排除Nexus 9000(NX-OS)上的TCP性能故障

目录

简介

本文档介绍TCP基础知识、Wireshark深度数据包分析和用于优化端到端性能的实用故障排除。

先决条件

要求

Cisco 建议您了解以下主题：

- IP/TCP

使用的组件

本文档中的信息基于以下软件和硬件版本：

- Cisco Nexus 9000云扩展与Cisco NX-OS 10.6(X)。

注意：有关第三方软件或硬件的配置和互操作性的任何问题均不在思科支持范围内。使用

第三方工具是展示您对思科设备的配置和操作的最佳方式。

本文档中的信息都是基于特定实验室环境中的设备编写的。本文档中使用的所有设备最初均采用原始（默认）配置。如果您的网络处于活动状态，请确保您了解所有命令的潜在影响。

背景信息

什么是TCP

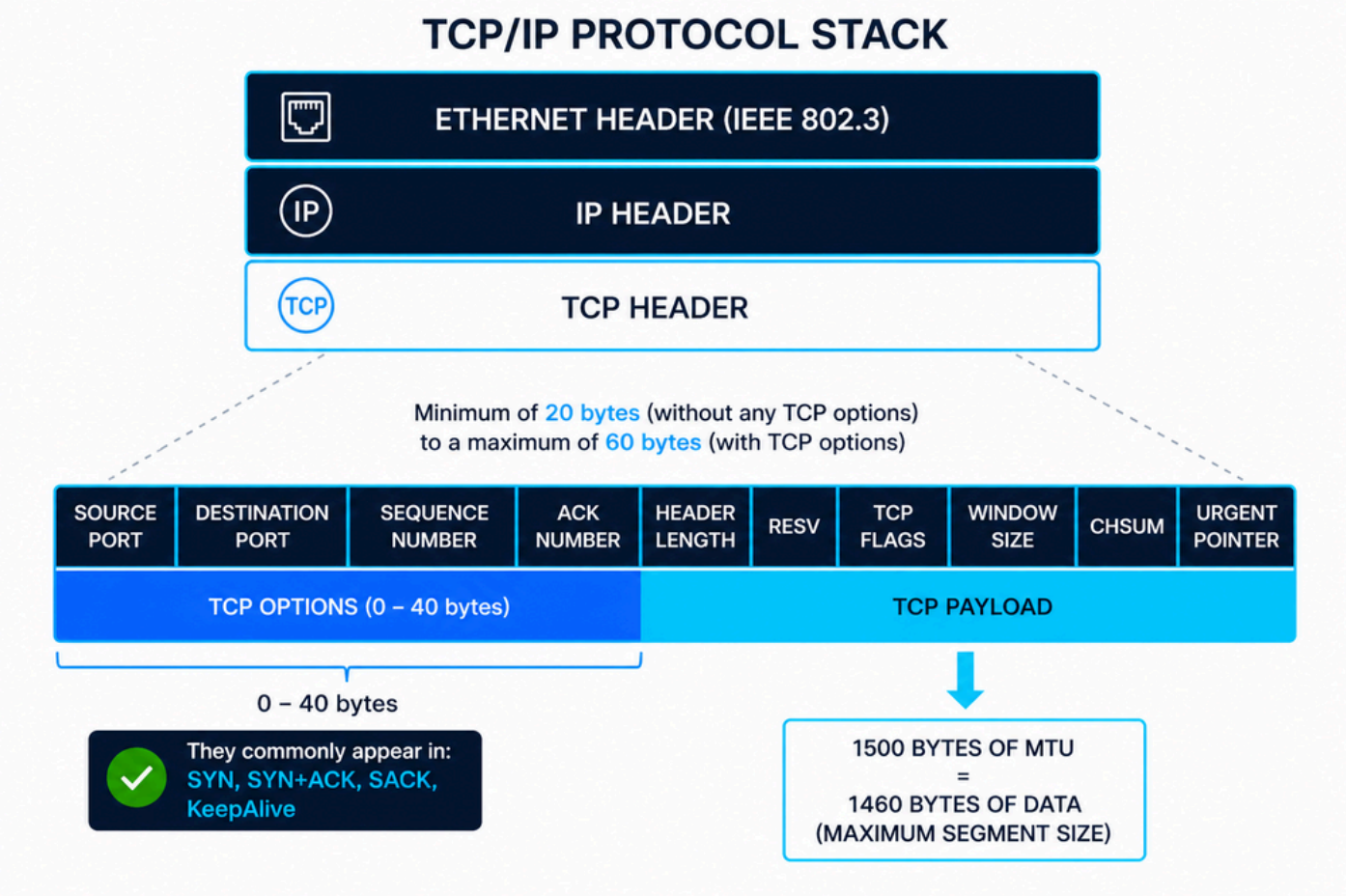
传输控制协议(TCP)是一种基本的传输层协议，在OSI模型的第4层运行，通过IP网络通信的应用之间提供可靠、有序且错误校验的字节流传输。

三大主要优势

1. 可靠性：TCP面向连接，通过要求接收方确认来保证传输。如果数据包在传输过程中丢失或损坏，TCP将自动重新传输数据以确保其到达目的地。
2. 订购交付：由于网络数据包的到达顺序可能混乱，TCP会为每个数据段分配序列号。这允许接收系统按最初发送的确切顺序重组数据。
3. 流量和拥塞控制：TCP动态管理数据传输速率，以满足接收方处理能力和当前网络条件，防止缓冲区溢出或网络拥塞导致的数据丢失。

TCP/IP封装概述

图中展示了TCP/IP协议栈，其中TCP数据段（第4层）封装在IP数据包（第3层）中，然后封装在IEEE 802.3定义的以太网帧（第2层）中。这种分层方法可确保模块化通信，其中每一层都添加自己的控制信息（报头）以确保传输、路由和数据完整性。



以太网报头(IEEE 802.3)

以太网报头通常为14个字节，包括：

- 目的MAC地址（6字节）
- 源MAC地址（6个字节）
- EtherType/长度（2字节）

此外，以太网帧还包含一个4字节的帧校验序列(FCS)帧尾，用于第2层进行错误检测。IEEE 802.3定义了成帧、最小/最大帧大小和物理传输限制，这些限制会直接影响TCP等上层协议。

IP报头(IPv4)

IPv4报头的最小大小为20字节，可通过选项扩展至60字节。关键字段包括：

- 源和目的 IP 地址
- 生存时间 (TTL)
- 协议（将TCP标识为负载）

IP层负责网络中的逻辑编址和路由，但不保证可靠性。

TCP报头结构

TCP报头的范围是20到60字节，具体取决于选项。关键字段包括：

- 源/目标端口
- 序列号
- 确认号
- 标志（SYN、ACK、FIN、RST等）
- 窗口大小
- 校验和

TCP为IP通信添加了可靠的传输、适当的排序和流量控制。

TCP选项（常用10）

TCP选项扩展了基本协议。最常见包括：

1. Maximum Segment Size(MSS) — 定义主机可以接受的最大TCP负载。
2. Window Scale — 将接收窗口扩展到65,535字节以上。
3. Selective Acknowledgment Permitted(SACK Permitted) — 启用选择性确认功能。
4. 选择性确认(SACK) — 指定接收的数据块以避免完全重新传输。
5. 时间戳 — 用于RTT计算和防止包装序列号(PAWS)。
6. No-Operation(NOP) — 用于选项对齐的填充。
7. 选项列表结束(EOL) — 标记TCP选项的结束。
8. TCP快速开放(TFO) — 允许在初始握手期间交换数据。
9. 多路径TCP(MPTCP) — 为单个TCP会话启用多个网络路径。
10. 用户超时选项(UTO) — 控制传输的数据可以保持未确认状态的时长。

TCP序列和确认行为（包括SYN/FIN）

SYN和FIN标志都使用一个序列号，即使没有负载时也是如此。TCP使用面向字节的排序模型运行，其中传输的每个字节（以及特定的控制标志）都会增加序列空间。此行为对于数据包捕获中的准确TCP分析和诊断排序或确认不一致至关重要。

ACK = SEQ +负载长度+(SYN ?1:0)+(FIN ?1:0）

其中：

- SEQ =初始序列号
- 负载长度=以字节为单位的数据大小
- SYN ?1:如果设置了SYN标志，则0 =添加1，否则为0
- FIN ?1:如果设置了FIN标志，则0 =添加1，否则为0
- ACK =下一个预期字节

示例 1：带数据的SYN（TCP快速打开）

- SEQ = 1000
- SYN = 1
- 负载长度= 200字节

ACK计算：

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

这反映在TCP握手期间发送数据的场景。负载和SYN标志都消耗序列空间。

示例 2：带数据的FIN（连接终止）

- SEQ = 3000
- FIN = 1
- 负载长度= 150字节

ACK计算：

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

这表明TCP可以在连接中断期间包含数据，并且负载和FIN标志会增加序列号。

MSS及其与MTU的关系

最大分段大小(MSS)定义了TCP在分段中可以发送的最大负载。

- 典型以太网MTU = 1500字节

- MSS = MTU − IP报头−TCP报头
- 标准MSS = 1460字节(1500 − 20 − 20)

如果存在TCP选项，则MSS会相应减少。MSS在TCP三次握手期间协商并防止IP层分段。

MSS协商在TCP三次握手中的工作方式

在TCP三次握手期间，使用SYN数据包中的MSS选项交换最大分段大小(MSS):

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

双方都有效地在说：

这是接受的最大TCP负载。

关键规则：MSS是定向的

MSS不会作为单一议定值进行协商。

相反：

- 每台主机都使用另一端通告的MSS。
- 这将创建两个独立的限制，每个方向一个。

因此：

- A使用B的MSS发送数据。
- B使用A的MSS发送数据。

源设备发送的TCP负载是否大于目的MSS?

在正常运行的TCP协议栈中：No.

- 发送方必须尊重接收方通告的MSS。
- 发送较大的数据段可能带来以下风险：
    - IP分段（如果超出MTU）

    - 数据包丢弃（如果分段被阻止或不受支持）
- 这将导致：
    - 重新传输
    - 性能下降
    - PMTUD（路径MTU发现）黑洞等问题

故障排除的实用见解

- 请务必检验TCP三次握手（SYN/SYN-ACK数据包）中的MSS值。
- 检查由下列原因造成的不匹配：
    - 隧道(VXLAN、GRE、IPsec)
    - 修改MSS（MSS夹紧）的防火墙
- 在Cisco NX-OS等平台上，MSS调整通常用于防止在封装路径之间分段

窗口大小（流量控制）

Window Size定义接收方在不确认的情况下可以接受的数据量。

它是什么：

- 一种防止缓冲区溢出的流量控制机制。

目的:

- 确保发送方不会压垮接收方。

从何处获取：

- 在数据包捕获中可见（例如，Wireshark）。
- 派生自OS TCP堆栈配置和缓冲区大小。

供应商/操作系统差异：

- 不同的实施(Linux、Windows、Cisco NX-OS)使用动态扩展和缓冲区调整，导致窗口大小不同。

零窗口条件：

- 当Window Size = 0时，接收器缓冲区已满。
- 发送方暂停传输并发送定期探测。

可变Windows机制

- 基于速率的流量控制
  - 它为发送方分配一个固定数据速率，并确保数据不会超过该分配。
  - 是流应用的理想选择。
  - 广播和组播传输
- 基于窗口的流量控制
  - 窗口大小随时间而变化。
  - 接收方通过向发送方窗口更新发送允许窗口信令来实现流量控制。

故障排除使用：

- 接收器端瓶颈→CPU、内存、应用程序)窗口较小或为零。
- 大窗口但吞吐量低→网络问题（延迟、拥塞）。
- 分析窗口行为对于诊断TCP会话中的性能问题至关重要。

Cisco Nexus 9000(NX-OS)上的TCP数据平面故障排除

本节介绍一种实用的方法，用于诊断运行NX-OS的Cisco Nexus交换机是否影响TCP流量转发或引入性能问题。通过一个假设的场景给出了该方法。

当观察到TCP延迟或性能下降时，通常首先会怀疑是网络导致延迟或性能下降。但是，必须通过数据驱动分析验证此假设。TCP故障排除的授权方法是数据包捕获，最好执行：

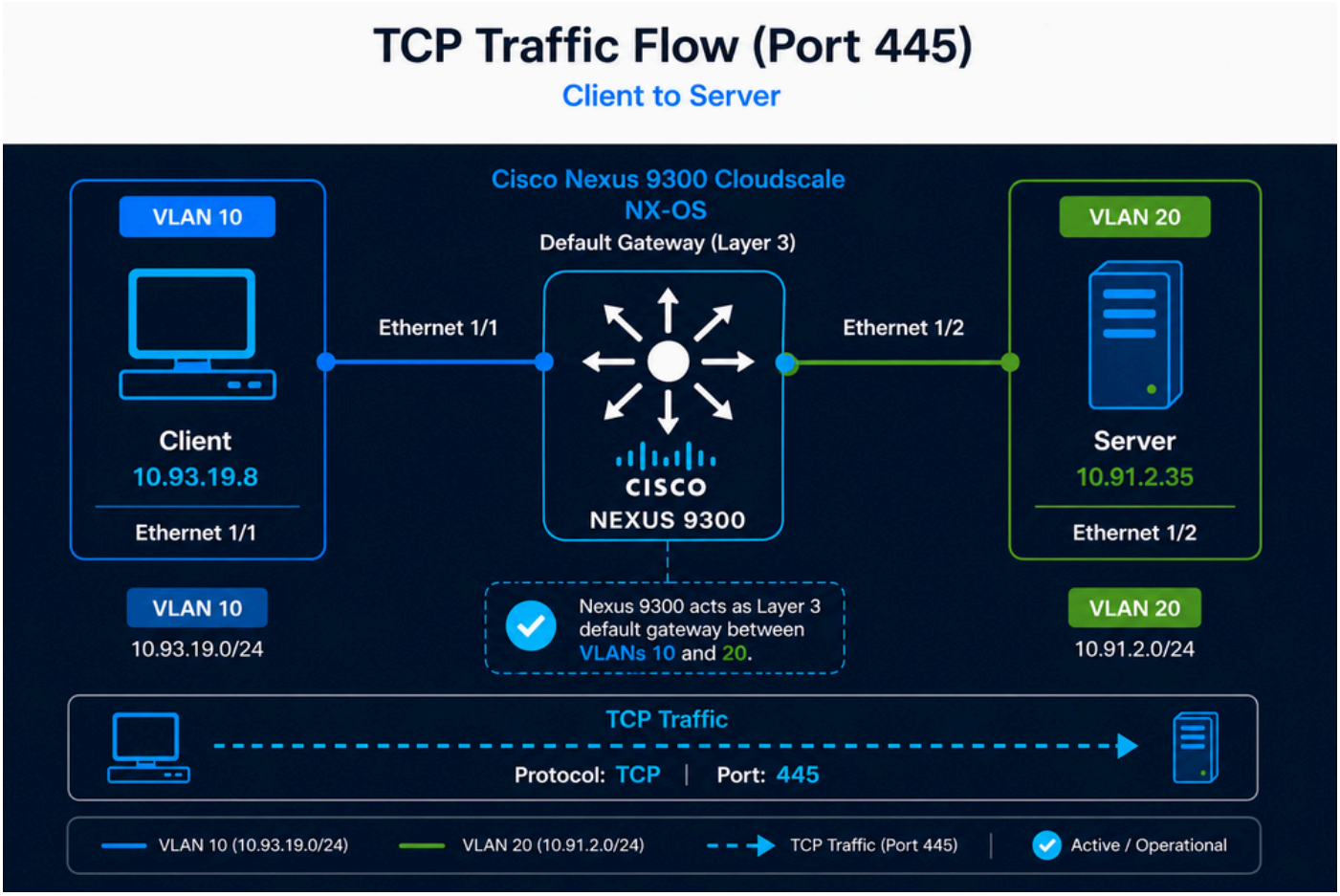
- 同时在源和目的地
- 在流量启动之前

这样可以确保对TCP三次握手的可视性，其中MSS、Window Scale和SACK等关键参数是经过协商的，不会在会话后期重复这些参数。如果无法进行同步捕获，则分析可以只进行一次捕获，但结论有限。

方案定义

一位用户已经发现，对大约7.5 TB的应用程序数据集的备份过程（以前大约在9小时内完成）现在需要将近21个小时。虽然客户端和服务器之间的TCP会话仍然成功建立，但备份持续时间的大幅增加表明吞吐量或整体TCP性能可能会降低。由于Nexus交换机是路径中唯一的网络设备，并且还提供第3层网关功能，因此网络管理员怀疑Nexus交换机是问题的原因。

- 客户端：10.93.19.8(VLAN 10)
- 服务器：10.91.2.35(VLAN 20)
- Nexus 9300充当默认网关
- TCP端口445



初始验证（可达性）

- 这些命令用于通过发送设置了不分片(DF)位的ICMP数据包，验证源与目标之间的路径MTU(PMTU)。这有助于确定在不分段的情况下可以遍历网络的最大数据包大小。此过程必须在源和目标上执行。
- 始终检验源和目标上物理接口的MTU。
- 在此场景中，访问仅可用于确定了MTU为1500的源主机。

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

- -c 10 -→送10个ICMP回应请求
- -I 192.168.10.10 — 使→此特定源IP/接口
- -s 1472 — 将→负载大小设置为1472字节

- -M do →设置DF（不分段）位
- 192.168.20.20 -→标IP

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

- -n 10 -→送10个ICMP回应请求
- -l 1472 — 将→负载大小设置为1472字节
- -f →设置不分段(DF)标志
- 192.168.20.20 -→标IP

为什么选择1472字节？

- ICMP负载= 1472字节
- IP报头= 20字节
- ICMP报头= 8字节
- 总数据包大小：1472 + 20 + 8 = 1500字节（标准MTU）
- 这将测试路径是否支持没有分段的1500字节MTU。如果您尝试发送1500字节的ICMP负载，ping可能会失败，因为在添加IP和ICMP报头后，数据包总大小将超过标准MTU。

可以得出的结论

- 如果ping成功（无丢包），则路径至少支持1500字节的MTU，并且不需要分段。
  - 清除ICMP结果→继续进行TCP分析
  - 间歇性ping成功→可能丢失数据包、瞬时拥塞、速率限制或转发问题；由于TCP需要无丢包路径才能高效运行，因此请继续数据包丢失分析。
- 如果ping失败并出现错误“需要分片”或超时，则路径中存在MTU小于1500字节的链路，由于DF位而无法转发数据包，这表示路径MTU问题。

如何使用它进行故障排除

- 逐渐减小负载大小(例如，1472 → 1400 → 1300)以标识成功的最大大小。
- 确定后，使用公式MTU =负载+ 28字节（IP + ICMP报头）计算MTU。

与TCP的实际相关性

- 如果MTU小于预期，TCP数据段可以分段或丢弃。
- 这会导致重新传输、延迟增加和吞吐量降低，直接影响应用性能。

确定流量路径（接口）

要对Cisco Nexus 9000交换机上的TCP性能进行有效的故障排除，必须确定哪些接口正在接收和转发源和目标之间的流量。

在简单拓扑中，这可以直接从物理连接推断。例如，如果客户端连接到Ethernet1/1，而服务器连接到Ethernet1/2，则流量路径非常简单。但是，在具有多个活动接口、端口通道或vPC配置的真实环境中，这种识别并不总是微不足道。

在这种情况下，推荐的方法是使用嵌入式逻辑分析器模块(ELAM)，它提供在ASIC（数据平面硬件）级别的可视性。

ELAM允许您在转发管道处理数据包时捕获该数据包，并揭示如下重要信息：

- Ingress 接口
- Egress 接口
- 转发决策（L2/L3查找结果）

这种方法比依赖控制平面工具准确得多，因为它反映了实际的硬件转发路径。

请注意，ELAM一次仅捕获一个数据包，因此必须精确定义过滤条件以匹配所需的流量（例如，源IP、目标IP、TCP端口），这一点非常重要。 如果过滤器过于宽泛，可能会捕获不相关的流量，例如ICMP或UDP，而不是预期的TCP流量。

此外，必须针对两个流量方向重复此过程：

- Source → Destination
- 目标→源

在使用vPC或ECMP的环境中，流量可以在多条路径之间实现负载均衡。因此，转发和返回流量可以流经不同的交换机或接口。在这些情况下，必须在每个相关Nexus交换机上执行ELAM以确保完整的可视性。

通过准确识别入口和出口接口，故障排除范围显着减小，从而能够集中验证接口计数器、QoS策略、MTU设置和沿准确转发路径的潜在拥塞点。

ELAM配置（Nexus 9300云扩展）

本示例过滤源IP地址为10.93.19.8、目的IP地址为10.91.2.35和TCP目的端口为445的流量。

ELAM设置

<#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 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
```

生成流量后，检索结果：

<#root>

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

report
```

反向流量捕获（完全可视性所必需的）

要验证返回路径，请通过交换源IP地址和目标IP地址重复配置：

<#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
```

操作说明

- ELAM仅捕获一个数据包，因此请确保在开始捕获时流量正在流动。
- 过滤器必须精确，以避免捕获不相关的流量。
- 在vPC环境中，在两台交换机上运行ELAM，因为流量可以在每个方向以不同的散列方式散列。
- 输出显示硬件中的入口接口、出口接口和转发决策，提供数据平面的权威可视性。

参考

Cisco Nexus 9000云扩展ASIC ELAM指南

接口级验证

接口级验证可确保Nexus交换机不会引入任何影响TCP流量的限制或异常。重点是确认配置、运行状态和硬件计数器与高性能数据平面转发的预期行为一致。

配置验证

- 检验接口上是否未应用任何限制性ACL:

<#root>

switch#

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

- 验证没有意外的QoS策略影响流量（接口级别和全局QoS，包括排队、策略和整形）：

<#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

- 确认第2层或第3层配置（交换机端口与路由接口），包括VLAN成员、STP状态和IP编址：

```
<#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
```

运行状态验证

- 验证MTU一致性并确保它与预期配置（例如，1500或9000字节）匹配：

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

- 确认接口速度和双工设置：

<#root>

switch#

show interface ethernet1/1-2 | include speed|duplex

- 验证接口稳定性（无抖动或频繁的链路转换）：

<#root>

switch#

show interface ethernet1/1-2 | include rate|flap

错误计数器验证

- 在测试之前清除计数器：

<#root>

switch#

clear counters interface all

- 监控错误计数器（仅非零值）：

<#root>

switch#

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

测试后验证

- 重新运行TCP流量测试并再次观察计数器：

<#root>

switch#

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

- 计数器不得增加；任何增加表示潜在的第1层或硬件相关问题，例如物理链路错误、CRC/FCS错误或缓冲区溢出/丢弃。

路由和ARP稳定性

确保路由和ARP稳定性对于确认Nexus交换机具有一致的第3层可达性至关重要，而且不会引入可能影响TCP性能的间歇性解决问题。路由条目或ARP解析中的不稳定可能导致数据包丢失、延迟增加或流量黑洞。

验证条件

- 源和目标的路由条目必须存在、稳定且不能频繁更改。
- 必须解析ARP条目，而不能持续刷新或丢失。

<#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

验证流量未传送到CPU

在Cisco Nexus 9000交换机中，转发在硬件(ASIC)中执行，CPU不参与正常的数据平面操作。因此，在控制平面中观察主机到主机TCP流量是异常的，表明数据包由于异常或配置错误而被传送。一旦流量必须由CPU处理，它就会受到控制平面策略管制，并且如果流量超过允许的控制平面速率，预计可以观察到丢弃。

验证方法

- 使用Ethanalyzer捕获到达控制平面的流量：

<#root>

switch#

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

预期行为

- 在CPU中无法观察到主机到主机TCP数据平面流量。

意外行为

- 如果与流匹配的数据包可见，则流量将被传送，这可能是由以下原因造成的：
  - 异常数据包处理（TTL过期、ACL日志记录、重定向）
  - 配置错误或功能不受支持
  - 不正确的硬件编程

确定数据包转发延迟

Nexus 9000交换机中的数据包转发延迟取决于数据包大小、转发模式和启用的功能。思科规范通常参考直通转发下64字节数据包的延迟。

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

- 直通转发（Nexus 9000中的默认设置）：
  - 在收到完整数据包之前开始转发。
  - 最大程度地减少延迟（亚微秒到~1微秒）。
- 存储转发：
  - 转发之前必须收到整个数据包。
  - 将延迟与数据包大小成正比。

其他功能可能会增加延迟：

- VXLAN封装/解封
- ACL查找（TCAM处理）
- QoS分类和排队
- 遥测(NetFlow、ERSPAN、sFlow)
- 拥塞时的缓冲

但是：

- 这些操作在硬件管道中执行。

延迟显着增加的唯一现实情况是拥塞：

- 数据包在出口队列中缓冲。
- 延迟取决于：
  - 队列深度
  - 接口利用率
  - QoS 策略

即使在这些情况下：

- 延迟通常在微秒到数百微秒的范围内。
- 持续的毫秒级延迟将意味着：
  - 严重拥塞
  - 超订用
  - QoS或缓冲配置错误

SPAN到CPU（数据平面的数据包捕获）

这样可将数据平面流量镜像到控制平面以捕获数据包，并导出到.pcapng文件，从而允许在Wireshark中进行详细分析。

配置

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

捕获执行

<#root>

```
switch#
```

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

技术注意事项

- 镜像到CPU的流量受控制平面策略(CoPP)的约束。
- 如果流量超过CoPP:
  - 只能在控制平面中丢弃数据包。
  - 这会在分析期间造成误报。
- 对于中低流量方案，建议使用SPAN到CPU。
- 对于高吞吐量环境：
  - 使用本地SPAN（外部分析器）
  - 使用ERSPAN进行远程捕获

方法	优势	限制
SPAN	准确，无封装	需要物理连接。
ERSPAN	远程捕获功能	易受网络拥塞的影响。

控制平面速率限制验证

为了确保SPAN到CPU的捕获是可靠的，必须验证控制平面是否因速率限制而丢弃镜像数据包。

验证命令

```
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
```

验证方法

- 以~3秒的间隔执行命令。
- 观察与SPAN相关的丢弃计数器。

解释

- SPAN行的丢弃计数器中没有递增表示捕获可靠。
- 丢弃计数器增加，表示控制平面中的数据包丢失，从而使捕获不可靠。

如果观察到丢包，必须将捕获方法更改为SPAN或ERSPAN。

TCP之前的基于ICMP的验证

ICMP测试在执行复杂的TCP分析之前提供数据平面完整性的基线验证。由于ICMP是无状态且更简单，因此它允许快速检测数据包丢失、重复或路径不一致。

SPAN捕获中的预期行为

- 每个ICMP数据包可以出现两次：
  - 进入入口一次
  - 出口一次
- 对于标准ping:
  - 回应请求→ 2个数据包
  - 回应应答→2个数据包

这可以确认数据平面中的转发正确且不存在丢包现象。

异常行为

- 缺少重复信息包或不对称信息包计数表示可能存在信息包丢失或捕获限制。
- 间歇超时表明存在第1层问题、拥塞或上游问题。

如果ICMP流量始终转发而不会丢失，则很有可能第2/3层也正确转发TCP流量。

使用数据包捕获确定Nexus交换机转发延迟

当使用SPAN捕获到CPU（或SPAN/ERSPAN）的流量时，可以观察到每个数据包两次：一次在入口上，一次在出口上。通过计算同一数据包的两个实例之间的时间差，可以使用此复制来估计Nexus交换机引入的转发延迟。

在实际中，可以通过比较重复的Echo Request和Echo Reply数据包之间的时间差值，使用之前捕获的ICMP流量来测量此延迟。这为交换机转发性能提供了简单而有效的基准。如果需要更深入的分析，可通过捕获流量并测量重复TCP数据包之间的时间差来对TCP流量应用相同的方法。

方法

- 识别数据包及其重复项（相同序列号）。
- 测量入口和出口副本之间的时间增量。
- 此增量表示交换机转发延迟的上限估计值，因为它可能包括镜像和时间戳开销。

Wireshark配置

- 启用时间增量显示：

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

- 为时间增量添加自定义列：

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

- 过滤相关流量（示例）：

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

- 按序列号或TCP数据流对数据包进行排序：

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

解释

- 重复数据包之间的时间增量可以在微秒范围内。
  - 如果是这种情况，Nexus交换机不会为数据包转发带来延迟。
- 一致的低增量可确认基于硬件的转发性能。
- 增减或不一致的增减可能表明：
  - 拥塞或缓冲

参考

- Cisco Nexus 9000系列产品手册
- Cisco Nexus 9000系列交换机设计指南
- Cisco Nexus 9000系列交换机上的智能缓冲区管理白皮书

源主机数据包捕获的TCP流量分析

本部分提供详细的方法来分析Wireshark中的TCP数据包捕获，包括配置文件配置（通过前面所述的假设情况）。显示的图像直接来自Wireshark。提醒一下，情景是：

一位用户已经发现，对大约6.5 TB的应用程序数据集的备份过程（以前大约在9小时内完成）现在需要将近21个小时。唯一可访问的网络设备是连接到源服务器(10.93.19.8)的Cisco Nexus 9300交换机。 交换机接口上配置的MTU为9000字节（巨帧），而服务器上的MTU未知。可从源服务器捕获数据包，并且所有之前的Nexus验证步骤都已完成，未检测到异常。

主要观察和限制

- 已排除Nexus交换机：
  - 无数据包丢弃
  - 无接口错误
  - 无QoS或ACL影响
  - 硬件转发已确认
- 接口配置:
  - 接入端口
  - MTU:9000 bytes
- 可用数据：
  - 源位置的数据包捕获
  - 端到端MTU知识
    - ping操作成功完成，且未使用包含1,472字节数据的1500字节数据包分段。
- 缺少数据：
  - 目标可视性
  - 目标服务器上没有可用的数据包捕获。

在Wireshark中，您可以创建定制配置文件，以定制要执行的特定类型的分析。

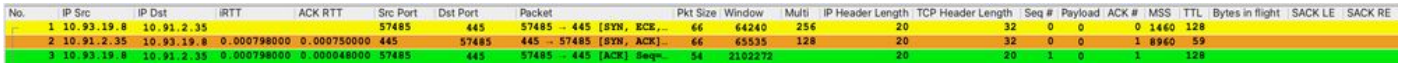
列说明

- tcp.analysis.initial_rtt(iRTT):根据TCP三次握手估计初始往返时间。
- tcp.analysis.ack_rtt(ACK RTT):测量TCP数据段与其对应确认之间的时间。
- tcp.window_size（窗口）：指示应用缩放之前接收方通告的TCP窗口大小。
- tcp.options.wscale.multiplier（多）：表示用于计算有效接收窗口的窗口缩放系数。
- tcp.seq（序列号）：显示TCP数据段中第一个字节的序列号。
- tcp.len（负载）：显示该网段的TCP负载大小（以字节为单位）。
- tcp.ack(ACK#):表示发送方下一个预期字节（累积确认）。
- tcp.options.mss_val(MSS):显示TCP握手期间通告的最大数据段大小。
- ip.ttl(TTL):显示生存时间值，该值对于标识跳数和路由行为非常有用。
- tcp.analysis.bytes_in_flight（传输中的字节）：表示当前传输中的未确认数据量。

TCP三次握手分析

必须捕获TCP三次握手，因为它包含定义会话行为方式的关键参数，如MSS、Window Scale和SACK。

如果没有此信息，任何TCP分析都是不完整的，可能导致有关性能或根本原因的结论不正确。



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			

流量识别

从数据包捕获：

- 源 IP 地址：10.93.19.8
- 目的 IP 地址：10.91.2.35

初始往返时间(iRTT)分析

初始RTT(iRTT)计算如下：

- iRTT = 798微秒

该值源自：

- 数据包2(SYN-ACK)ACK RTT:目标响→SYN的时间为750 µs。
- 数据包3(ACK)ACK RTT:48微秒→源确认SYN-ACK的时间。

大部分延迟（约94%）位于转发路径(客户端→服务器→客户端)，而来自源的响应时间最短，表明客户端上没有CPU或应用延迟。

TCP端口标识

- 目的 TCP 端口:445

端口445对应于Microsoft Server Message Block(SMB)，通常用于文件共享、网络驱动器和Windows身份验证服务。此协议对延迟和吞吐量都很敏感，因此高度依赖TCP效率和网络稳定性。

TCP窗口大小分析

- 源窗口（缩放）：64,240 bytes
- 目标窗口：65,535 bytes

TCP窗口表示在等待确认之前可以发送的数据量。在这种情况下，源比目的地限制稍多。这些值对于现代环境来说相对较小，可能会限制吞吐量，尤其是在RTT增加时。

最大理论吞吐量可通过以下方式估计：

吞吐量= TCP窗口大小/ RTT

替换观察到的值：

- TCP窗口大小= 64,240字节
- RTT = 798微秒= 0.000798秒

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

这表示上限吞吐量，假设：

- 无数据包丢失
- 无重新传输
- 理想的网络条件

吞吐量、传输时间和所需条件分析

在当前吞吐量为644 Mbps的情况下，传输6.5 TB的文件大约需要23.5小时，这与观察到的性能下降情况相符。要实现9小时的传输时段，吞吐量必须提高至约1.68 Gbps，需要更大的TCP时段（大约增加2.7倍）或显着更低的RTT（约291微秒）。

在当前条件下（64 KB窗口和~798 µs RTT），不可能达到9小时目标，因为TCP吞吐量受带宽延迟产品的限制。如果不增加窗口大小或减少延迟，协议将无法利用更高的可用带宽，从而使目标无法实现。

场景	吞吐量	预计传输时间(6.5 TB)	必需的TCP窗口	必需的RTT
当前状态	644 Mbps（约80.5 MB/s）	约23.5小时	64 KB	798微秒
目标（9小时）	~1683 Mbps（约210 MB/s）	9 小时	~172 KB	~291微秒

这在过去有效，表示网络、应用程序、源或目标发生更改。必须指出的是，仅基于这一初步分析，就可以得出一个重要的结论：在当前TCP窗口大小和RTT条件下，无法实现9小时目标。

下表比较了吞吐量随RTT和TCP窗口大小增加或减小而变化的情况。

RTT对吞吐量的影响（固定窗口大小= 64,240字节）

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

TCP窗口大小影响（固定RTT = 798 µs）

TCP窗口大小	吞吐量(MB/s)	吞吐量(Mbps)
16 KB(16,384 B)	约20.5 MB/s	约164 Mbps
64 KB(64,240 B)	约80.5 MB/s	约644 Mbps
256 KB(262,144 B)	~328 MB/s	约2,624 Mbps
1兆字节（1,048,576字节）	~1,314 MB/s	约10.5 Gbps

技术解释

- 吞吐量与RTT成反比，→迟会降低性能。
- 吞吐量与TCP窗口大小成正比，→大窗口增加容量。
- 小窗口大小严重限制了吞吐量，即使在低延迟环境中也是如此。
- 高速网络(10G+)需要窗口扩展以充分利用带宽。

这表明RTT和TCP窗口大小都是TCP性能的关键因素，在排除吞吐量问题时必须一起进行分析。

IP和TCP报头长度

- IP报头长度：20 bytes
- TCP报头长度：32 bytes

20字节的IP报头表示不存在IP选项。32字节的TCP报头确认正在使用TCP选项，在基本报头外增加12个字节。这些选项通常包括MSS、Window Scale和SACK Permitted。

TCP选项分析和TTL

两个终端上均启用了选择性确认(SACK)。这在图片中不可见。SACK允许接收方确认非连续的数据块，从而向发送方准确通知哪些数据段已成功接收。

例如，如果收到数据段1000-2000和3000-4000，但缺少2000-3000，则接收方可以明确指出这一点。如果没有SACK，发送方会在间隙后重新传输所有数据；使用SACK时，仅重新传输丢失的部分。这显着提高了在丢包环境中的性能。

数据包1(SYN)分析

- 序列号:0（Wireshark标准化）
- 负载：0 bytes
- 确认编号:0
- MSS:1460 bytes
- TTL:128

Wireshark将序列号规范化为零，以实现可读性，但实际上它是一个很大的随机值。连接建立期间预期没有负载。MSS值1460字节表示1500字节的MTU（20字节IP报头+ 20字节TCP报头）。TTL 128可以是基于Windows的主机，在捕获中看到此值表明捕获可能是在源位置或非常靠近源位置通过第2层进行的。

数据包2(SYN-ACK)分析

- 确认编号:1

ACK值为1，因为SYN标志会使用一个序列号，即使没有负载时也是如此。因此，ACK = SEQ + 1。

- TTL:59

观测到的TTL为59，表明初始TTL为64，这意味着数据包经过大约5个路由跳(64 − 59 = 5)。 每个路由的跃点将TTL递减1。

分散风险和网络影响

大约五个路由跳的存在会引入潜在的性能风险，尤其是与MTU不匹配和分段相关的风险。

如果任何中间链路的MTU低于原始数据包大小，则会发生分段。这会导致以下几种后果：

- 由于分段和重组开销，延迟增加。
- 数据包丢失的概率更高，因为丢失一个分段需要重新传输整个数据包。
- 吞吐量降低，因为TCP会将丢失视为拥塞并降低其发送速率。
- 处理分片的网络设备上的CPU使用率提高。
- 如果ICMP被阻止，则路径MTU发现(PMTUD)失败的风险导致静默数据包丢弃。

考虑到这些因素，确保路径中的MTU一致或在必要时实施MSS夹紧至关重要。

TCP RTT分析：ACK RTT与初始RTT

当ACK RTT大于iRTT时，表明与TCP握手期间建立的基线相比，延迟已增加。

这意味着网络或终端在会话期间引入额外的延迟，通常是由于：

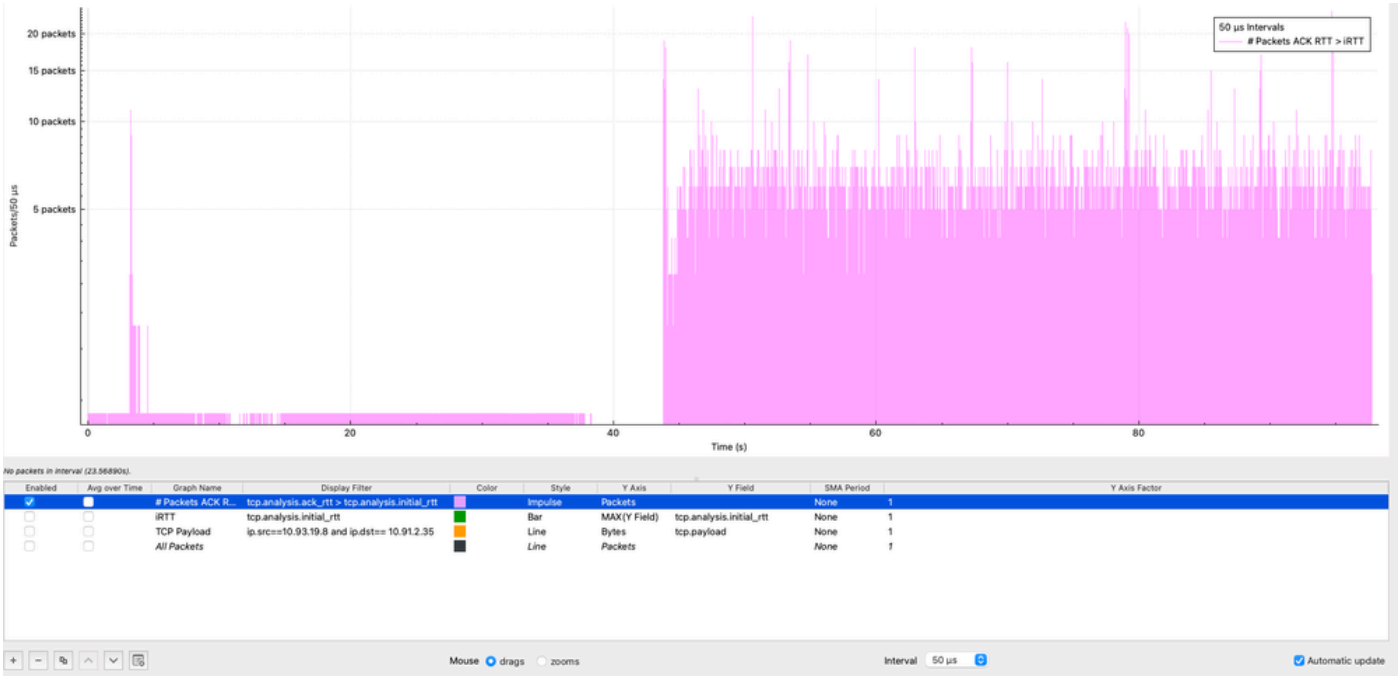
- 网络拥塞或排队
- 接收器或应用处理延迟
- 中间设备（防火墙、负载均衡器）
- 重新传输

如果此情况在整个TCP会话期间持续出现，则会导致：

- 降低TCP吞吐量
- 低效的窗口利用率
- 应用性能降低

在Wireshark中，可以通过使用I/O Graphs（I/O图形）功能来直观显示ACK RTT > iRTT条件的发生频率：统计→ I/O Graphs，应用显示过滤器(tcp.analysis.ack_rtt > tcp.analysis.initial_rtt)，选择Impulse style，将Y轴设置为Packets，并使用50微秒的时间间隔。

在图中，紫色脉冲表示每个50微秒间隔内满足此条件的数据包数量。如前所述，此情况在整个数据包捕获过程中持续出现，表明会话期间的延迟始终高于初始基线。此行为强烈表明性能持续下降，而不是瞬变状态，这进一步表明有必要调查端到端路径中的潜在来源，如拥塞、缓冲或终端处理延迟。

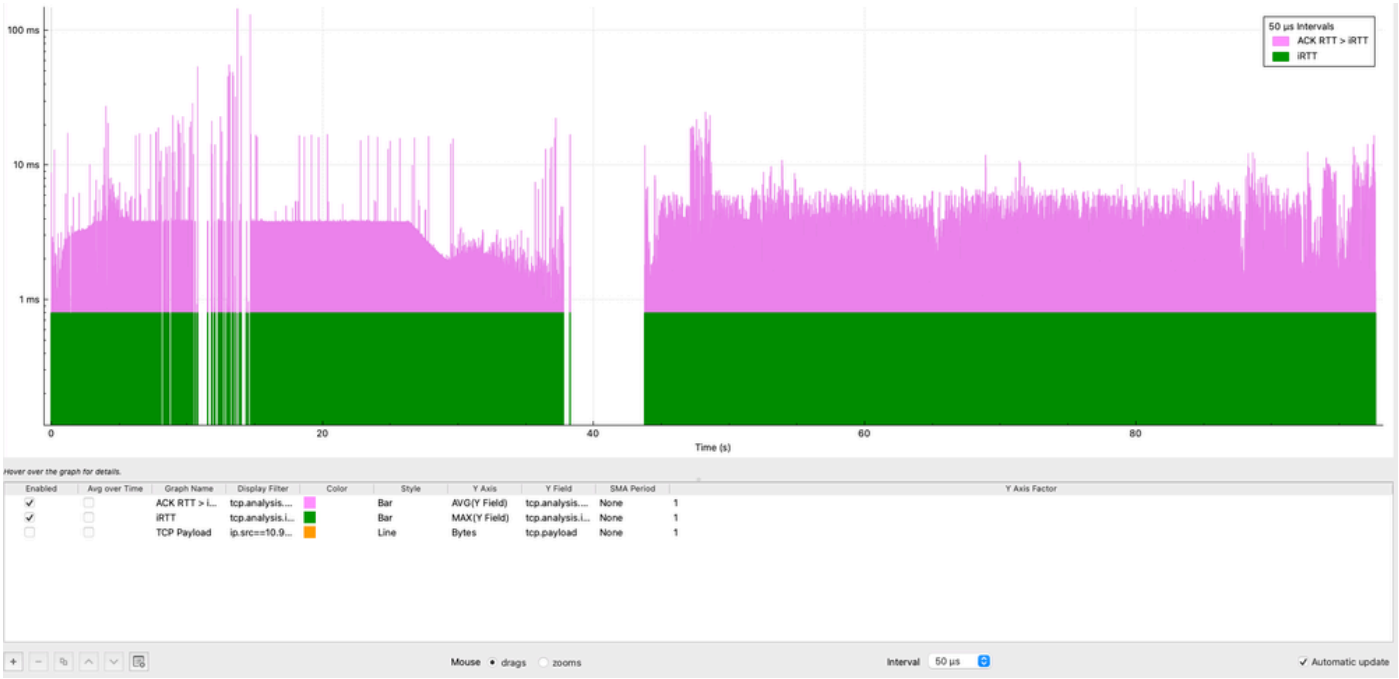


确定iRTT被超出的时间也很重要，而不仅仅是频率。虽然Wireshark不允许直接在字段之间进行减法，但可以使用I/O图形进行直观比较：

- 导航到Statistics → I/O图形
- 图1:
  - 显示过滤器：tcp.analysis.ack_rtt > tcp.analysis.initial_rtt
  - 样式：栏
  - Y轴：平均值
  - Y字段：tcp.analysis.ack_rtt
  - 间隔:50微秒
- 图2:
  - 显示过滤器：tcp.analysis.initial_rtt
  - 样式：栏
  - Y轴：最大
  - Y字段：tcp.analysis.initial_rtt
- 然后右键单击图形并启用Log scale。

在此可视化中，紫色图形表示条件ACK RTT > iRTT，该条件在整个TCP会话中始终存在。数据显示持续的延迟膨胀，多个峰值达到11毫秒，最大峰值超过100毫秒，表示基线iRTT的11倍到100倍。

此行为确认延迟增加不是暂时的，而是持久的，表明系统问题会随着时间影响会话。这种持续偏差强烈表明存在网络拥塞、缓冲(bufferbloat)或终端处理延迟等因素。



TCP重传和虚假重传分析

本部分通过分析一段时间的重新传输来评估TCP可靠性，从而验证数据包丢失是否会导致性能下降。

一段时间内的TCP重新传输

图中显示了TCP重新传输随时间变化的分布。总共观察到42次重传，仅占总流量的0.00125%。

这种重传级别可以忽略不计，并且清楚地表明数据包丢失不是造成这种情况的原因。

Wireshark配置（TCP重新传输）

`Statistics → I/O Graphs`

- 显示过滤器：

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

- 样式：冲动或条形图

- Y轴：数据包
- 间隔:1 秒

TCP虚假重传

该图显示源10.93.19.8在1秒间隔内生成的TCP虚假重传次数。

在Wireshark中，TCP虚假重传表示主机重传了实际上并未丢失的数据段。原始数据包成功到达接收方，但发送方由于不准确的计时估计而错误地假设丢失。此行为并不表示真正的数据包丢失，而是表明发送方重新传输逻辑效率低下。

在本捕获中：

- 源10.93.19.8在大约8微秒后重新传输数据包。
- 而典型的重新传输计时器大约是200毫秒。

这确认重新传输行为完全由源TCP堆栈控制，而不是由网络控制。

观察到的虚假重传总数为1,112，占捕获总流量的0.0332%。

Wireshark配置（TCP虚假重传）

```
Statistics → I/O Graphs
```

- 显示过滤器：

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

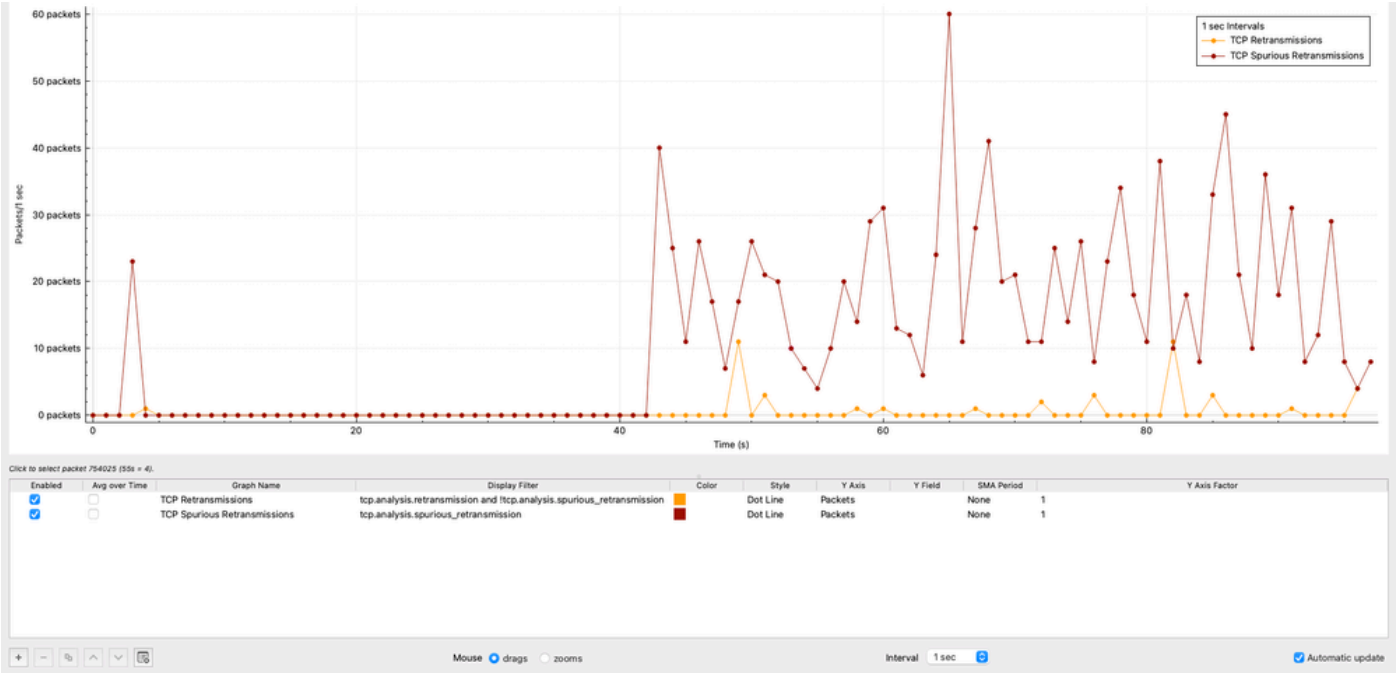
- 样式：冲动或条形图
- Y轴：数据包
- 间隔:1 秒

技术解释

- 实际重新传输的百分比极低，表明网络中不存在丢包现象。
- 虚假重传的存在表示源主机过早地作出重传决策。

- 此行为可能会对效率产生轻微影响，但并不是导致吞吐量严重下降的主要原因。

此分析进一步强调，此问题与网络可靠性无关，而与TCP行为、延迟或终端性能有关。

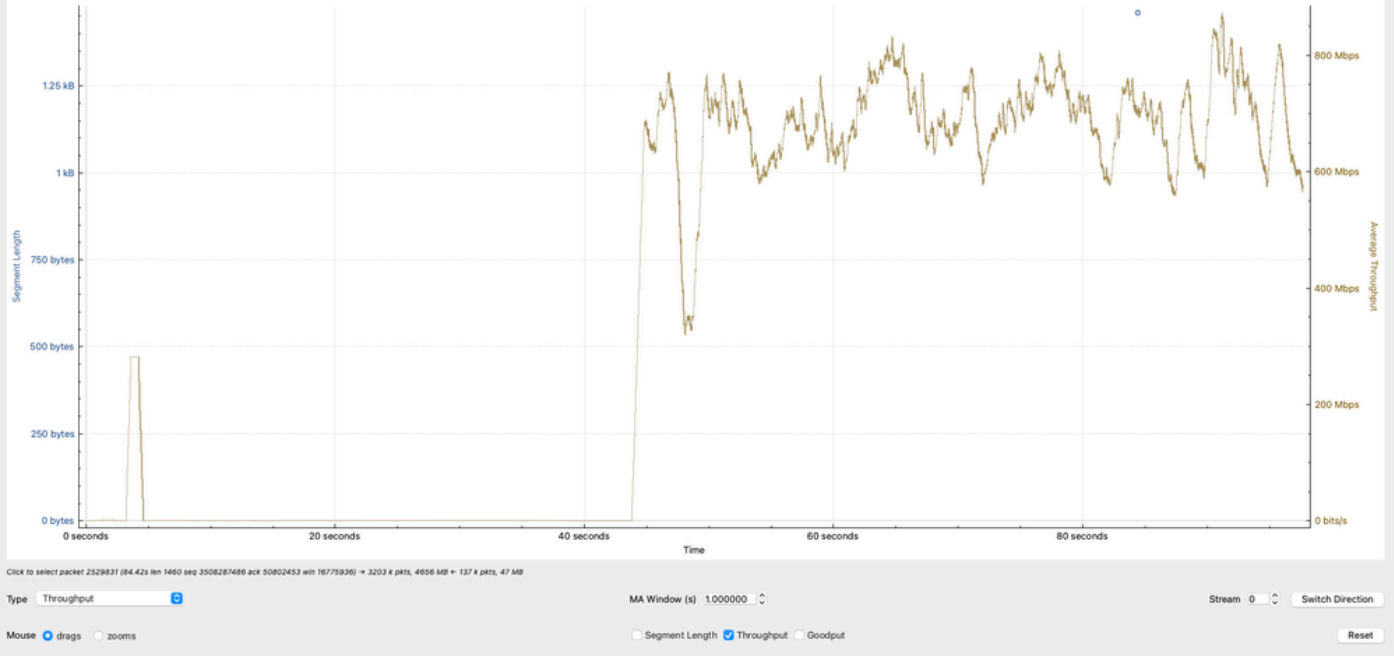


有效的吞吐量分析

该图显示了有效吞吐量，根据TCP负载（实际传输的数据）计算得出，单位为每秒Mb。观察到的吞吐量主要在600 Mbps和800 Mbps之间波动，这表明网络在主动传输数据的同时，并没有达到更高的带宽潜力。

Wireshark配置（有效吞吐量）

`Statistics → TCP Streams Graphs → Throughout`



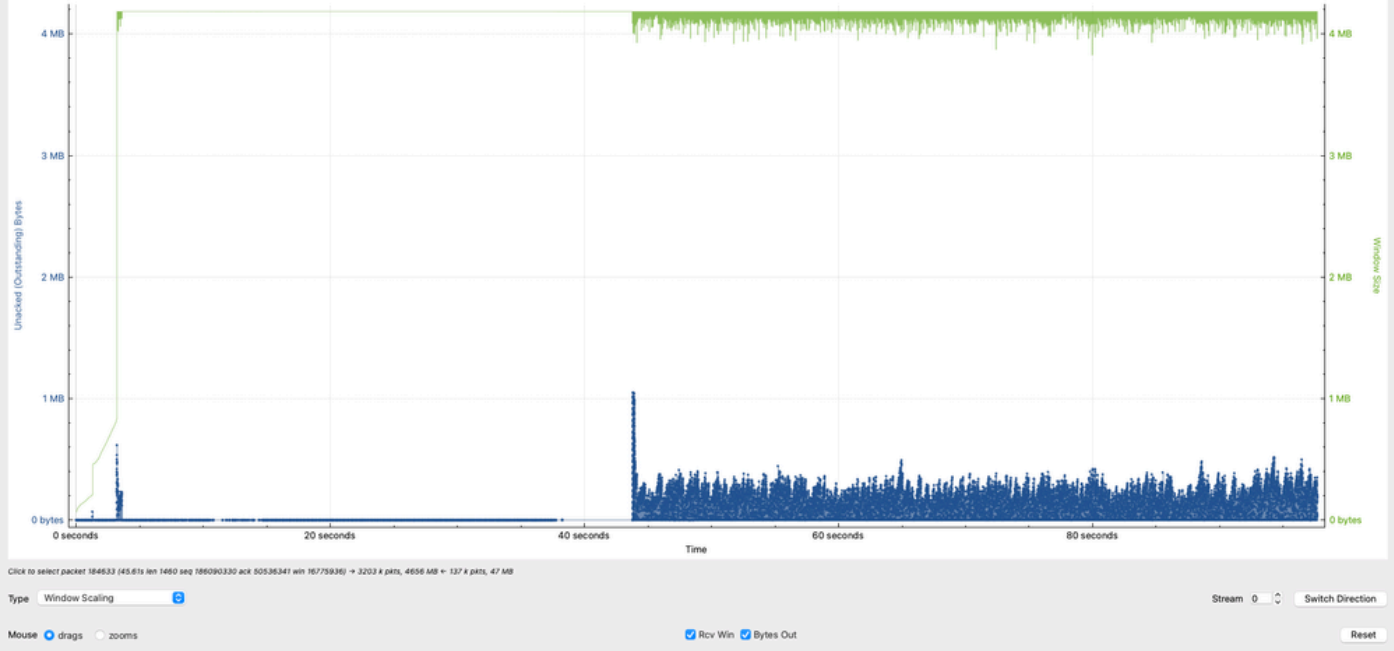
技术解释

- 600-800 Mbps的吞吐量范围与基于TCP窗口大小和RTT的先前计算一致。
- 吞吐量变化反映了：
  - RTT波动
  - TCP拥塞控制调整
  - 应用程序调步或缓冲
- 由于吞吐量不接近线速（例如10G），因此限制不是物理带宽，而是TCP效率限制。
- 此分析确认观察到的吞吐量与TCP限制（窗口大小和延迟）一致，进一步表明瓶颈不是由于数据包丢失或接口容量所致，而是由于传输层行为和终端条件所致。

飞行数据（TCP窗口）分析

该图通过比较接收方容量与传输中的实际数据（传输中的字节数），突出显示TCP会话中的一个关键行为。

- 绿线表示10.91.2.35（接收方）可以接受的TCP数据量（有效接收窗口）。
- 蓝线表示当前从10.93.19.8（发送方）传输的TCP数据量。



观测到的Data in Flight峰值约为1 MB，在8 KB和5 KB附近有额外的峰值，但主要集中在1 KB和250 KB之间。

这表明，虽然接收方能够处理更大量的数据，但发送方并未始终如一地利用可用窗口。

Wireshark配置（飞行中的数据与窗口）

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

技术解释

- 接收机(10.91.2.35)通告一个明显更大的窗口，表明它能够接收更多的数据。
- 发送方(10.93.19.8)未充分利用可用窗口，如较低和不一致的Data in Flight值所示。
  - 发送方可以理想地将Data in Flight值保持在更靠近接收方通告窗口(~1 MB)的位置，以最大限度地提高吞吐量。
  - 无法维持高飞行数据水平直接限制了吞吐量，它是在源位置出现TCP效率低下的有力指标，而不是网络容量问题。

TCP负载与MSS随时间变化的分析

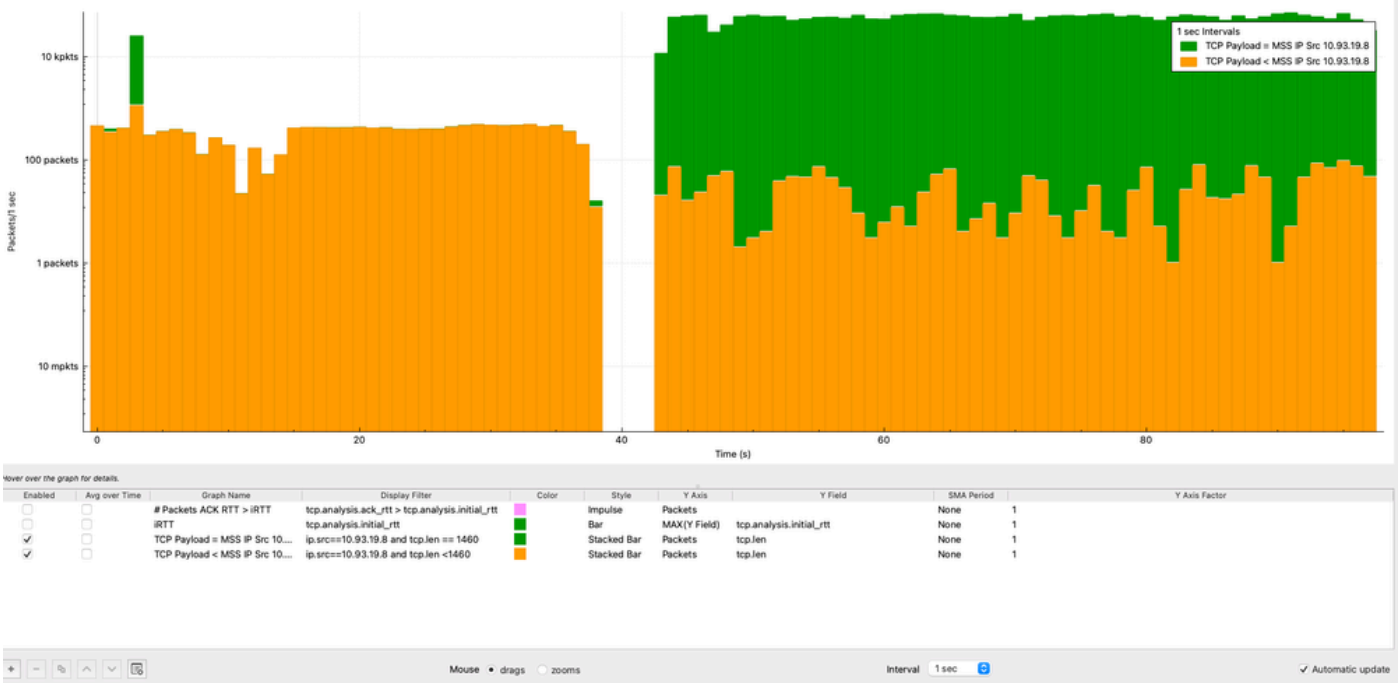
根据MSS分析随时间变化的TCP负载大小有助于确定发送方是否高效地利用每个TCP数据段。从源IP地址(10.93.19.8)的角度执行此分析。

在Wireshark中，图形配置如下：

- 图1（MSS大小的数据包）：
  - 显示过滤器：ip.src==10.93.19.8和tcp.len == 1460
  - 样式：堆积条形图
  - Y轴：数据包
  - 间隔:1 秒
- 图2(所有数据包≤MSS):
  - 显示过滤器：ip.src==10.93.19.8和tcp.len <= 1460
  - 样式：堆积条形图
  - Y轴：数据包
  - 间隔:1 秒
- 应用对数刻度以获得更佳的可视性

根据分析：

- 大多数数据包（>10,000个数据包/秒）始终达到1460字节的MSS值。
- 由于正常的TCP行为（ACK、分段或流尾数据），较小部分的数据包负载较少。



根本原因分析(RCA):TCP性能下降

此分析表明，要确定TCP性能问题的根本原因，需要采用全面的端到端方法，而不是假设网络是性能退化的主要根源。

在Cisco Nexus 9300交换机上执行了广泛的验证，包括接口计数器、QoS策略、路由和ARP稳定性、CPU突发验证、基于SPAN的数据包捕获以及使用ELAM的ASIC级转发验证。所有结果一致证实交换机在预期参数下运行：

- 无数据包丢弃
- 无异常延迟（微秒范围）
- 无QoS或控制平面影响
- 正确的硬件转发

此外，TCP分析还显示：

- 可忽略的重新传输(0.00125%)
- 没有数据包丢失的证据
- 源位置的MSS利用率一致
- 与TCP窗口和RTT限制一致的吞吐量
- 可用TCP窗口利用不足（飞行数据分析）
- 网络不是瓶颈
- 源服务器正在限制性能

结论

性能下降是由于源服务器在支持Jumbo的环境中与MTU 1500一起操作导致的，这阻碍了有效利用可用网络容量。

解决方案

将源服务器上的MTU从1500字节增加到9000字节，以便与目标和网络基础设施保持一致。优势：

- 启用更大的TCP数据段
- 降低数据包开销
- 提高整体吞吐量

技术思考

此分析的主要内容是在排除网络性能故障时避免过早总结的重要性。虽然最初将问题归结到网络是很常见的，但此案例清楚地表明网络在整个数据平面路径中运行正常。只有从源和目标两个角度执行深入的TCP分析（包括握手参数、RTT行为、窗口利用率、重新传输和负载效率），才能准确识别真正的瓶颈。

花时间详细分析TCP行为，可防止误诊，减少不必要的网络更改，并确保补救工作针对实际根本原因。

关于此翻译
思科采用人工翻译与机器翻译相结合的方式将此文档翻译成不同语言，希望全球的用户都能通过各自的语言得到支持性的内容。

请注意：即使是最好的机器翻译，其准确度也不及专业翻译人员的水平。

Cisco Systems, Inc. 对于翻译的准确性不承担任何责任，并建议您总是参考英文原始文档（已提供链接）。