排除Nexus 9000(NX-OS)上的TCP效能故障

目錄

簡介

本檔案介紹TCP基本資訊、Wireshark深度封包分析及實際疑難排解，以最佳化端對端效能。

必要條件

需求

思科建議您瞭解以下主題：

- IP/TCP

採用元件

本文中的資訊係根據以下軟體和硬體版本：

- Cisco Nexus 9000雲擴展採用Cisco NX-OS 10.6(X)。

附註：任何有關第三方軟體或硬體的組態和互通性的問題均不在思科支援範圍之內。使用

第三方工具是展示您對思科裝置的配置和操作的最佳方式。

本文中的資訊是根據特定實驗室環境內的裝置所建立。文中使用到的所有裝置皆從已清除（預設）的組態來啟動。如果您的網路運作中，請確保您瞭解任何指令可能造成的影響。

背景資訊

什麼是TCP

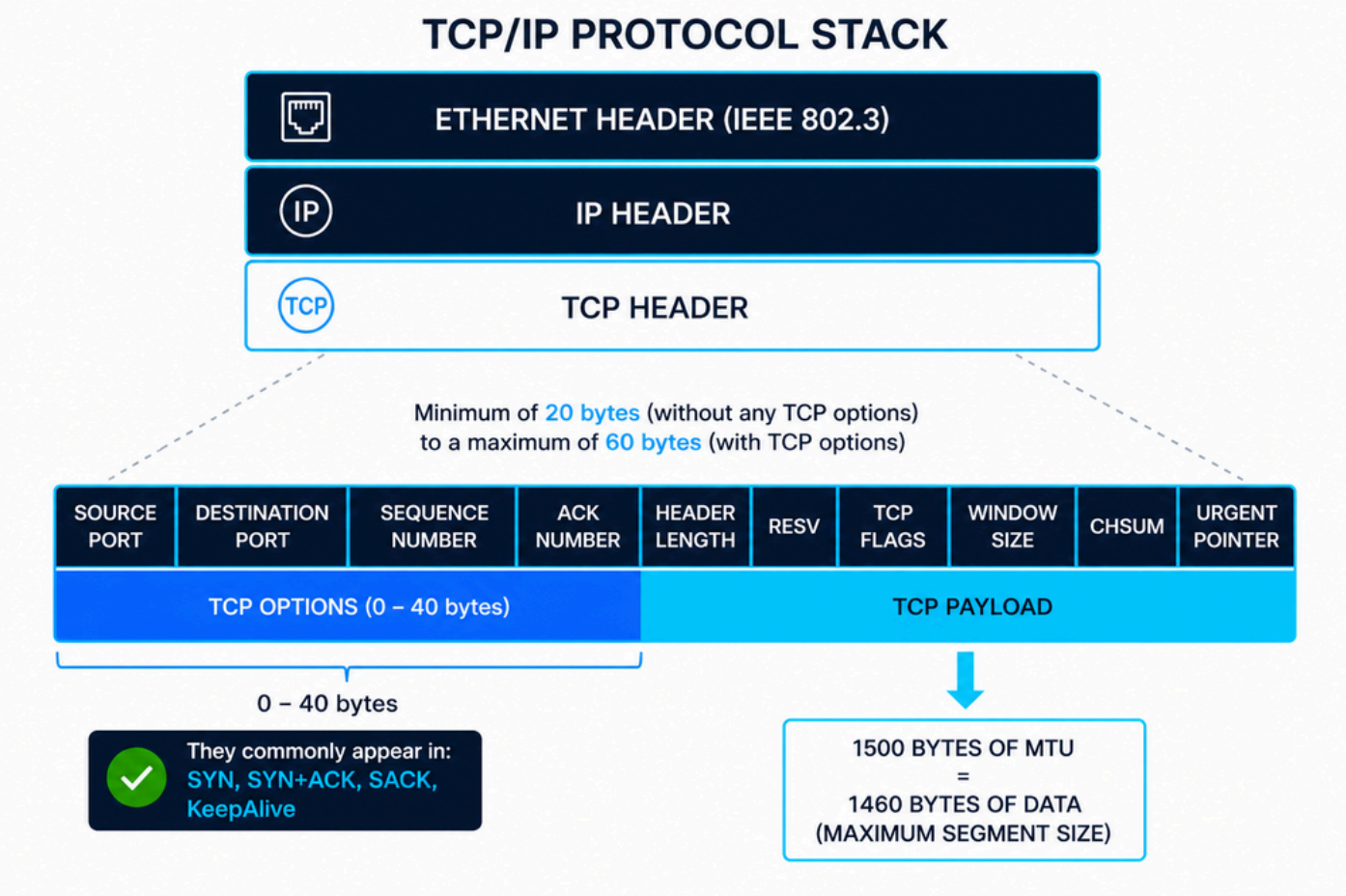
傳輸控制通訊協定(TCP)是一種基本的傳輸層通訊協定，在OSI模型的第4層運作，為透過IP網路通訊的應用程式之間的位元組串流提供可靠、有序、且經過錯誤檢查的傳輸。

三個主要優勢

1. 可靠性：TCP面向連線，通過要求接收方確認來保證傳輸。如果封包在傳輸過程中遺失或損毀，TCP會自動重新傳輸資料，以確保其到達目的地。
2. 訂購的交付：由於網路資料包的到達順序可能混亂，TCP會為每個資料段分配序列號。這允許接收系統按最初傳送的確切順序重組資料。
3. 流量和擁塞控制：TCP動態地管理資料傳輸速率以匹配接收方處理能力和當前網路狀況，防止由緩衝區溢位或網路擁塞引起的資料丟失。

TCP/IP封裝概述

該圖表示TCP/IP堆疊，其中TCP資料段（第4層）封裝在IP資料包（第3層）中，然後封裝在IEEE 802.3定義的乙太網幀（第2層）中。此分層方法可確保模組化通訊，其中每一層都新增自己的控制資訊（報頭）以確保傳輸、路由和資料完整性。



乙太網路標頭(IEEE 802.3)

乙太網報頭通常為14個位元組，由以下部分組成：

- 目標MAC地址（6位元組）
- 源MAC地址（6位元組）
- EtherType/Length（2位元組）

此外，乙太網幀還包括一個4位元組的幀校驗序列(FCS)報尾，用於第2層進行錯誤檢測。 IEEE 802.3定義了幀、最小/最大幀大小和物理傳輸限制，這些限制直接影響TCP等上層協定。

IP標頭(IPv4)

IPv4標頭的最小大小為20位元組，可使用選項擴展至最多60位元組。關鍵欄位包括：

- 源IP地址和目標IP地址
- 生存時間(TTL)
- 協定（將TCP標識為負載）

IP層負責網路間的邏輯定址和路由，但它並不能保證可靠性。

TCP標頭結構

TCP報頭範圍從20到60位元組，具體取決於選項。關鍵欄位包括：

- 來源/目的地連線埠
- 序列號
- 確認編號
- 標誌（SYN、ACK、FIN、RST等）
- 視窗大小
- 校驗和

TCP為IP通訊新增了可靠的傳輸、正確的排序和流量控制。

TCP選項（常見10）

TCP選項擴展了基本協定。最常見包括：

1. 最大片段大小(MSS) — 定義主機可以接受的最大TCP負載。
2. Window Scale — 將接收視窗擴展到65,535位元組以上。
3. 允許選擇性確認（允許SACK） — 啟用選擇性確認功能。
4. 選擇性確認(SACK) — 指定接收的資料塊以避免完全重新傳輸。
5. 時間戳 — 用於RTT計算和針對包裝序列號(PAWS)的保護。
6. 無操作(NOP) — 填充選項的對齊。
7. 選項清單結束(EOL) — 標籤TCP選項的結束。
8. TCP快速開放(TFO) — 允許在初始握手期間交換資料。
9. 多重路徑TCP(MPTCP) — 為單個TCP作業階段啟用多個網路路徑。
10. 使用者超時選項(UTO) — 控制傳輸的資料保持未確認狀態的時間長度。

TCP序列和確認行為（包括SYN/FIN）

SYN和FIN標誌都使用一個序列號，即使沒有負載時也是如此。TCP使用面向位元組的排序模型運行，其中傳輸的每個位元組和特定的控制標誌會增加序列空間。此行為對於在資料包捕獲中進行TCP精確分析以及診斷排序或確認不一致至關重要。

ACK = SEQ +負載長度+(SYN ?1:0)+(FIN ?1:0)

其中：

- SEQ =初始序列號
- 負載長度=以位元組為單位的資料大小
- SYN ?1:0 =如果設定了SYN標誌，則新增1；否則為0
- FIN ?1:0 =如果設定了FIN標誌，則新增1，否則為0
- ACK =下一個期望位元組

範例 1：包含資料的SYN（TCP快速開啟）

- SEQ = 1000
- SYN = 1
- 負載長度= 200位元組

ACK計算：

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

這反映在TCP交握期間傳送資料的情況。負載和SYN標誌都佔用序列空間。

範例 2：帶資料的FIN（連線終止）

- SEQ = 3000
- FIN = 1
- 負載長度= 150位元組

ACK計算：

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

這表明TCP可以在連線斷開期間包含資料，並且負載和FIN標誌會增加序列號。

MSS及其與MTU的關係

最大區段大小(MSS)定義TCP可以在區段中傳送的最大負載。

- 典型乙太網MTU = 1500位元組

- MSS = MTU − IP標頭和−標頭
- 標準MSS = 1460位元組(1500 − 20 − 20)

如果存在TCP選項，則MSS會相應地降低。MSS是在TCP三次握手期間交涉的，用於防止IP層進行分段。

MSS交涉在TCP三次握手中的運作方式

在TCP三次握手期間，會使用SYN封包中的MSS選項交換最大區段大小(MSS):

- 主機A→主機B(SYN):通告其MSS（例如，1460）
- 主機B→主機A(SYN-ACK):通告其MSS（例如，1380）

雙方都在有效地表示：

這是接受的最大TCP負載。

Key Rule:MSS是方向性的

MSS不會作為單一協定價值交涉。

而非：

- 每台主機都使用另一端通告的MSS。
- 這將建立兩個獨立的限制，每個方向一個。

因此：

- A使用B的MSS傳送資料。
- B使用A的MSS傳送資料。

來源是否可傳送比目的地MSS更多的TCP負載

在正常運作的TCP堆疊中：編號

- 傳送者必須尊重接收者通告的MSS。
- 傳送更大的資料段會有風險：

    - IP分段（如果超過MTU）
    - 封包捨棄（如果分段遭封鎖或不受支援）
- 這將導致：
    - 重新傳輸
    - 效能降級
    - PMTUD（路徑MTU探索）黑洞之類的問題

故障排除實用見解

- 請一律在TCP三次握手（SYN/SYN-ACK封包）中驗證MSS值。
- 檢查由下列原因造成的不匹配：
    - 通道(VXLAN、GRE、IPsec)
    - 修改MSS（MSS夾緊）的防火牆
- 在Cisco NX-OS等平台上，MSS調整通常用於防止封裝路徑之間的分段

視窗大小（流量控制）

視窗大小定義接收方可以在不確認的情況下接受的資料量。

它是：

- 一種防止緩衝區溢位的流量控制機制。

目的:

- 確保傳送方不會壓垮接收方。

獲取位置：

- 在資料包捕獲（例如Wireshark）中可見。
- 源自OS TCP堆疊配置和緩衝區大小。

供應商/作業系統可變性：

- 不同的實施(Linux、Windows、Cisco NX-OS)使用動態擴展和緩衝區調整，導致視窗大小不同。

零視窗條件：

- 當Window Size = 0時，接收方緩衝區已滿。
- 傳送方暫停傳輸並傳送定期探測。

可變Windows機制

- 基於速率的流量控制
  - 它為傳送方分配一個固定資料速率，並確保資料不會超過該分配。
  - 流應用的理想之選。
  - 廣播和組播傳輸
- 基於視窗的流量控制
  - 視窗大小會隨著時間而變化。
  - 接收方通過傳送允許視窗到傳送方視窗更新來實現流量控制。

故障排除使用：

- 接收器端的→小或零視窗（CPU、記憶體、應用程式）。
- 大視窗但吞吐量低→網路問題（延遲、擁塞）。
- 分析視窗行為對於診斷TCP會話中的效能問題至關重要。

Cisco Nexus 9000(NX-OS)上的TCP資料平面故障排除

本節介紹用於診斷運行NX-OS的Cisco Nexus交換機是否影響TCP流量轉發或引入效能問題的實用方法。通過假設情景給出了該方法。

觀察到TCP延遲或效能下降時，通常最初懷疑是網路導致延遲或效能下降。但是，必須通過資料驅動分析驗證此假設。TCP故障排除的授權方法是資料包捕獲，最好執行：

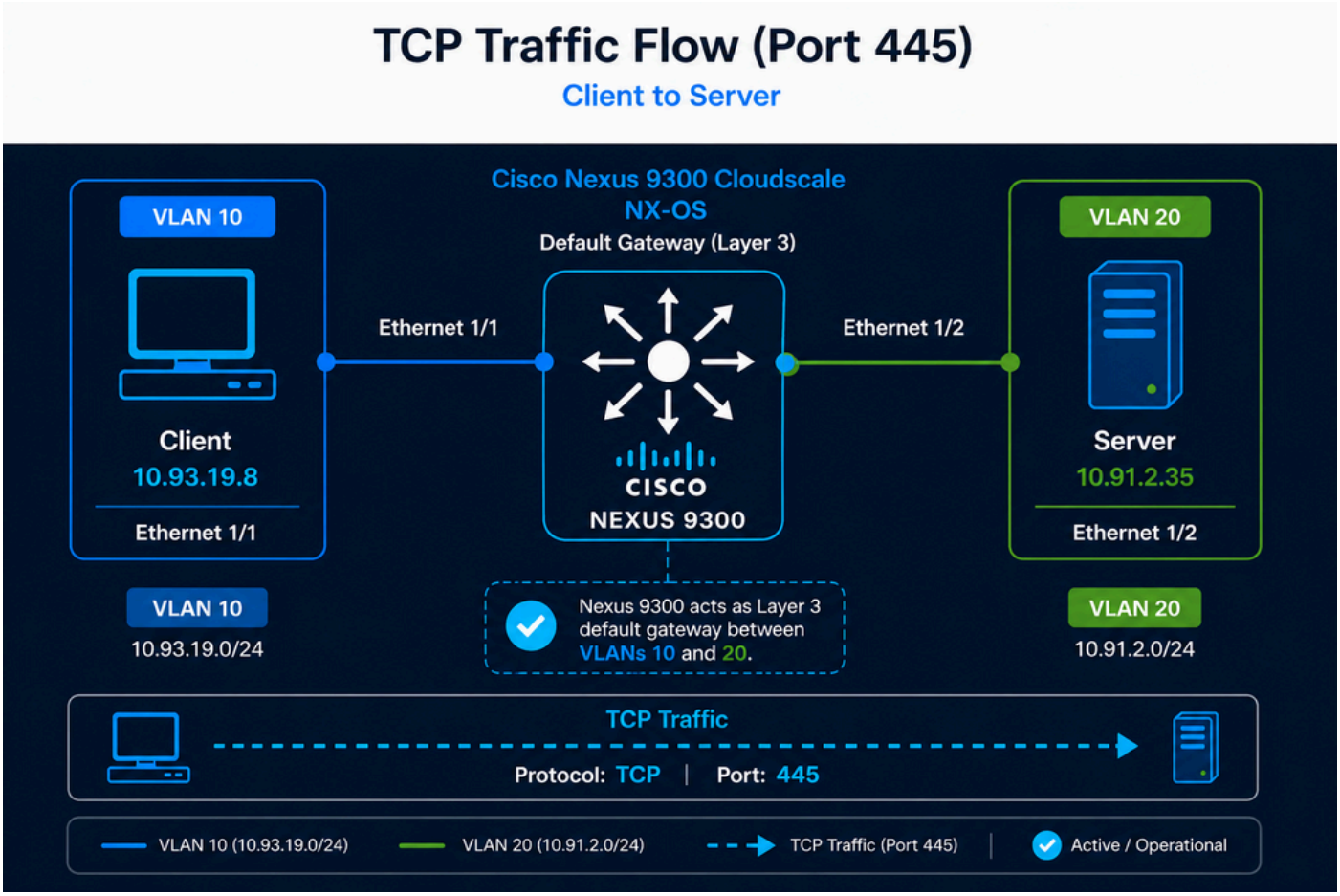
- 同時在源和目標
- 流量發起之前

這可以確保對TCP三次握手的可視性，在此三次握手中，MSS、視窗縮放和SACK等關鍵引數是經過協商的，不會在會話後期重複這些引數。如果無法進行同時捕獲，則可以使用一次捕獲進行分析，但結論有限。

方案定義

一位使用者已發現，對大約7.5 TB的應用程式資料集的備份過程（以前大約在9小時內完成）現在需要將近21小時。雖然客戶端和伺服器之間的TCP會話仍然成功建立，但備份持續時間的大幅增加表明吞吐量或整體TCP效能可能會降低。由於Nexus交換機是路徑中唯一的網路裝置，並且還提供第3層網關功能，因此網路管理員懷疑Nexus交換機是問題的原因。

- 用戶端:10.93.19.8(VLAN 10)
- 伺服器:10.91.2.35(VLAN 20)
- Nexus 9300充當預設閘道
- TCP埠445



初始驗證（可達性）

- 這些指令用於透過傳送已設定不分段(DF)位元的ICMP封包，來驗證來源和目的地之間的路徑MTU(PMTU)。這有助於確定無需分段即可通過網路傳輸的最大資料包大小。必須在源主機和目的主機上執行此過程。
- 請始終檢驗源和目標上物理介面的MTU。
- 在此案例中，只有來源主機可以存取，且識別的MTU為1500。

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

- -c 10 →傳送10個ICMP回應請求
- -I 192.168.10.10 -→用此特定源IP/介面
- -s 1472 →將ICMP負載大小設定為1472位元組

- -M do →設定DF（不分段）位
- 192.168.20.20 IP→目的IP

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

- -n 10 -→送10個ICMP回應請求
- -l 1472 →將ICMP負載大小設定為1472位元組
- -f →設定不分段(DF)標誌
- 192.168.20.20 IP→目的IP

為什麼要使用1472位元組？

- ICMP負載= 1472位元組
- IP報頭= 20位元組
- ICMP報頭= 8位元組
- 總資料包大小：1472 + 20 + 8 = 1500位元組（標準MTU）
- 此指令會測試路徑是否支援沒有分段的1500位元組MTU。如果嘗試傳送1500位元組的ICMP負載，Ping可能會失敗，因為在新增IP和ICMP標頭後，總封包大小將超過標準MTU。

可以得出的結論

- 如果ping成功（沒有封包遺失），則路徑至少支援1500位元組的MTU，且不需要分段。
  - 清除ICMP結果→進行TCP分析
  - 間歇性ping成功→可能丟失資料包、瞬時擁塞、速率限制或轉發問題；請繼續資料包丟失分析，因為TCP需要無丟失路徑才能高效運行。
- 如果ping失敗並出現錯誤「需要分段」或超時，則路徑中存在MTU低於1500位元組的鏈路，由於DF位而無法轉發資料包，這表示路徑MTU問題。

如何使用進行疑難排解

- 逐步減小負載大小(例如，1472 → 1400 → 1300)，以標識成功的最大大小。
- 識別後，使用公式MTU = payload + 28位元組（IP + ICMP標頭）計算MTU。

與TCP的實際關聯性

- 如果MTU小於預期，TCP區段可能會分段或遭捨棄。
- 這會導致重新傳輸、延遲增加和吞吐量降低，直接影響應用效能。

標識流量路徑（介面）

要對Cisco Nexus 9000交換機上的TCP效能進行有效的故障排除，必須確定哪些介面正在接收和轉發源與目標之間的流量。

在簡單拓撲中，這可以直接從物理連線推斷。例如，如果使用者端連線到Ethernet1/1，伺服器連線到Ethernet1/2，則流量路徑會非常簡單。但是，在具有多個活動介面、埠通道或vPC配置的真實環境中，這種識別並不總是瑣碎的。

在這種情況下，推薦的方法是使用嵌入式邏輯分析器模組(ELAM)，它可在ASIC（資料平面硬體）級別提供可視性。

ELAM允許您在轉發管道處理資料包時捕獲該資料包，並顯示以下重要資訊：

- 輸入介面
- 輸出介面
- 轉發決策（L2/L3查詢結果）

此方法比依賴控制平面工具準確得多，因為它反映了實際的硬體轉發路徑。

必須注意的是，ELAM一次僅捕獲一個資料包，因此必須精確定義過濾標準以匹配所需的流量（例如，源IP、目標IP、TCP埠）。 如果過濾器範圍過廣，則可能會捕獲不相關的流量，例如ICMP或UDP，而不是預期的TCP流量。

此外，必須為兩個流量方向重複此過程：

- 源→目標
- 目的→源

在使用vPC或ECMP的環境中，流量可以跨多個路徑進行負載均衡。因此，轉發和返回流量可以經過不同的交換機或介面。在這些情況下，必須在每個相關Nexus交換機上執行ELAM以確保完整的可視性。

通過準確識別入口和出口介面，故障排除範圍顯著減小，從而能夠集中驗證介面計數器、QoS策略、MTU設定和沿準確轉發路徑的潛在擁塞點。

ELAM配置（Nexus 9300雲規模）

此範例過濾來源IP為10.93.19.8、目的地IP為10.91.2.35和TCP目的地連線埠445的流量。

ELAM設定

<#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
```

生成流量後，檢索結果：

<#root>

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

report
```

反向流量捕獲（對於完全可視性是必需的）

要驗證返迴路徑，請通過交換源IP地址和目標IP地址重複配置：

```
<#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
```

操作說明

- ELAM僅捕獲一個資料包，因此請確保開始捕獲時流量正在流動。
- 過濾器必須精確，以避免捕獲不相關的流量。
- 在vPC環境中，在兩台交換機上運行ELAM，因為流量可以在每個方向上以不同的雜湊方式運行。
- 輸出顯示硬體中的輸入介面、輸出介面和轉發決策，提供資料平面的權威可視性。

參考

Cisco Nexus 9000雲規模ASIC ELAM指南

介面層級驗證

介面級驗證可確保Nexus交換機不會引入任何影響TCP流量的限制或異常。重點是確認配置、運行狀態和硬體計數器與高效能資料平面轉發的預期行為一致。

組態驗證

- 驗證沒有限制性ACL應用於介面：

<#root>

switch#

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

- 驗證沒有意外的QoS策略影響流量（介面級別和全域性QoS，包括隊列、策略和調節）：

<#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

- 確認第2層或第3層配置（交換機埠與路由介面），包括VLAN成員身份、STP狀態和IP定址：

<#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

運行狀態驗證

- 驗證MTU一致性並確保它與預期配置（例如1500或9000位元組）相匹配：

<#root>

switch#

show interface ethernet1/1-2 | include MTU

- 確認介面速度和雙工設定：

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

- 驗證介面穩定性（無抖動或頻繁的鏈路轉換）：

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

錯誤計數器驗證

- 在測試之前清除計數器：

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

- 監控錯誤計數器（僅非零值）：

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

測試後驗證

- 再次運行TCP流量測試並觀察計數器：

<#root>

switch#

show interface counters errors non-zero | include Port|Eth1/1|Eth1/2

- 計數器不能遞增；任何增加表示潛在的第1層或硬體相關問題，例如物理鏈路錯誤、CRC/FCS錯誤或緩衝區溢位/丟棄。

路由和ARP穩定性

確保路由和ARP穩定性對於確認Nexus交換機具有一致的第3層可達性至關重要，而且不會引入可能會影響TCP效能的間歇性解決方案問題。路由條目或ARP解析中的不穩定可能會導致資料包丟失、延遲增加或流量黑洞。

驗證條件

- 源和目標的路由條目必須存在、穩定，並且不能頻繁更改。
- 必須解析ARP條目，而不能持續刷新或丟失。

<#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

驗證流量是否未傳送到CPU

在Cisco Nexus 9000交換機中，轉發在硬體(ASIC)中執行，而CPU不參與正常的資料平面操作。因此，在控制平面中觀察主機到主機TCP流量是異常的，表明資料包由於異常或配置錯誤而遭到攻擊。一旦流量必須由CPU處理，它就會受到控制平面策略管制，並且如果流量超過允許的控制平面速率，預計可以觀察到丟棄。

驗證方法

- 使用Ethanalyzer捕獲到達控制平面的流量：

<#root>

switch#

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

預期行為

- 在CPU中無法觀察到主機到主機TCP資料平面流量。

意外行為

- 如果與流匹配的資料包可見，則流量將被傳送，這可能是由以下原因造成的：
  - 異常資料包處理（TTL過期、ACL日誌記錄、重定向）
  - 配置錯誤或功能不受支援
  - 硬體程式設計不正確

確定資料包轉發延遲

Nexus 9000交換機的資料包轉發延遲取決於資料包大小、轉發模式和啟用的功能。思科規範通常針對直通轉發下的64位元組資料包參考延遲。

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

- 直通轉發（Nexus 9000中的預設值）：
  - 在收到完整資料包之前開始轉發。
  - 最大程度減少延遲（亞微秒到~1微秒）。
- 存轉：
  - 轉發前必須收到整個資料包。
  - 將延遲與資料包大小成比例。

其他功能可能會引入增量延遲：

- VXLAN封裝/解除封裝
- ACL查詢（TCAM處理）
- QoS分類和佇列
- 遙測(NetFlow、ERSPAN、sFlow)
- 擁塞期間的緩衝

但是：

- 這些操作在硬體管道中執行。

延遲顯著增加的唯一現實情況是擁塞：

- 資料包在出口隊列中緩衝。
- 延遲取決於：
  - 隊列深度
  - 介面利用率
  - QoS策略

即使在這些情況下：

- 延遲通常在微秒到數百微秒的範圍內。
- 持續的毫秒級延遲將意味著：
  - 嚴重擁塞
  - 超額認購
  - QoS或緩衝配置錯誤

SPAN到CPU（資料平面的封包擷取）

這樣可將資料平面流量映象到控制平面以進行資料包捕獲並匯出到.pcapng檔案，從而允許在Wireshark中進行詳細分析。

組態

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

捕獲執行

<#root>

```
switch#
```

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

技術注意事項

- 映象到CPU的流量受控制階段管制(CoPP)的約束。
- 如果流量超過CoPP:
  - 資料包只能在控制平面中丟棄。
  - 這樣會在分析期間產生誤報。
- 對於低到中等流量的情況，建議使用SPAN到CPU。
- 對於高吞吐量環境：
  - 使用本地SPAN（外部分析器）
  - 使用ERSPAN進行遠端擷取

方法	優勢	限制
範圍	準確，無封裝	需要物理連線。
ERSPAN	遠端捕獲功能	易受網路擁塞的影響。

控制平面速率限制驗證

為了確保SPAN到CPU的捕獲是可靠的，必須驗證控制平面是否由於速率限制而丟棄映象資料包。

驗證命令

```
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
```

驗證方法

- 以~3秒的時間間隔執行命令。
- 觀察與SPAN相關的捨棄計數器。

解釋

- SPAN行的捨棄計數器中沒有遞增表示可靠的擷取。
- 丟棄計數器的增加表示控制平面中的資料包丟失，從而使捕獲不可靠。

如果觀察到捨棄專案，必須將擷取方法變更為SPAN或ERSPAN。

TCP之前基於ICMP的驗證

ICMP測試在執行複雜的TCP分析之前提供資料平面完整性的基線驗證。由於ICMP是無狀態且更簡單，因此它允許快速檢測資料包丟失、重複或路徑不一致。

SPAN擷取中的預期行為

- 每個ICMP封包可能會出現兩次：
  - Once on ingress
  - 出口一次
- 對於標準ping:
  - Echo Request → 2 packets
  - Echo Reply → 2 packets

這將確認資料平面中的正確轉發和丟包現象。

異常行為

- 缺少重複項或非對稱資料包計數表示可能存在資料包丟失或捕獲限制。
- 間歇性超時表明存在第1層問題、擁塞或上游問題。

如果ICMP流量連續轉發，而沒有丟失，則很有可能在第2/3層也正確轉發TCP流量。

使用資料包捕獲確定Nexus交換機轉發延遲

使用SPAN到CPU（或SPAN/ERSPAN）擷取流量時，每個封包可以觀察兩次：一次在輸入，一次在輸出。此重複可用於通過計算同一資料包的兩個例項之間的時間差來估計Nexus交換機引入的轉發延遲。

在實踐中，此延遲可以通過比較重複回應請求和回應回複資料包之間的時間增量來使用先前捕獲的ICMP流量進行測量。這為交換機轉發效能提供了簡單而有效的基準。如果需要更深入的分析，同樣的方法也可應用於TCP流量，方法是捕獲流量並測量重複的TCP資料包之間的時間差。

方法

- 識別資料包及其重複項（相同序列號）。
- 測量入口副本和出口副本之間的時間增量。
- 此增量表示交換機轉發延遲的上限估計，因為它可能包括映象和時間戳開銷。

Wireshark配置

- 啟用時間增量顯示：

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

- 為時間增量新增自定義列：

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

- 過濾相關流量（示例）：

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

- 按序列號或TCP資料流對資料包排序：

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

解釋

- 重複資料包之間的時間增量可以處於微秒範圍內。
  - 如果是這種情況，Nexus交換機不會為資料包轉發帶來延遲。
- 一致的低增量確認基於硬體的轉發效能。
- 較高或不一致的增量可能表示：
  - 擁塞或緩衝

參考資料

- Cisco Nexus 9000系列產品手冊
- Cisco Nexus 9000系列交換機設計手冊
- Cisco Nexus 9000系列交換機上的智慧緩衝區管理白皮書

來源主機封包擷取的TCP流量分析

本節提供詳細的方法，用於透過先前所述的假設案例分析Wireshark中的TCP封包擷取（包括設定檔組態）。圖示直接取自Wireshark。提醒一下，情景是：

一位使用者已經發現，對於約6.5 TB的應用程式資料集的備份過程（以前大約在9小時內完成）現在需要將近21小時。唯一可訪問的網路裝置是連線到源伺服器(10.93.19.8)的Cisco Nexus 9300交換機。 交換機介面上配置的MTU為9000位元組（巨型幀），而伺服器上的MTU未知。可從源伺服器捕獲資料包，並且所有之前的Nexus驗證步驟均已完成，未檢測到異常。

主要觀察和限制

- 已排除Nexus交換機：
  - 無資料包丟棄
  - 無介面錯誤
  - 無QoS或ACL影響
  - 硬體轉發已確認
- 介面配置：
  - 接入埠
  - MTU:9000 位元組
- 可用資料：
  - 源位置的資料包捕獲
  - 端到端MTU知識
    - 使用包含1,472位元組資料的1500位元組封包成功完成ping時，不會進行分段。
- 缺少資料：
  - 目標可視性
  - 目標伺服器上沒有可用的資料包捕獲。

在Wireshark中，可以建立定製配置檔案，該配置檔案根據要執行的特定分析型別定製。

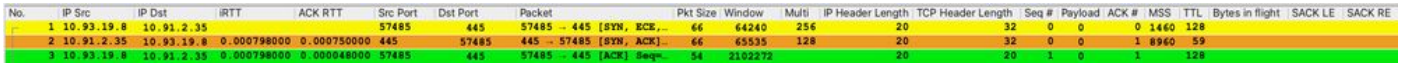
列說明

- tcp.analysis.initial_rtt(iRTT):根據TCP三次握手估計初始往返時間。
- tcp.analysis.ack_rtt(ACK RTT):度量TCP資料段與其相應確認之間的時間。
- tcp.window_size（視窗）：指示應用縮放之前接收方通告的TCP視窗大小。
- tcp.options.wscale.multiplier（多重）：表示用於計算有效接收視窗的視窗縮放係數。
- tcp.seq（序列號）：顯示TCP資料段中第一個位元組的序列號。
- tcp.len（負載）：顯示該網段的TCP負載大小（以位元組為單位）。
- tcp.ack(ACK#):指示來自傳送方的下一個預期位元組（累積確認）。
- tcp.options.mss_val(MSS):顯示在TCP握手過程中通告的最大資料段大小。
- ip.ttl(TTL):顯示「生存時間」值，該值對於標識跳數和路由行為非常有用。
- tcp.analysis.bytes_in_flight（傳輸中的位元組）：表示當前正在傳送的未確認資料的數量。

TCP三次握手分析

必須捕獲TCP三次握手，因為它包含定義會話行為方式的關鍵引數，如MSS、視窗縮放和SACK。如果沒有此資訊，則任何TCP分析都是不完整的，並且可能導致有關效能或根本原因的結論不正確

。



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			

流量識別

在封包擷取中：

- 源IP地址：10.93.19.8
- 目標IP地址：10.91.2.35

初始來回時間(iRTT)分析

初始RTT(iRTT)計算如下：

- iRTT = 798微秒

該值源自：

- 封包2(SYN-ACK)ACK RTT:目標響→SYN的時間為750 µs。
- 資料包3(ACK)ACK RTT:48微秒→源確認SYN-ACK的時間。

大多數延遲(~94%)位於轉發路徑(客戶端→伺服器→客戶端)，而來自源的響應時間最短，表明客戶端上沒有CPU或應用程式延遲。

TCP連線埠識別

- 目的地TCP連線埠：445

埠445對應於Microsoft Server Message Block(SMB)，常用於檔案共用、網路驅動器和Windows身份驗證服務。此通訊協定對延遲和輸送量都很敏感，因此高度依賴TCP效率與網路穩定性。

TCP視窗大小分析

- 源視窗（縮放）：64,240 位元組
- 目標視窗：65,535 位元組

TCP視窗表示在等待確認之前可以傳送的資料量。在這種情況下，源的限制稍高於目的地。這些值對於現代環境來說相對較小，並且會限制吞吐量，特別是在RTT增加時。

最大理論吞吐量可使用以下命令進行估計：

吞吐量= TCP視窗大小/RTT

替換觀察到的值：

- TCP視窗大小= 64,240位元組
- RTT = 798微秒= 0.000798秒

吞吐量≈ 64,240 / 0.000798 ≈ 80.5 MB/s（約644 Mbps）

這表示上限吞吐量，假設：

- 無資料包丟失
- 無重新傳輸
- 理想的網路條件

吞吐量、轉移時間和必需條件分析

在當前吞吐量644 Mbps下，傳輸6.5 TB的檔案大約需要23.5小時，這與觀察到的降級情況相符。要實現9小時的傳輸時段，吞吐量必須提高到大約1.68 Gbps，需要更大的TCP時段（大約增加2.7倍）或顯著更低的RTT（約291微秒）。

在當前條件下（64 KB視窗和~798 µs RTT），不可能達到9小時的目標，因為TCP吞吐量受頻寬延遲產品的限制。如果不增加視窗大小或減少延遲，協定將無法利用更高的可用頻寬，從而使目標無法實現。

案例	吞吐量	估計傳輸時間(6.5 TB)	必需的TCP視窗	必需的RTT
當前狀態	644 Mbps（約80.5 MB/s）	約23.5小時	64千位元	798微秒
目標（9小時）	~1683 Mbps（約210 MB/s）	9小時	約172 KB	~291微秒

這在過去起作用，表示網路、應用程式、源或目標中發生了更改。必須指出的是，僅基於這一初步分析，就可以得出一個重要的結論：在當前TCP視窗大小和RTT條件下，無法實現9小時目標。

下表比較了RTT和TCP視窗大小增加或減少時吞吐量的變化。

RTT對吞吐量的影響（固定視窗大小= 64,240位元組）

RTT	吞吐量（MB/秒）	吞吐量(Mbps)
200微秒（0.0002秒）	約321 MB/秒	約2,568 Mbps
798微秒(0.000798秒)	約80.5 MB/s	約644 Mbps
2毫秒（0.002秒）	約32.1 MB/s	約257 Mbps
10毫秒（0.01秒）	約6.4 MB/s	約51 Mbps

TCP視窗大小影響（固定RTT = 798 µs）

TCP視窗大小	吞吐量（MB/秒）	吞吐量(Mbps)
16 KB（16,384位元組）	約20.5 MB/s	約164 Mbps
64 KB（64,240位元組）	約80.5 MB/s	約644 Mbps
256千位元（262,144位元組）	約328 MB/秒	約2,624 Mbps
1 MB（1,048,576位元組）	約1,314 MB/秒	約10.5 Gbps

技術解釋

- 吞吐量與RTT成反比→延遲降低效能。
- 吞吐量與TCP視窗大小成正比，→大視窗增加容量。
- 小視窗大小嚴重限制了吞吐量，即使在低延遲環境中也是如此。
- 高速網路(10G+)需要視窗擴展以充分利用頻寬。

這表明RTT和TCP視窗大小都是TCP效能的關鍵因素，在排除吞吐量問題時必須一起進行分析。

IP和TCP標頭長度

- IP報頭長度：20 位元組
- TCP報頭長度：32 位元組

20位元組的IP報頭表示不存在IP選項。32位元組的TCP報頭確認正在使用TCP選項，從而在基本報頭之外新增12個位元組。這些選項通常包括MSS、Window Scale和SACK Permitted。

TCP選項分析和TTL

兩個端點上啟用選擇性確認(SACK)。這在圖片中不可見。SACK允許接收者確認非連續資料區塊，並準確通知傳送者哪些區段已成功接收。

例如，如果收到段1000-2000和3000-4000，但缺少2000-3000，則接收方可以明確指出這一點。沒有SACK時，傳送方會重新傳輸間隙後的所有資料；使用SACK時，僅重新傳輸缺失的部分。這顯著提高了在丟包環境中的效能。

封包1(SYN)分析

- 序列號：0（Wireshark標準化）
- 裝載:0 位元組
- ACK#:0
- MSS:1460 位元組
- TTL:128

Wireshark將序列號標準化為零以實現可讀性，但實際上它是一個很大的隨機值。連線建立期間預計沒有負載。MSS值1460位元組表示1500位元組的MTU（20位元組IP標頭+ 20位元組TCP標頭）。TTL 128可以是基於Windows的主機，如果在捕獲中看到此值，則表明捕獲可能在第2層或非常靠近源位置進行。

封包2(SYN-ACK)分析

- ACK#:1

ACK值為1，因為SYN旗標會消耗一個序號，即使沒有負載亦是如此。因此，ACK = SEQ + 1。

- TTL:59

觀測到的TTL 59表示初始TTL為64，表示封包穿越了約5個路由躍點(64 − 59 = 5)。 每個路由躍點將TTL遞減1。

分段風險和網路影響

大約五個路由躍點的存在會引入潛在效能風險，尤其是與MTU不匹配和分段相關的風險。

如果任何中間連結的MTU小於原始封包大小，則可能會進行分段。這會導致以下幾個後果：

- 分段和重組開銷導致延遲增加。
- 資料包丟失的概率更高，因為丟失單個片段需要重新傳輸整個資料包。
- 吞吐量降低，因為TCP會將丟失解釋為擁塞並降低其傳送速率。
- 處理分段的網路裝置上的CPU利用率提高。
- 如果封鎖了ICMP，路徑的MTU探索(PMTUD)失敗的風險，會導致無聲的封包捨棄。

鑑於這些因素，確保路徑的MTU一致或在必要時實施MSS鉗位至關重要。

TCP RTT分析：ACK RTT與初始RTT

當ACK RTT大於iRTT時，這表示與TCP握手期間建立的基線相比，延遲已增加。

這表示網路或端點在作業階段期間引入額外延遲，通常是因為以下原因：

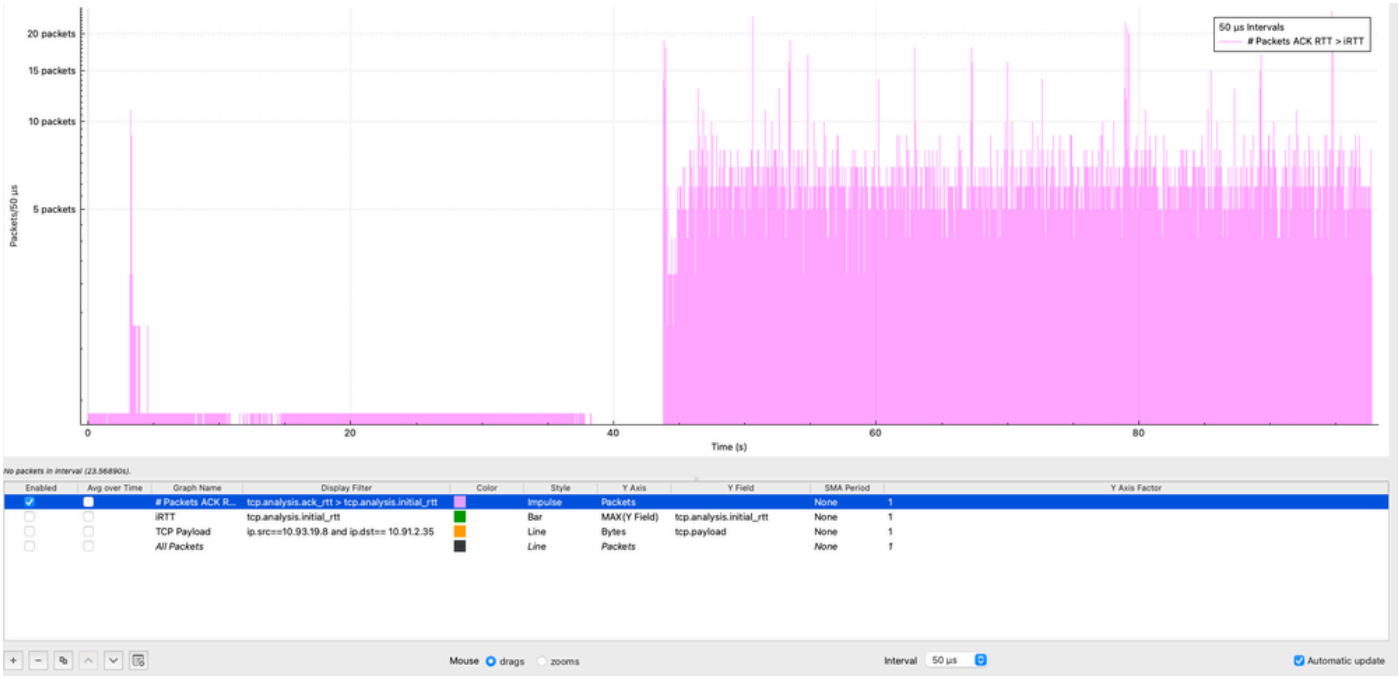
- 網路擁塞或佇列
- 接收器或應用處理延遲
- 中間裝置（防火牆、負載均衡器）
- 重新傳輸

如果此情況在整個TCP會話中持續出現，則會導致：

- 降低TCP吞吐量
- 低效的視窗利用率
- 應用程式效能降低

在Wireshark中，可以使用I/O Graphs（I/O圖表）功能來直觀顯示ACK RTT > iRTT條件的發生頻率：統計→ I/O圖，應用顯示過濾器(tcp.analysis.ack_rtt > tcp.analysis.initial_rtt)，選擇「脈衝」樣式，將Y軸設定為「資料包」，並使用50微秒的時間間隔。

在圖中，紫色脈衝表示每個50微秒間隔內滿足此條件的資料包數。如前所述，此情況在整個資料包捕獲期間持續出現，表明會話期間的延遲始終高於初始基線。此行為強烈表明效能持續下降，而不是瞬變狀態，這更加突出了調查端到端路徑上擁塞、緩衝或端點處理延遲等潛在來源的必要性。

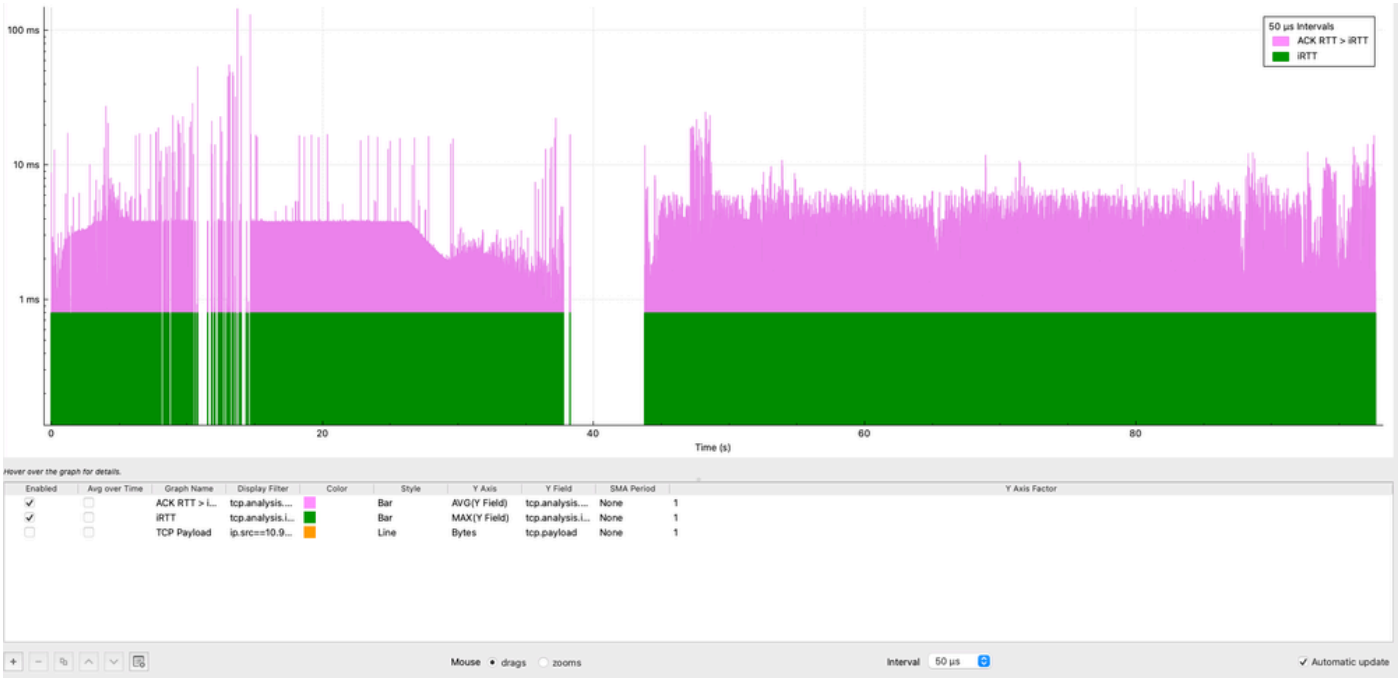


確定iRTT被超出的時間也很重要，而不僅僅是頻率。雖然Wireshark不允許直接在欄位之間減法，但可以使用I/O圖形進行直觀比較：

- 導航到Statistics → I/O Graphs
- 圖1:
  - 顯示篩選條件:tcp.analysis.ack_rtt > tcp.analysis.initial_rtt
  - 樣式：欄
  - Y軸：平均值
  - Y欄位：tcp.analysis.ack_rtt
  - Interval:50微秒
- 圖2:
  - 顯示篩選條件:tcp.analysis.initial_rtt
  - 樣式：欄
  - Y軸：最大值
  - Y欄位：tcp.analysis.initial_rtt
- 然後按一下右鍵圖形並啟用Log scale。

在此視覺化顯示中，紫色圖形表示條件ACK RTT > iRTT，該條件在整個TCP會話中始終存在。資料顯示持續的延遲膨脹，多個峰值達到11毫秒，最大峰值超過100毫秒，表示基線iRTT的11倍到100倍。

此行為可確認延遲增加不是暫時的，而是持續的，這表明存在系統問題，會隨著時間的推移影響會話。這種持續偏差強烈表明存在網路擁塞、緩衝(bufferbloat)或終端處理延遲等因素。



TCP重傳與偽重傳分析

本節通過分析一段時間的重新傳輸來評估TCP可靠性，從而驗證資料包丟失是否會導致效能下降。

隨時間變化的TCP重新傳輸

圖中顯示了TCP重傳隨時間變化的分佈。總共觀察到42次重傳，僅佔總流量的0.00125%。

這種重傳級別可以忽略不計，清楚地表明資料包丟失並不是此場景中的起因。

Wireshark配置（TCP重新傳輸）

```
Statistics → I/O Graphs
```

- 顯示篩選條件:

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

- 樣式：衝動或條形圖
- Y軸：資料包

- Interval:1秒

TCP虛假重傳

該圖顯示源10.93.19.8在1秒間隔內生成的TCP偽重新傳輸數。

在Wireshark中，TCP虛假重傳表示主機重新傳輸了實際上並未丟失的資料段。原始資料包成功到達接收方，但是傳送方由於不準確的定時估計而錯誤地假定丟失。此行為並不表示真正的丟包，而是表明傳送方存在低效的重傳邏輯。

在此擷取中：

- 源10.93.19.8在大約8微秒後重新傳輸資料包。
- 而典型的重新傳輸計時器大約是200毫秒。

這確認重新傳輸行為完全由源TCP堆疊控制，而不是由網路控制。

觀察到的虛假重傳總數為1,112，代表總捕獲流量的0.0332%。

Wireshark配置（TCP偽重傳）

```
Statistics → I/O Graphs
```

- 顯示篩選條件:

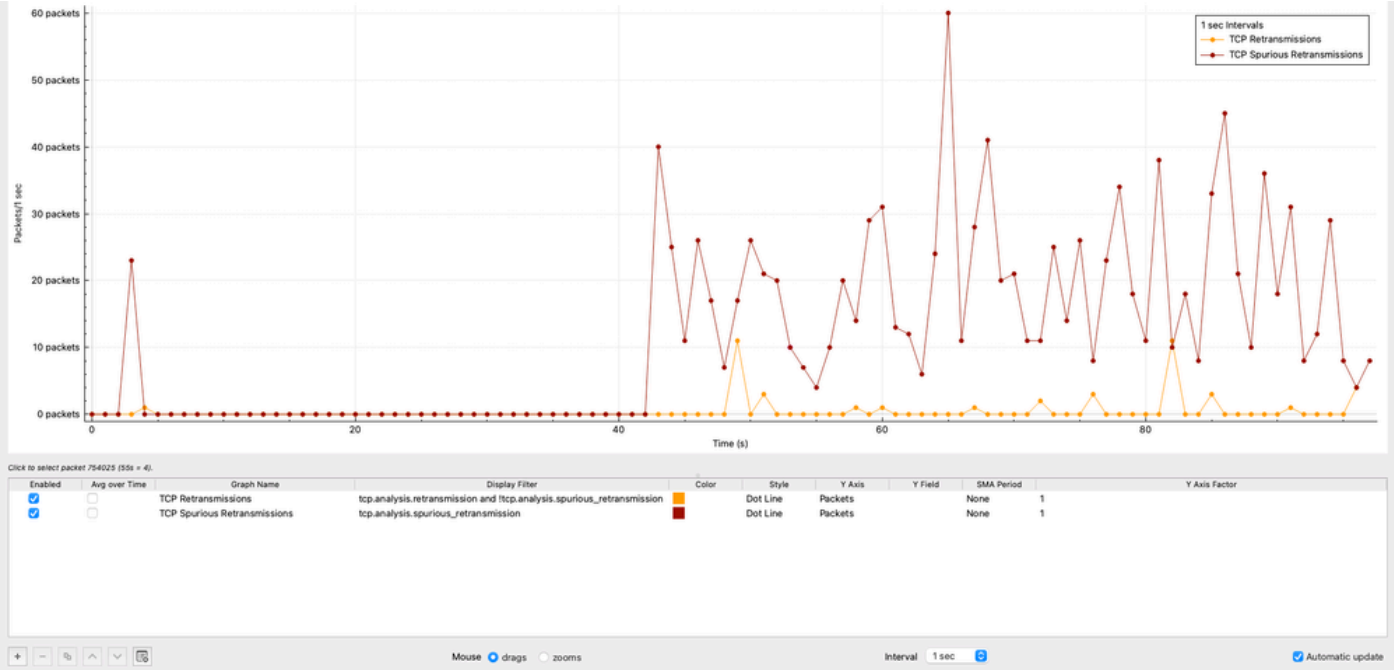
```
tcp.analysis.spurious_retransmission and ip.src==10.93.19.8
```

- 樣式：衝動或條形圖
- Y軸：資料包
- Interval:1秒

技術解釋

- 實際重新傳輸的百分比極低，表明網路中不存在丟包現象。
- 假重傳的存在表示源主機過早地作出重傳決定。
- 此行為可能會輕微影響效率，但並不是導致吞吐量嚴重下降的主要原因。

此分析進一步強調，此問題與網路可靠性無關，而與TCP行為、延遲或終端效能有關。

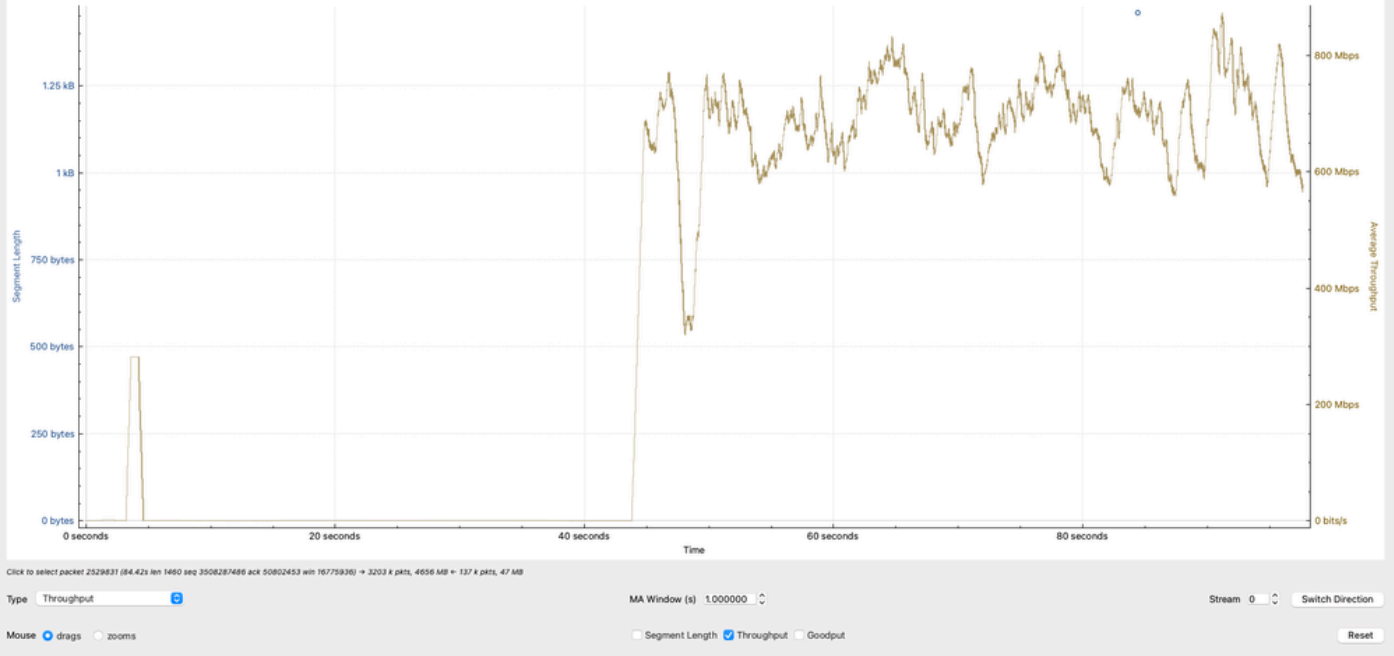


有效吞吐量分析

該圖顯示了基於TCP負載（實際傳輸的資料）計算的有效吞吐量，單位為每秒Mb。觀察到的吞吐量主要在600 Mbps和800 Mbps之間振盪，這表明網路在主動傳輸資料的同時，並未達到更高的頻寬潛力。

Wireshark配置（有效吞吐量）

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



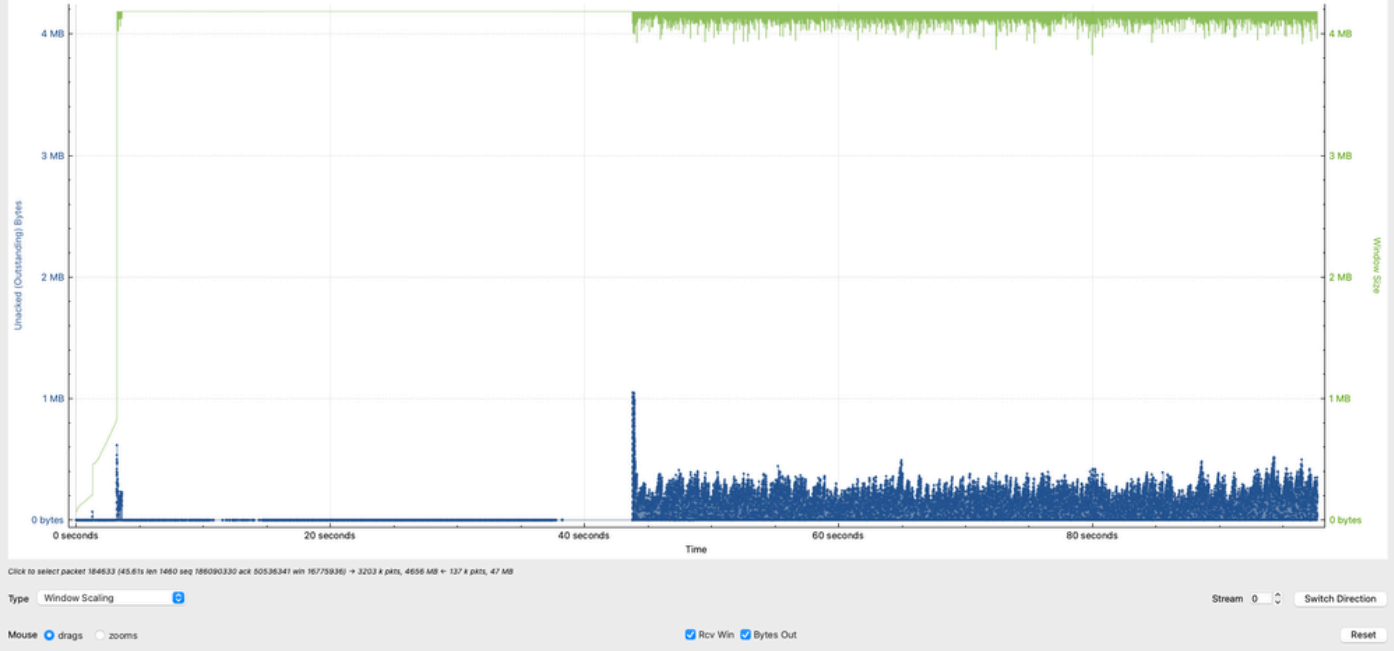
技術解釋

- 600-800 Mbps的吞吐量範圍與基於TCP視窗大小和RTT的先前計算一致。
- 吞吐量變化反映出：
  - RTT波動
  - TCP擁塞控制調整
  - 應用程式調步或緩衝
- 由於吞吐量不接近線速（例如，10G），因此限制不是物理頻寬，而是TCP效率限制。
- 此分析確認觀察到的吞吐量與TCP限制（視窗大小和延遲）一致，進一步表明瓶頸不是由於資料包丟失或介面容量所致，而是由於傳輸層行為和終端條件所致。

飛行資料（TCP視窗）分析

該圖通過比較接收方容量與傳輸中的實際資料（傳輸中的位元組數），突出顯示TCP會話中的一個關鍵行為。

- 綠線表示10.91.2.35（接收方）可以接受的TCP資料量（有效接收視窗）。
- 藍線表示當前從10.93.19.8（傳送方）傳輸的TCP資料量。



觀測到的Data in Flight峰值在1 MB左右，在8 KB和5 KB附近還有峰值，但主要集中在1 KB和250 KB之間。

這表明，雖然接收方能夠處理更大量的資料，但傳送方並沒有始終如一地利用可用視窗。

Wireshark配置（飛行中的資料與視窗）

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

技術解釋

- 接收機(10.91.2.35)通告一個明顯更大的視窗，表明它能夠接收更多的資料。
- 傳送方(10.93.19.8)未充分利用可用視窗，如飛行資料值中較低和不一致的值所示。
  - 傳送者理想地可以將Data in Flight值維持在更靠近接收者通告視窗(~1 MB)的位置，以最大限度地提高吞吐量。
  - 無法維持高飛行資料級別直接限制吞吐量，它強烈表明源位置的TCP效率低下，而不是網路容量問題。

TCP負載與MSS隨時間變化的分析

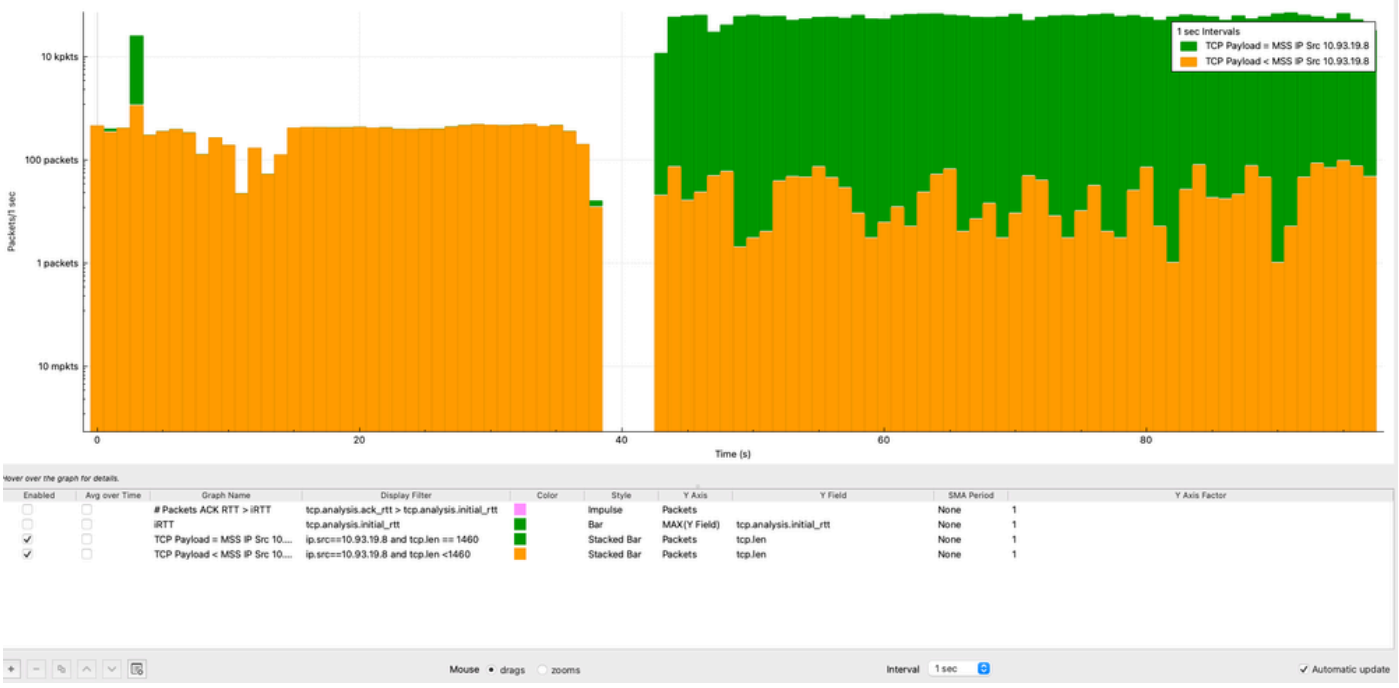
根據MSS分析隨時間變化的TCP負載大小，有助於判斷傳送者是否高效地利用每個TCP區段。此分析是從來源IP位址(10.93.19.8)的角度執行的。

在Wireshark中，圖配置如下：

- 圖1（MSS大小的資料包）：
  - 顯示篩選條件:ip.src==10.93.19.8和tcp.len == 1460
  - 樣式：堆積條形圖
  - Y軸：資料包
  - Interval:1秒
- 圖2(所有資料包≤MSS):
  - 顯示篩選條件:ip.src==10.93.19.8和tcp.len <= 1460
  - 樣式：堆積條形圖
  - Y軸：資料包
  - Interval:1秒
- 應用對數刻度以獲得更佳的可視性

根據分析：

- 大多數資料包（>10,000個資料包/秒）始終達到1460位元組的MSS值。
- 由於一般TCP行為（ACK、分段或串流結束資料），較小部分封包攜帶的負載較少。



根本原因分析(RCA):TCP效能下降

此分析證明，確定TCP效能問題的根本原因需要全面的端到端方法，而不是假設網路是效能下降的主要來源。

在Cisco Nexus 9300交換機上執行廣泛的驗證，包括介面計數器、QoS策略、路由和ARP穩定性、CPU點數驗證、基於SPAN的資料包捕獲以及使用ELAM的ASIC級轉發驗證。所有結果一致證實交換機在預期引數範圍內運行：

- 無資料包丟棄
- 無異常延遲（微秒範圍）
- 無QoS或控制平面影響
- 正確硬體轉發

此外，TCP分析還顯示：

- 可忽略的重新傳輸(0.00125%)
- 沒有資料包丟失的證據
- 源位置的MSS利用率一致
- 吞吐量與TCP視窗和RTT約束保持一致
- 可用TCP視窗利用不足（飛行分析中的資料）
- 網路不是瓶頸
- 源伺服器正在限制效能

結論

效能下降是由於源伺服器在支援Jumbo的環境中使用MTU 1500操作而導致的，從而阻止了有效使用可用網路容量。

解決方案

將源伺服器上的MTU從1500位元組增加到9000位元組，以便與目標和網路基礎架構保持一致。優勢：

- 啟用較大的TCP資料段
- 降低資料包開銷
- 提高整體吞吐量

技術思考

此分析的主要內容是在排除網路效能故障時避免過早總結的重要性。雖然最初將問題歸咎於網路是很常見的，但此案例清楚地表明網路在整個資料平面路徑中運行正常。只有從源和目標兩個角度執行深入的TCP分析（包括握手引數、RTT行為、視窗利用率、重新傳輸和負載效率），才能準確識

別真正的瓶頸。

花時間詳細分析TCP行為，可防止誤診，減少不必要的網路更改，並確保糾正工作針對實際根本原因。

關於此翻譯

思科已使用電腦和人工技術翻譯本文件，讓全世界的使用者能夠以自己的語言理解支援內容。請注意，即使是最佳機器翻譯，也不如專業譯者翻譯的內容準確。Cisco Systems, Inc. 對這些翻譯的準確度概不負責，並建議一律查看原始英文文件（提供連結）。