소이며 처리량 문제를 해결할 때 함께 분석해야 한다는 것을 보여줍니다.

IP 및 TCP 헤더 길이

- IP 헤더 길이: 20바이트
- TCP 헤더 길이: 32바이트

20바이트 IP 헤더는 IP 옵션이 없음을 나타냅니다. 32바이트 TCP 헤더는 TCP 옵션이 사용되고 있음을 확인하고 기본 헤더 뒤에 12바이트를 추가합니다. 이러한 옵션에는 일반적으로 MSS, Window Scale 및 SACK Permitted가 포함됩니다.

TCP 옵션 분석 및 TTL

SACK(Selective Acknowledgment)는 두 엔드포인트 모두에서 활성화됩니다. 그림에는 보이지 않는 부분입니다. SACK를 사용하면 수신자가 인접하지 않은 데이터 블록을 승인하여 어떤 세그먼트가 성공적으로 수신되었는지 발신자에게 정확히 알릴 수 있습니다.

예를 들어, 세그먼트 1000-2000 및 3000-4000이 수신되었지만 2000-3000이 누락된 경우 수신기는 이를 명시적으로 나타낼 수 있다. SACK가 없으면 발신자는 공백 후 모든 데이터를 재전송합니다. SACK를 사용하면 누락된 부분만 재전송됩니다. 따라서 패킷 손실이 있는 환경에서 성능이 크게 향상됩니다.

패킷 1(SYN) 분석

- 시퀀스 번호: 0(Wireshark 표준화)
- 페이로드: 0바이트
- ACK 번호: 0
- MSS: 1460바이트

- TTL: 128

Wireshark는 읽을 수 있도록 시퀀스 번호를 0으로 표준화하지만 실제로는 큰 임의 값입니다. 연결 설정 중에 페이로드의 부재가 예상됩니다. 1460바이트의 MSS 값은 1500바이트의 MTU를 나타냅니다(20바이트 IP 헤더 + 20바이트 TCP 헤더). TTL 128은 Windows 기반 호스트일 수 있으며, 캡처에서 이 값을 확인하면 캡처는 레이어 2를 통해 소스에서 또는 그 바로 근처에서 수행되었을 가능성이 높습니다.

패킷 2(SYN-ACK) 분석

- ACK 번호: 1

페이로드가 없는 경우에도 SYN 플래그는 하나의 시퀀스 번호를 사용하기 때문에 ACK 값은 1입니다. 따라서 ACK = SEQ + 1입니다.

- TTL: 59

관측된 TTL 59는 초기 TTL 64를 나타냅니다. 즉 패킷이 약 5개의 라우팅 홉을 통과했습니다(64 − 59 = 5). 라우팅된 각 홉은 TTL을 1씩 줄입니다.

단편화 위험 및 네트워크 영향

약 5개의 라우팅 홉이 있으면 특히 MTU 불일치 및 단편화와 관련된 잠재적인 성능 위험이 발생합니다.

중간 링크의 MTU가 원래 패킷 크기보다 작으면 조각화가 발생할 수 있습니다. 이는 다음과 같은 몇 가지 결과로 이어집니다.

- 단편화 및 리어셈블리 오버헤드로 인해 레이턴시가 증가했습니다.
- 단일 프래그먼트를 잃으면 전체 패킷을 재전송해야 하므로 패킷 손실의 가능성이 높습니다.
- TCP가 손실을 혼잡으로 해석하고 전송 속도를 줄이므로 처리량이 감소합니다.
- 프래그먼트화를 처리하는 네트워크 디바이스에서 CPU 사용률이 증가했습니다.
- ICMP가 차단되어 무음 패킷이 삭제되는 경우 경로 MTU 검색(PMTUD) 실패의 위험.

이러한 요인으로 인해 경로 전체에서 MTU를 일관되게 유지하거나 필요한 경우 MSS 클램핑을 구현하는 것이 중요합니다.

TCP RTT 분석: ACK RTT 대 초기 RTT

ACK RTT가 iRTT보다 크면 레이턴시가 TCP 핸드셰이크 중에 설정된 베이스라인에 비해 증가했음을 나타냅니다.

이는 네트워크 또는 엔드포인트가 세션 중에 다음과 같은 이유로 추가 지연을 초래하고 있음을 의미합니다.

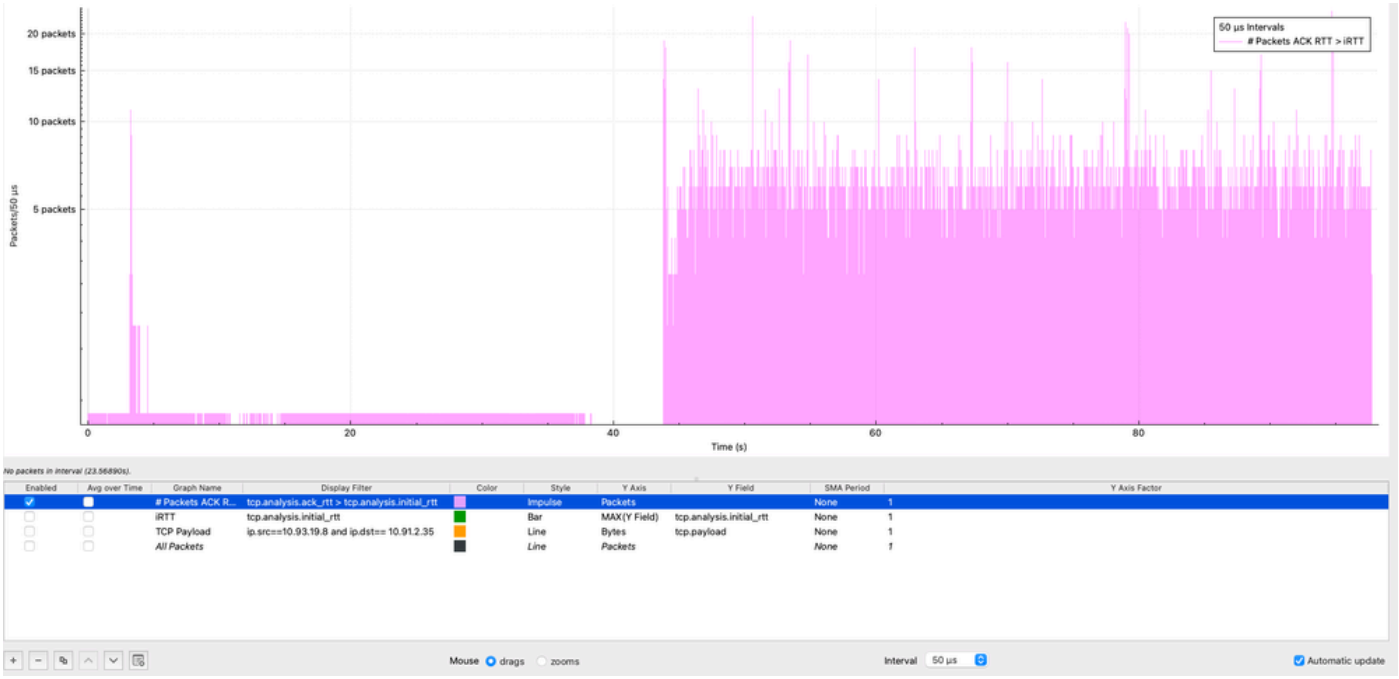
- 네트워크 혼잡 또는 대기열 처리
- 수신기 또는 애플리케이션 처리 지연
- 중간 장치(방화벽, 로드 밸런서)
- 재전송

이 상태가 TCP 세션 전체에서 지속되면 다음과 같이 이어집니다.

- TCP 처리량 감소
- 비효율적인 창 활용
- 애플리케이션 성능 저하

Wireshark에서는 다음과 같은 I/O Graphs(I/O 그래프) 기능을 사용하여 ACK RTT > iRTT 상태가 얼마나 자주 발생하는지 시각화할 수 있습니다. 통계 → I/O 그래프, 디스플레이 필터 적용 (tcp.analysis.ack_rtt > tcp.analysis.initial_rtt), Impulse 스타일 선택, Y축을 Packets로 설정, 간격 50마이크로초 사용

그래프에서 보라색 임펄스는 각 50 마이크로초 간격 동안 이 조건을 충족하는 패킷 수를 나타냅니다. 관찰된 바와 같이, 이 상태는 전체 패킷 캡처 동안 지속되며, 이는 세션 중 레이턴시가 초기 베이스라인보다 지속적으로 더 높음을 나타냅니다. 이러한 동작은 일시적인 상태가 아니라 지속적인 성능 저하를 시사하므로 엔드 투 엔드 경로 전반에서 혼잡, 버퍼링 또는 엔드포인트 처리 지연과 같은 잠재적 원인을 조사할 필요가 강화됩니다.

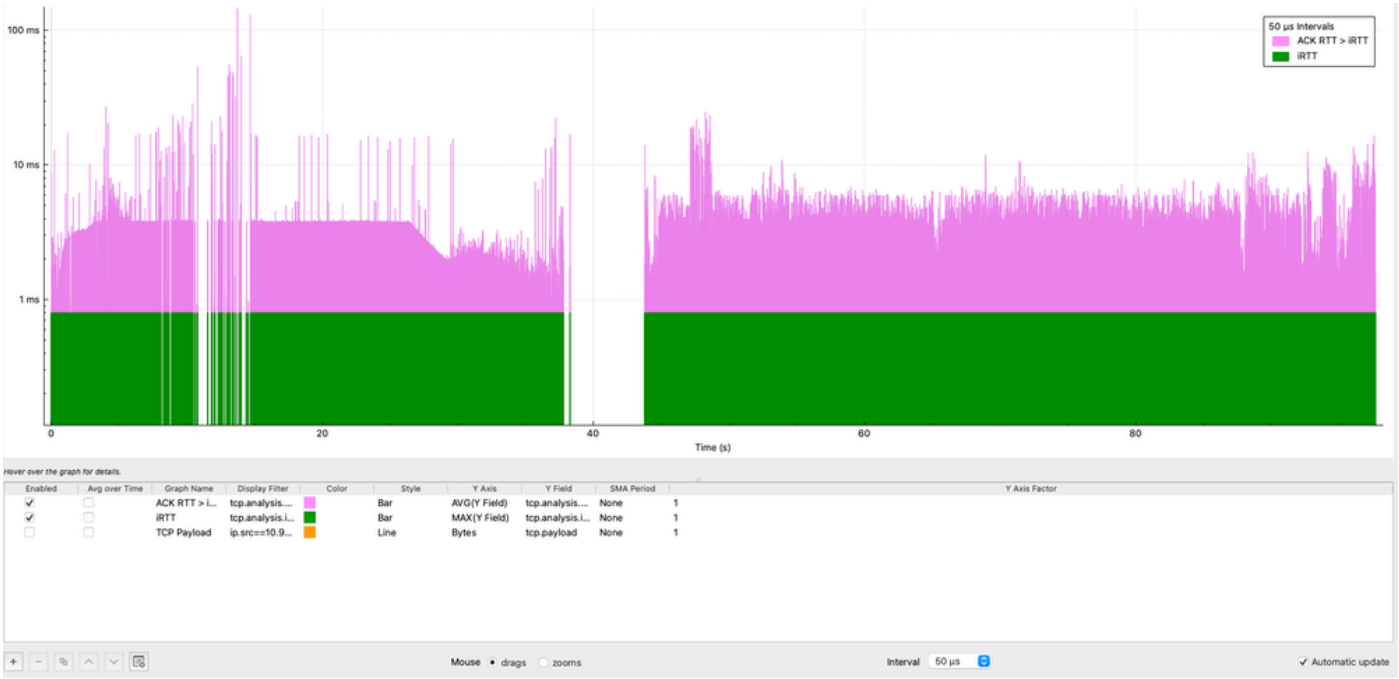


또한 iRTT가 얼마나 자주 초과되는지 확인하는 것이 중요합니다. Wireshark는 필드 간의 빼기를 직접 허용하지 않지만 I/O 그래프를 사용하여 시각적 비교를 수행할 수 있습니다.

- Statistics → I/O Graphs로 이동
- 그래프 1:
  - 표시 필터: tcp.analysis.ack_rtt > tcp.analysis.initial_rtt
  - 스타일: 막대
  - Y축: 평균
  - Y 필드: tcp.analysis.ack_rtt
  - 간격: 50마이크로초
- 그래프 2:
  - 표시 필터: tcp.analysis.initial_rtt
  - 스타일: 막대
  - Y축: 최대
  - Y 필드: tcp.analysis.initial_rtt
- 그런 다음 마우스 오른쪽 버튼으로 그래프를 클릭하고 Log scale을 활성화합니다.

이 시각화에서 보라색 그래프는 상태 ACK RTT > iRTT를 나타내며, 이는 전체 TCP 세션 동안 일관되게 존재합니다. 이 데이터는 여러 피크가 11밀리초에 도달하고 최대 스파이크가 100밀리초를 넘어 기본 iRTT의 11x~100x를 나타내는 지속적인 레이턴시 인플레이션을 보여줍니다.

이러한 동작을 통해 지연 시간 증가가 일시적이지 않고 지속적임을 확인하며, 이는 시간이 지남에 따라 세션에 영향을 주는 시스템 문제를 나타냅니다. 이러한 지속적인 편차는 네트워크 정체, 버퍼링(버퍼링) 또는 엔드포인트 처리 지연과 같은 요인을 강력하게 시사합니다.



TCP 재전송 및 가짜 재전송 분석

이 섹션에서는 시간에 따른 재전송을 분석하여 패킷 손실이 성능 저하에 기여하는지 여부를 검증하여 TCP 안정성을 평가합니다.

시간에 따른 TCP 재전송

그래프는 시간에 따른 TCP 재전송의 분포를 보여줍니다. 총 42건의 재전송이 관찰되어 전체 트래픽의 0.00125%에 불과했습니다.

이러한 재전송 수준은 무시할 수 있으며 패킷 손실이 이 시나리오에 영향을 주는 요인이 아님을 분명히 나타냅니다.

Wireshark 컨피그레이션(TCP 재전송)

```
Statistics → I/O Graphs
```

- 표시 필터:

```
tcp.analysis.retransmission and !tcp.analysis.spurious_retransmission
```

- 스타일: 임펄스 또는 막대
- Y축: 패킷
- 간격: 1초

TCP 스퓨리어스 재전송

그래프는 소스 10.93.19.8에서 생성된 1초 간격의 TCP 스퓨리어스 재전송 수를 보여줍니다.

Wireshark에서 TCP Spurious Retransmission은 호스트가 실제로 손실되지 않은 세그먼트를 재전송했음을 나타냅니다. 원래 패킷이 성공적으로 수신자에게 도달했지만, 발신자가 타이밍 추정이 부정확하여 손실을 잘못 가정했습니다. 이 동작은 실제 패킷 손실을 나타내는 것이 아니라 발신자의 비효율적인 재전송 로직을 나타냅니다.

이 캡처에서는

- 소스 10.93.19.8은 ~8마이크로초 후에 패킷을 재전송합니다.
- 일반적인 재전송 타이머는 ~200밀리초 정도입니다.

이는 재전송 동작이 네트워크가 아닌 소스 TCP 스택에 의해 완전히 제어된다는 것을 확인합니다.

관찰된 총 스퓨리어스 재전송 수는 1,112회로, 총 캡처된 트래픽의 0.0332%를 나타냅니다.

Wireshark 컨피그레이션(TCP 스퓨리어스 재전송)

```
Statistics → I/O Graphs
```

- 표시 필터:

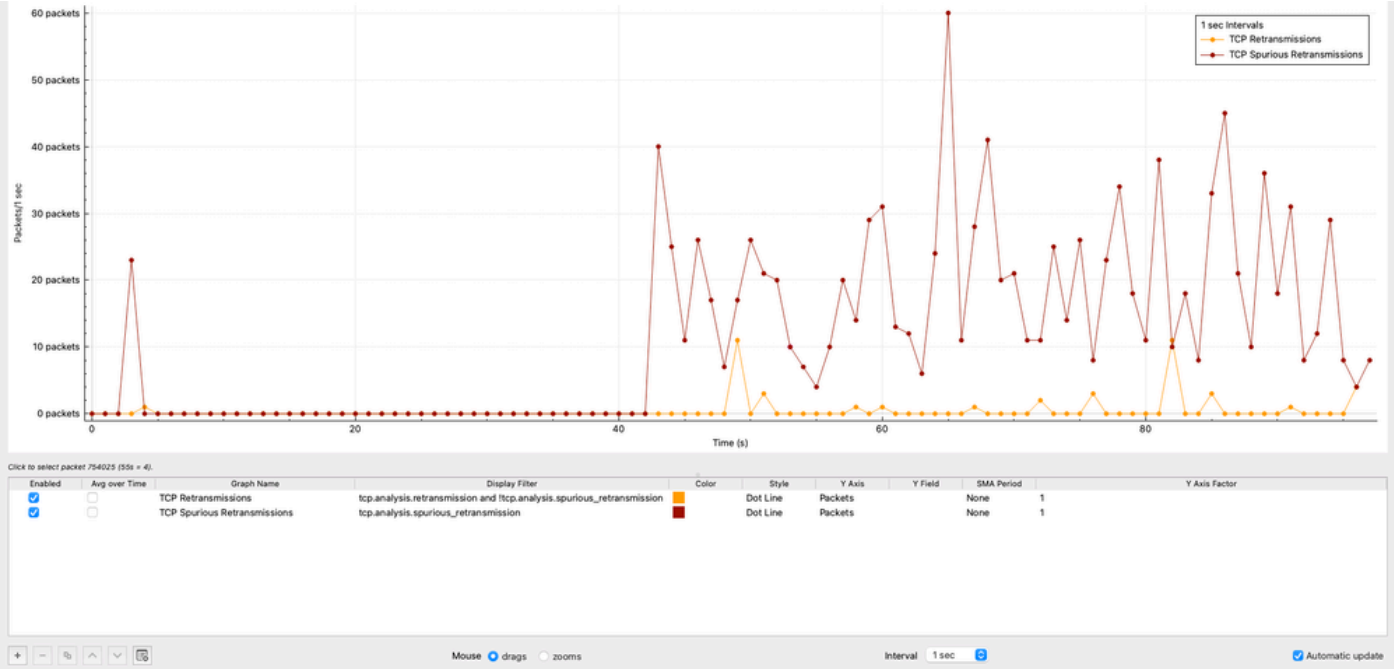
```
tcp.analysis.spurious_retransmission and ip.src==10.93.19.8
```

- 스타일: 임펄스 또는 막대
- Y축: 패킷
- 간격: 1초

기술 해석

- 실제 재전송의 극히 낮은 비율은 네트워크에 패킷 손실이 없음을 확인합니다.
- 스퓨리어스 재전송의 존재는 소스 호스트에 의한 조기 재전송 결정을 나타낸다.
- 이러한 동작은 효율성에 다소 영향을 미칠 수 있지만 심각한 처리량 저하의 주요 원인은 아닙니다.

이 분석은 문제가 네트워크 신뢰성과 관련된 것이 아니라 TCP 동작, 레이턴시 또는 엔드포인트 성능과 관련된 것임을 더욱 강화합니다.

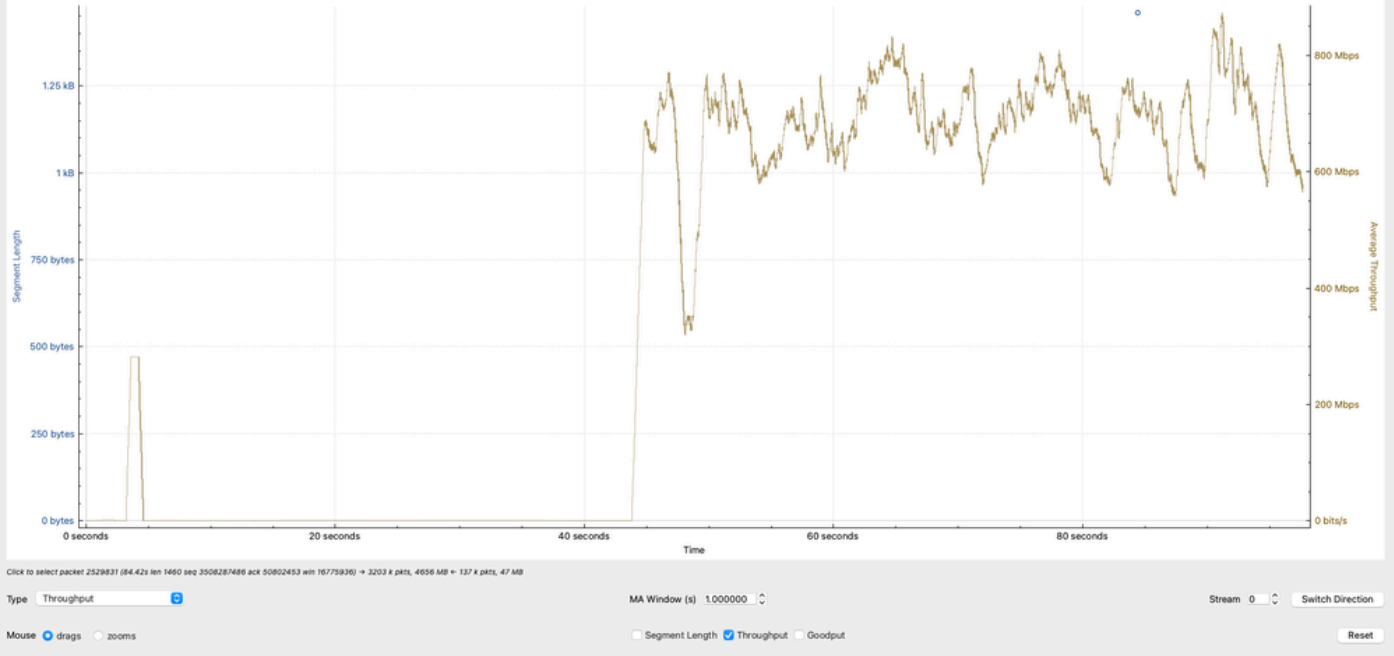


효과적인 처리량 분석

그래프는 TCP 페이로드(실제 전송된 데이터)를 기반으로 계산된 유효 처리량을 초당 메가비트 단위로 보여줍니다. 관찰된 처리량은 주로 600Mbps ~ 800Mbps로 진동하며, 이는 네트워크가 적극적으로 데이터를 전송하는 동안 더 높은 대역폭 잠재력에 도달하지 못하고 있음을 나타냅니다.

Wireshark 컨피그레이션(유효 처리량)

```
Statistics → TCP Streams Graphs → Throughout
```



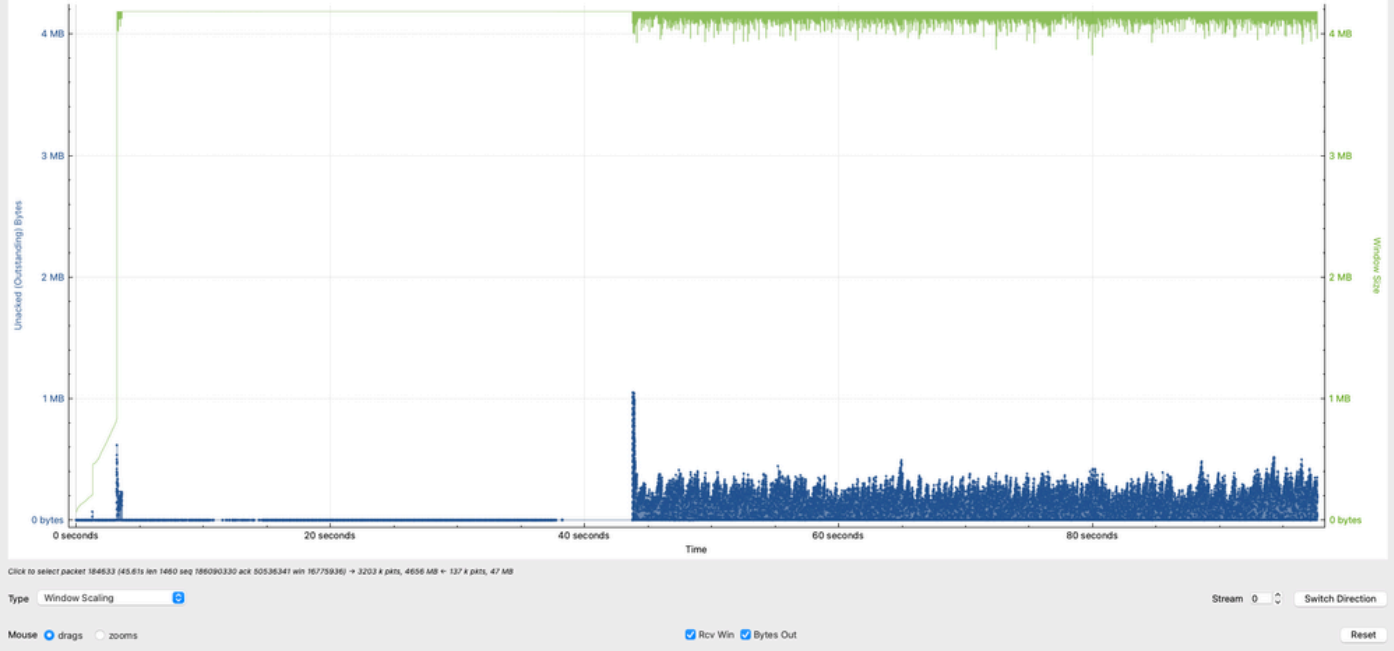
기술 해석

- 처리량 범위 600-800Mbps는 TCP 윈도우 크기 및 RTT를 기반으로 한 이전 계산과 일치합니다.
- 처리량의 가변성은 다음을 반영합니다.
  - RTT 변동
  - TCP 혼잡 제어 조정
  - 애플리케이션 속도 조절 또는 버퍼링
- 처리량은 라인 레이트(예: 10G)에 근접하지 않으므로, 물리적 대역폭이 아니라 TCP 효율성 제약 조건이라는 제한이 따릅니다.
- 이 분석을 통해 관찰된 처리량이 TCP 제한 사항(윈도우 크기 및 레이턴시)과 일치함을 확인함으로써, 병목 현상이 패킷 손실 또는 인터페이스 용량이 아니라 전송 레이어 동작 및 엔드포인트 조건 때문임을 강화합니다.

전송 중인 데이터(TCP 창) 분석

이 그래프는 수신기 용량과 실제 전송 데이터(전송 중인 바이트)를 비교하여 TCP 세션의 중요한 동작을 강조 표시합니다.

- 녹색 선은 10.91.2.35(수신기)가 허용할 수 있는 TCP 데이터의 양을 나타냅니다(유효 수신 창).
- 파란색 선은 10.93.19.8(발신자)에서 현재 전송 중인 TCP 데이터의 양을 나타냅니다.



비행에서 관찰된 데이터는 약 1MB에서 피크이며, 약 8KB 및 5KB의 추가 피크도 있지만, 주로 1KB와 250KB 사이에 집중됩니다.

이는 수신자가 더 많은 양의 데이터를 처리할 수 있지만 발신자가 사용 가능한 창을 일관성 있게 활용하지 않음을 나타냅니다.

Wireshark 컨피그레이션(이동 중의 데이터 vs 창)

```
Statistics → TCP Streams Graphs → Throughout
```

기술 해석

- 수신기(10.91.2.35)는 훨씬 더 큰 창을 광고하여 더 많은 데이터를 수신할 수 있음을 나타냅니다.
- 발신자(10.93.19.8)가 더 낮고 일치하지 않는 Data in Flight 값에서 볼 수 있듯이 사용 가능한 창을 제대로 활용하지 못하고 있습니다.
    - 발신자는 처리량을 최대화하기 위해 Data in Flight 값을 수신자 알림 윈도우(~1MB)에 가깝게 유지하는 것이 좋습니다.
    - 전송 중인 데이터 수준을 높게 유지할 수 없는 경우 처리량이 직접적으로 제한되며, 네트워크 용량 문제가 아니라 소스에서 TCP 비효율성을 나타내는 강력한 지표입니다.

TCP 페이로드 대 MSS Over Time 분석

시간이 지남에 따라 MSS에 대한 TCP 페이로드 크기를 분석하면 발신자가 각 TCP 세그먼트를 효

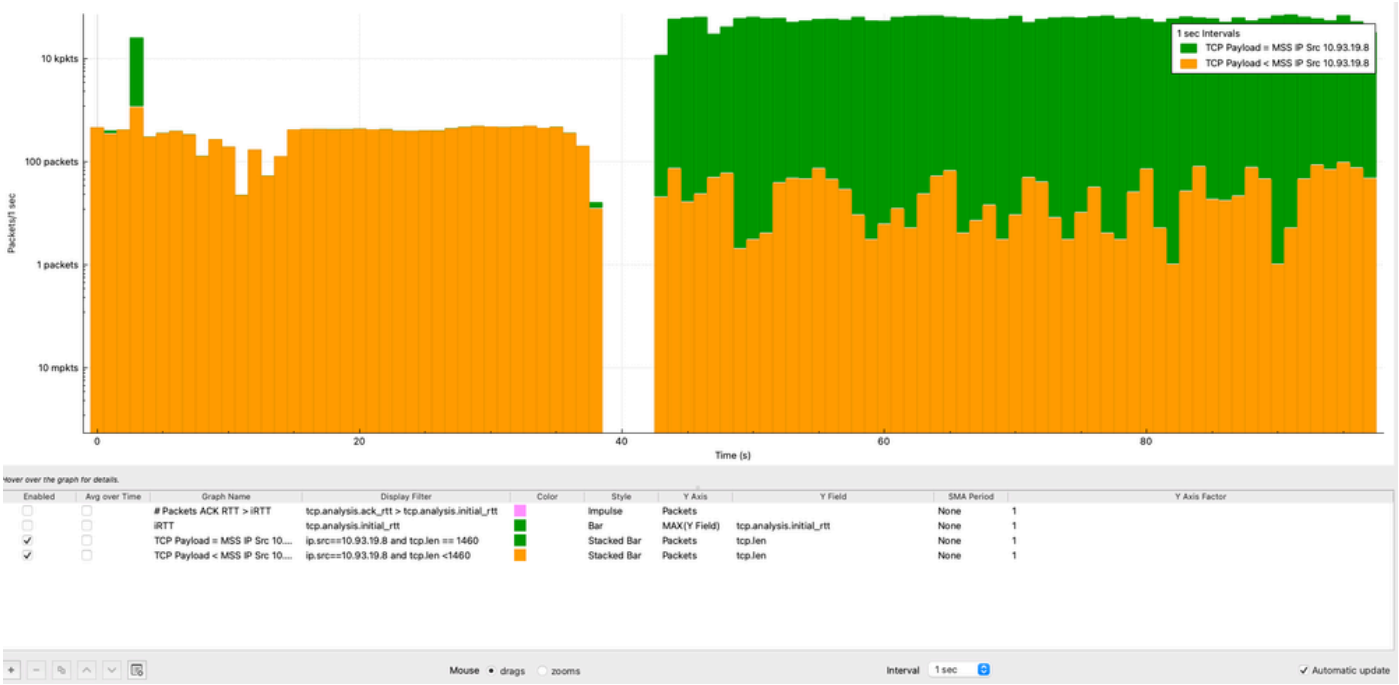
율적으로 사용하고 있는지 여부를 확인할 수 있습니다. 이 분석은 소스 IP 주소(10.93.19.8)의 관점에서 수행됩니다.

Wireshark에서 그래프는 다음과 같이 구성됩니다.

- 그래프 1(MSS 크기 패킷):
  - 표시 필터: ip.src==10.93.19.8 및 tcp.len == 1460
  - 스타일: 누적 가로 막대형
  - Y축: 패킷
  - 간격: 1초
- 그래프 2(모든 패킷≤ MSS):
  - 표시 필터: ip.src==10.93.19.8 및 tcp.len <= 1460
  - 스타일: 누적 가로 막대형
  - Y축: 패킷
  - 간격: 1초
- 로그 눈금을 적용하여 더 나은 시각화

분석 결과:

- 패킷의 대부분(초당 10,000개 패킷 이상)은 일관되게 MSS 값 1460바이트에 도달합니다.
- 일반 TCP 동작(ACK, 세그멘테이션 또는 스트림 종료 데이터)으로 인해 패킷의 더 작은 부분이 더 적은 페이로드를 전달합니다.



RCA(근본 원인 분석): TCP 성능 저하

이 분석에서는 네트워크가 주요 성능 저하의 원인이라고 가정하기 보다는 전체적인 엔드 투 엔드 접근 방식이 TCP 성능 문제의 근본 원인을 파악하는 데 필요하다는 것을 보여 줍니다.

Cisco Nexus 9300 스위치에서는 인터페이스 카운터, QoS 정책, 라우팅 및 ARP 안정성, CPU punt 확인, SPAN 기반 패킷 캡처, ELAM을 사용한 ASIC 레벨 포워딩 확인 등 광범위한 검증이 수행되었습니다. 모든 결과에서 스위치가 예상 매개 변수 내에서 작동하고 있음을 일관되게 확인했습니다.

- 패킷 삭제 없음
- 비정상적인 레이턴시 없음(마이크로초 범위)
- QoS 또는 컨트롤 플레인에 영향을 미치지 않음
- 올바른 하드웨어 전달

또한 TCP 분석에서 다음을 확인했습니다.

- 무시할 수 있는 재전송(0.00125%)
- 패킷 손실의 증거 없음
- 소스에서 일관된 MSS 사용률
- TCP 윈도우 및 RTT 제약에 따른 처리량
- 사용 가능한 TCP 윈도우 미활용(전송 중 데이터 분석)
- 네트워크가 병목 지점이 아님
- 원본 서버에서 성능을 제한하고 있습니다.

결론

성능 저하는 점보 가능 환경에서 MTU 1500으로 작동하는 소스 서버로 인해 발생하여 사용 가능한 네트워크 용량을 효율적으로 사용하지 못하게 됩니다.

솔루션

대상 및 네트워크 인프라에 맞게 소스 서버의 MTU를 1500바이트에서 9000바이트로 늘립니다. 이점:

- 더 큰 TCP 세그먼트 활성화
- 패킷 오버헤드 감소
- 전체 처리량 개선

기술적 성찰

이 분석에서 중요한 교훈은 네트워크 성능을 트러블슈팅할 때 섣부른 결론을 피하는 것입니다. 초기에 네트워크에 문제를 부여하는 것이 일반적이지만, 이 사례는 네트워크가 전체 데이터 플레인 경로에서 올바르게 작동하고 있음을 분명히 보여 줍니다. 소스 및 대상 관점(핸드셰이크 매개변수, RTT 동작, 창 사용률, 재전송, 페이로드 효율성 등)에서 심층적인 TCP 분석을 수행해야만 진정한 병목 현상을 정확하게 식별할 수 있었습니다.

시간을 들여 TCP 동작을 세부적으로 분석함으로써 오진단을 방지하고, 불필요한 네트워크 변경을 줄이며, 치료 노력이 실제 근본 원인을 찾도록 보장합니다.

이 번역에 관하여

Cisco는 전 세계 사용자에게 다양한 언어로 지원 콘텐츠를 제공하기 위해 기계 번역 기술과 수작업 번역을 병행하여 이 문서를 번역했습니다. 아무리 품질이 높은 기계 번역이라도 전문 번역가의 번역 결과물만큼 정확하지는 않습니다. Cisco Systems, Inc.는 이 같은 번역에 대해 어떠한 책임도 지지 않으며 항상 원본 영문 문서(링크 제공됨)를 참조할 것을 권장합니다.