TCPパフォーマンス低下のトラブルシューティング

内容

はじめに

このドキュメントでは、TCPの基礎、Wiresharkの詳細なパケット分析、およびエンドツーエンドのパフォーマンスを最適化するための実用的なトラブルシューティングについて説明します。

前提条件

要 件

次の項目に関する知識があることが推奨されます。

- IP/TCP

使用するコンポーネント

このドキュメントの情報は、次のソフトウェアとハードウェアのバージョンに基づいています。

- Cisco NX-OS 10.6(X)を使用したCisco Nexus 9000クラウドの拡張

注：サードパーティ製のソフトウェアまたはハードウェアの設定と相互運用性に関する質問は、シスコのサポート対象外です。サードパーティ製のツールを使用すると、シスコ機器の設定と動作をデモンストレーションできます。

このドキュメントの情報は、特定のラボ環境にあるデバイスに基づいて作成されたものです。このドキュメントで使用するすべてのデバイスは、クリアな（デフォルト）設定で作業を開始しています。本稼働中のネットワークでは、各コマンドによって起こる可能性がある影響を十分確認してください。

バックグラウンド情報

TCPとは

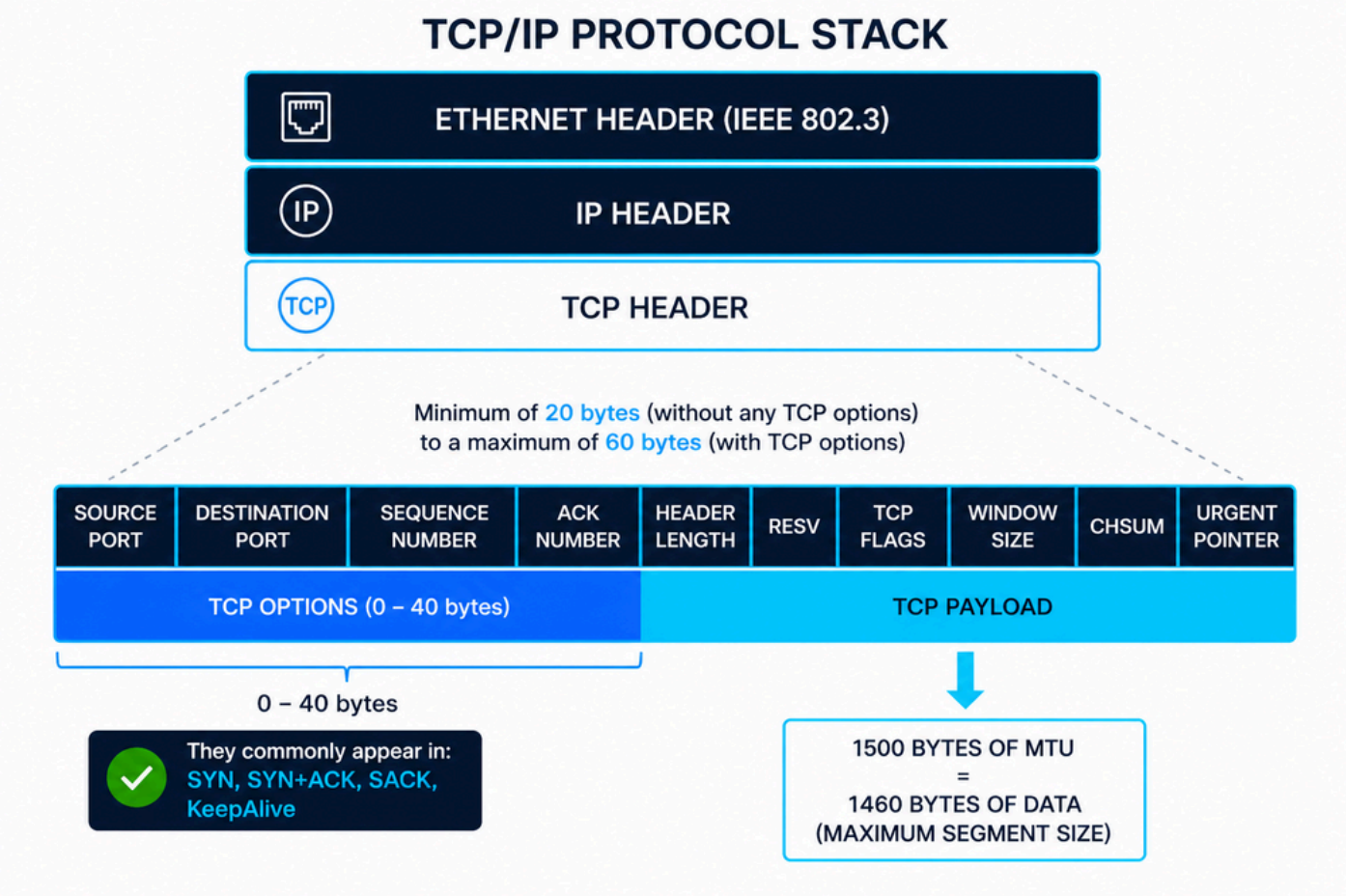
Transmission Control Protocol（TCP；伝送制御プロトコル）は、OSIモデルのレイヤ4で動作する基本的なトランスポート層プロトコルであり、IPネットワーク上で通信するアプリケーション間で、信頼性が高く、順序付けされ、エラーチェックが行われたバイトストリームの配信を提供します。

3つの主な利点

1. 信頼性：TCPはコネクション型であり、受信側からの確認応答を要求することによって配信を保証します。送信中にパケットが失われたり破損したりすると、TCPはデータを自動的に再送信して、宛先に確実に到達するようにします。
2. 順序どおりの配信：ネットワークパケットは順序どおりに到着できないため、TCPは各セグメントにシーケンス番号を割り当てます。これにより、受信側システムでは、最初に送信された順序と同じ順序でデータを再構成できます。
3. フローおよび輻輳制御：TCPは、受信側の処理能力と現在のネットワーク状態に合わせてデータ伝送速度を動的に管理し、バッファオーバーフローやネットワーク輻輳によるデータ損失を防止します。

TCP/IPカプセル化の概要

この図は、TCPセグメント（レイヤ4）がIPパケット（レイヤ3）内にカプセル化され、次にIEEE 802.3によって定義されたイーサネットフレーム（レイヤ2）内にカプセル化されるTCP/IPスタックを表しています。この階層型アプローチでは、モジュラ通信が保証され、各層が独自の制御情報（ヘッダー）を追加して、配信、ルーティング、およびデータ整合性を保証します。



イーサネットヘッダー(IEEE 802.3)

イーサネットヘッダーは通常14バイトで、次のもので構成されます。

- 宛先MACアドレス（6バイト）
- 送信元MACアドレス（6バイト）
- EtherType/Length（2バイト）

また、イーサネットフレームには、レイヤ2でのエラー検出用に4バイトのフレームチェックシーケンス(FCS)トレーラが含まれています。IEEE 802.3では、フレーミング、最小/最大フレームサイズ、およびTCPなどの上位層プロトコルに直接影響する物理的な配信制約が定義されています。

IPヘッダー(IPv4)

IPv4ヘッダーの最小サイズは20バイトで、オプションで最大60バイトまで拡張可能です。主なフィールドは次のとおりです。

- 発信元および宛先 IP アドレス
- 存続可能時間(TTL)
- プロトコル（TCPをペイロードとして識別）

IP層は、ネットワーク全体の論理アドレッシングとルーティングを担当しますが、信頼性は保証しません。

TCPヘッダー構造

TCPヘッダーの範囲は20 ~ 60バイトで、オプションによって異なります。主なフィールドは次のとおりです。

- 送信元/宛先ポート
- シーケンス番号
- 確認応答番号
- フラグ（SYN、ACK、FIN、RSTなど）
- ウィンドウサイズ
- Checksum

TCPは、信頼性の高い配信、適切なシーケンシング、およびフロー制御をIP通信に追加します。

TCPオプション（共通10）

TCPオプションは基本プロトコルを拡張します。最も一般的なものは次のとおりです。

1. Maximum Segment Size（MSS；最大セグメントサイズ）：ホストが受け入れ可能な最大のTCPペイロードを定義します。
2. ウィンドウスケール：受信ウィンドウを65,535バイトを超えて拡張します。
3. Selective Acknowledgment Permitted(SACK Permitted)：選択的な確認応答機能をイネーブルにします。
4. Selective Acknowledgment(SACK)：完全な再送信を回避するために、受信データブロックを指定します。
5. タイムスタンプ：RTTの計算とWrapped Sequence Number（PAWS；折り返しシーケンス番号）からの保護に使用されます。
6. No-Operation(NOP)：オプションの配置のためのパディング。
7. End of Option List(EOL):TCPオプションの終了を示します。
8. TCP Fast Open(TFO)：最初のハンドシェイク時にデータ交換を許可します。
9. マルチパスTCP(MPTCP)：単一のTCPセッションで複数のネットワークパスを有効にします。
10. User Timeout Option (UTO) – 送信データが未確認のまま残る期間を制御します。

TCPシーケンスおよび確認応答動作（SYN/FINを含む）

SYNフラグとFINフラグはどちらも、ペイロードが存在しない場合でも、1つのシーケンス番号を消費します。TCPはバイト指向のシーケンシングモデルを使用して動作します。このモデルでは、送信されるすべてのバイト（および特定の制御フラグ）がシーケンス空間を進めます。この動作は、パケットキャプチャでの正確なTCP分析や、シーケンシングまたは確認応答の不整合の診断に不可欠です。

ACK = SEQ + Payload Length + (SYN ?1 : 0) + (FIN ?1 : 0)

ここで、

- SEQ =初期順序番号
- ペイロード長=バイト単位のデータサイズ
- SYN?1: 0 = SYNフラグが設定されている場合は1を追加し、それ以外の場合は0を追加
- フィン？1: 0 = FINフラグが設定されている場合は1を追加し、それ以外の場合は0を追加
- ACK =次の予測バイト

例1：データを含むSYN(TCP Fast Open)

- シーケンス= 1000
- SYN = 1
- ペイロード長= 200バイト

ACK計算：

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

これは、TCPハンドシェイク中にデータが送信されるシナリオを反映しています。ペイロードとSYNフラグの両方がシーケンス空間を消費します。

例2：データを含むFIN（接続の終了）

- シーケンス= 3000
- フィン= 1
- ペイロード長= 150バイト

ACK計算：

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

これは、接続のティアダウン時にTCPにデータを含めることができ、ペイロードとFINフラグの両方でシーケンス番号が増分されることを示しています。

MSSおよびMTUとの関係

最大セグメントサイズ(MSS)は、TCPがセグメントで送信できる最大ペイロードを定義します。

- 通常のイーサネットMTU = 1500バイト
- MSS = MTU − IPヘッダー− TCPヘッダー
- 標準MSS = 1460バイト(1500 − 20 − 20)

TCPオプションが存在する場合は、それに応じてMSSが減少します。MSSはTCP 3ウェイハンドシェイク中にネゴシエートされ、IPレイヤでのフラグメンテーションを防止します。

TCP 3ウェイハンドシェイクにおけるMSSネゴシエーションの動作

最大セグメントサイズ(MSS)は、SYNパケットのMSSオプションを使用して、TCP 3ウェイハンドシェイク中に交換されます。

- ホストA→ホストB(SYN):MSSをアドバタイズします（1460など）。
- ホストB→ホストA(SYN-ACK):MSSをアドバタイズします（1380など）。

それぞれの側が効果的に次のことを言っています。

これは受け入れられる最大のTCPペイロードです。

キールール：MSSは方向性

MSSは、単一の合意値としてネゴシエートされません。

その代わり：

- 各ホストは、相手側からアドバタイズされたMSSを使用します。
- これにより、方向ごとに1つずつ、2つの独立した制限が作成されます。

したがって

- AはBのMSSを使用してデータを送信します。
- BはAのMSSを使用してデータを送信します。

送信元は宛先MSSよりも多くのTCPペイロードを送信できるか

正常に機能しているTCPスタック：いいえ。

- 送信側は、受信側によってアドバタイズされたMSSを尊重する必要があります。
- より大きなセグメントを送信すると、次のようなリスクがあります。
  - IPフラグメンテーション（MTUを超えた場合）
  - パケットドロップ（フラグメンテーションがブロックまたはサポートされていない場合）
- これにより、次のことが実現されます。
  - 再転送
  - パフォーマンスの低下
  - PMTUD(Path MTU Discovery)ブラックホールなどの問題

トラブルシューティングのための実用的な洞察

- TCP 3ウェイハンドシェイク（SYN/SYN-ACKパケット）のMSS値を常に確認します。
- 次の原因による不一致を確認します。
  - トンネル(VXLAN、GRE、IPsec)
  - MSSを変更するファイアウォール（MSSクランプ）
- Cisco NX-OSなどのプラットフォームでは、カプセル化されたパス間のフラグメンテーションを防ぐために、MSS調整がよく使用されます

ウィンドウサイズ（フロー制御）

ウィンドウサイズは、受信側が確認応答なしで受け入れるデータの量を定義します。

内容：

- バッファオーバーフローを防止するフロー制御メカニズム。

目的:

- 送信者が受信者を圧迫しないようにします。

入手先：

- パケットキャプチャ（Wiresharkなど）に表示されます。
- OSのTCPスタック構成とバッファサイズから取得。

ベンダー/OSの多様性：

- 実装が異なる(Linux、Windows、Cisco NX-OS)場合は、動的なスケーリングとバッファチューニングを使用するため、ウィンドウサイズが異なります。

ゼロウィンドウ状態：

- Window Sizeが0の場合、受信側バッファがいっぱいになります。
- 送信側は送信を一時停止し、定期的なプローブを送信します。

可変Windowsメカニズム

- レートベースフロー制御
  - 送信側に固定データレートを割り当て、その割り当てを超えるデータが送信されないようにします。
  - ストリーミングアプリケーションに最適です。
  - ブロードキャストおよびマルチキャスト配信
- ウィンドウベースのフロー制御
  - ウィンドウサイズは時間とともに変化します。
  - 受信側は、送信側ウィンドウの更新に許可ウィンドウをシグナリングすることで、フロー制御を実現します。

トラブルシューティングの使用：

- 受信側のボトルネック（CPU、メモリ、アプリケーション）→小さいまたはゼロのウィンドウ。
- ウィンドウは大きいがスループット→低い：ネットワークの問題（遅延、輻輳）。
- ウィンドウの動作の分析は、TCPセッションのパフォーマンスの問題を診断するために重要です。

Cisco Nexus 9000(NX-OS)でのTCPデータプレーンのトラブルシューティング

このセクションでは、NX-OSを実行しているCisco NexusスイッチがTCPトラフィックの転送に

影響を与えているかどうかを診断する、またはパフォーマンスの問題を引き起こす実用的な方法論について説明します。このアプローチは、仮説シナリオを通じて提示されます。

TCPの遅延やパフォーマンスの低下が見られる場合、最初にネットワークが原因であると疑われるのが一般的です。ただし、この前提はデータ駆動型分析を通じて検証する必要があります。 TCPトラブルシューティングの正式な方法はパケットキャプチャで、理想的には次のように実行されます。

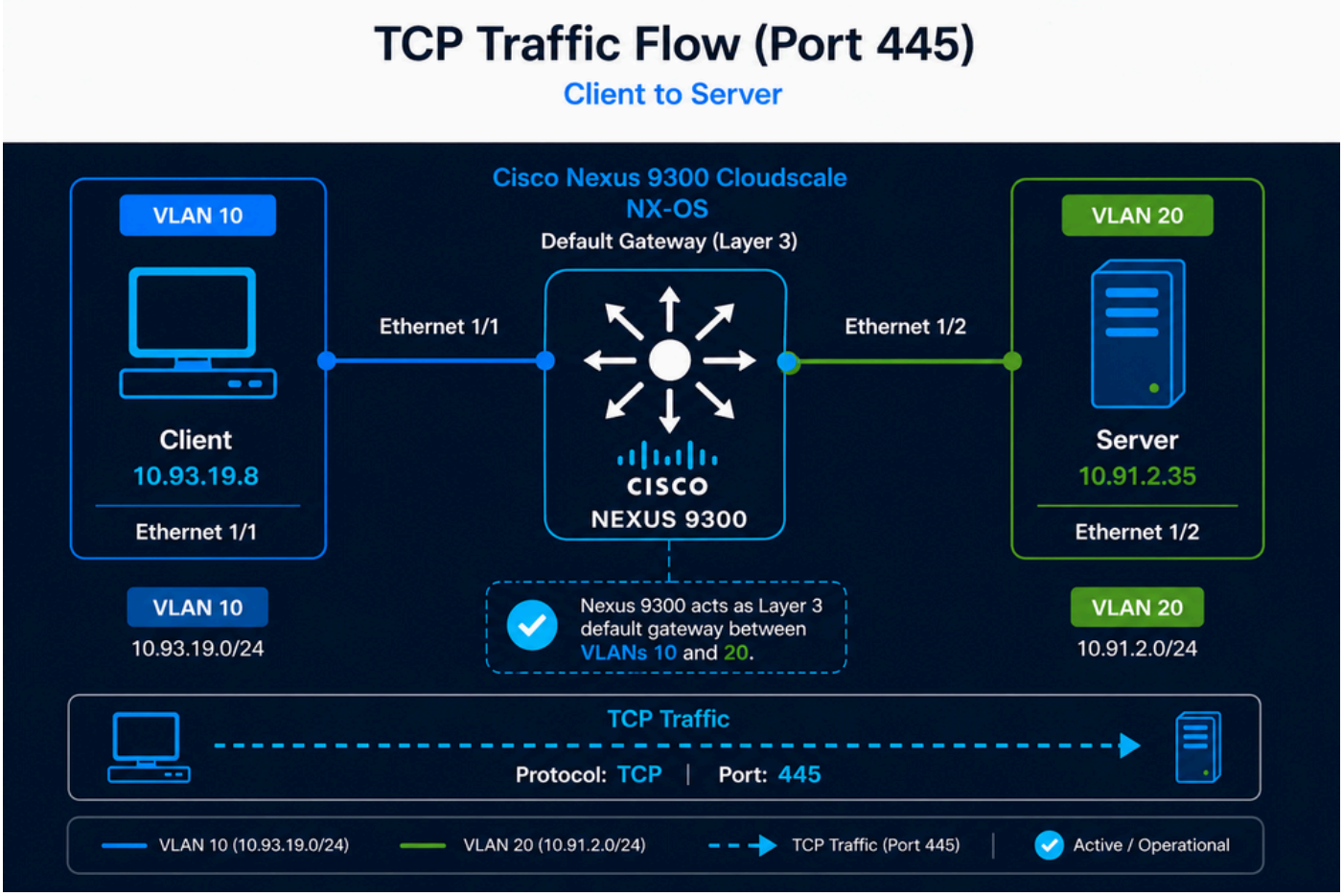
- 送信元と宛先で同時に
- トラフィック開始前

これにより、MSS、ウィンドウスケール、SACKなどの重要なパラメータがネゴシエートされ、セッションの後半で繰り返されないTCP 3ウェイハンドシェイクに対する可視性が確保されます。 同時キャプチャが不可能な場合は、1回のキャプチャで分析を続行できますが、結論は制限されます。

シナリオの定義

約9時間で完了していた約7.5 TBのアプリケーション・データセットのバックアップ・プロセスが、約21時間かかっていることをユーザーが特定しました。クライアントとサーバの間のTCPセッションは引き続き正常に確立されますが、バックアップ期間が大幅に長くなることは、スループットまたはTCP全体のパフォーマンスが低下する可能性を示唆しています。Nexusスイッチはパス内の唯一のネットワークデバイスであり、レイヤ3ゲートウェイ機能も提供するため、ネットワーク管理者はNexusスイッチが問題の原因ではないかと疑います。

- クライアント：10.93.19.8(VLAN 10)
- サーバ：10.91.2.35(VLAN 20)
- デフォルトゲートウェイとして機能するNexus 9300
- TCP ポート 445



初期検証（到達可能性）

- これらのコマンドは、Don't Fragment（DF；フラグメントなし）ビットが設定されたICMPパケットを送信することによって、発信元と宛先間のパスMTU(PMTU)を検証するために使用されます。これは、フラグメンテーションなしでネットワークを通過できる最大パケットサイズを決定するのに役立ちます。 このプロセスは、送信元と宛先の両方で実行する必要があります。
- 送信元と宛先の両方で、物理インターフェイスのMTUを必ず確認してください。
- このシナリオでは、1500のMTUが特定された送信元ホストだけがアクセスできます。

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

- -c 10 → 10のICMPエコー要求を送信
- -I 192.168.10.10 →この特定の送信元IP/インターフェイスを使用
- -s 1472 → ICMPペイロードサイズを1472バイトに設定
- -M do → DF (Don't Fragment)ビットを設定する
- 192.168.20.20 →宛先IP

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

- -n 10 → 10個のICMPエコー要求を送信
- -l 1472 → ICMPペイロードサイズを1472バイトに設定
- -f → Don't Fragment (DF)フラグを設定する
- 192.168.20.20 →宛先IP

1472バイトが必要な理由

- ICMPペイロード= 1472バイト
- IPヘッダー= 20バイト
- ICMPヘッダー= 8バイト
- 合計パケットサイズ：1472 + 20 + 8 = 1500バイト（標準MTU）
- これにより、パスがフラグメンテーションのない1500バイトMTUをサポートするかどうかをテストします。1500バイトのICMPペイロードを送信しようとすると、IPヘッダーとICMPヘッダーを追加した後に合計パケットサイズが標準MTUを超えるため、pingが失敗する可能性があります。

結論付けられること

- pingが成功した場合（パケット損失がない場合）、パスは少なくとも1500バイトのMTUをサポートし、フラグメンテーションは必要ありません。
  - ICMPの結果→消去し、TCP分析に進みます。
  - 断続的なpingの成功→、考えられるパケット損失、一時的な輻輳、レート制限、または転送の問題です。TCPは効率的に動作するために損失のないパスを必要とするため、パケット損失の分析に進んでください。
- pingがエラー「Fragmentation needed」で失敗するか、タイムアウトした場合、パス内に1500バイトよりも小さいMTUのリンクが存在します。このパケットはDFビットが原因で転送できず、これはパスMTUの問題を示しています。

トラブルシューティングのためにこの設定を使用する方法

- 成功する最大サイズを特定するために、ペイロードサイズを徐々に小さくします(たとえば、1472 → 1400 → 1300)。
- 特定されたら、MTU =ペイロード+ 28バイト（IP + ICMPヘッダー）という式を使用してMTUを計算します。

TCPとの実際的な関連性

- MTUが予想よりも小さい場合、TCPセグメントはフラグメント化されるか、廃棄されます

。
- これにより、再送信、遅延の増加、スループットの低下が発生し、アプリケーションのパフォーマンスに直接影響が及びます。

トラフィックパスの特定（インターフェイス）

Cisco Nexus 9000スイッチのTCPパフォーマンスを効果的にトラブルシューティングするには、送信元と宛先の間でトラフィックを送受信しているインターフェイスを判別することが不可欠です。

単純なトポロジでは、これは物理接続から直接推測できます。たとえば、クライアントがEthernet1/1に接続され、サーバがEthernet1/2に接続されている場合、トラフィックパスは単純です。ただし、複数のアクティブインターフェイス、ポートチャネル、またはvPC設定を使用する実際の環境では、この識別は必ずしも単純ではありません。

このような場合は、Embedded Logic Analyzer Module(ELAM)を使用することをお勧めします。ELAMは、ASIC（データプレーンハードウェア）レベルで可視性を提供します。

ELAMを使用すると、転送パイプラインで処理されるパケットをキャプチャして、次のような重要な情報を明らかにすることができます。

- 入力インターフェイス
- 出力インターフェイス
- 転送の決定（L2/L3ルックアップ結果）

この方法は、実際のハードウェア転送パスを反映するため、コントロールプレーンツールに依存するよりも大幅に正確です。

ELAMが一度にキャプチャするパケットは1つだけです。そのため、フィルタリング基準は目的のトラフィック（送信元IP、宛先IP、TCPポートなど）に一致するように正確に定義する必要があります。 フィルタ範囲が広すぎると、目的のTCPフローではなく、ICMPやUDPなどの無関係なトラフィックがキャプチャされる危険性があります。

さらに、両方のトラフィック方向について、このプロセスを繰り返す必要があります。

- 送信元→宛先
- 宛先→送信元

vPCまたはECMPを使用する環境では、トラフィックを複数のパスにロードバランシングできます。その結果、転送トラフィックとリターントラフィックは異なるスイッチまたはインターフェ

イスを通過できます。このようなシナリオでは、完全な可視性を確保するために、関連する各NexusスイッチでELAMを実行する必要があります。

入力インターフェイスと出力インターフェイスを正確に特定することで、トラブルシューティングの範囲が大幅に縮小され、インターフェイスカウンタ、QoSポリシー、MTU設定、および潜在的な輻輳ポイントを、正確な転送パスに沿って集中的に検証できます。

ELAM設定（Nexus 9300クラウドスケール）

この例では、送信元IPが10.93.19.8、宛先IPが10.91.2.35、およびTCP宛先ポートが445のトラフィックをフィルタリングします。

ELAMの設定

```
<#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
```

トラフィックを生成した後、結果を取得します。

<#root>

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

report
```

リバーストラフィックキャプチャ（完全な可視性のために必須）

リターンパスを検証するには、送信元と宛先のIPアドレスを入れ替えて設定を繰り返します。

<#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
```

操作に関する注意事項

- ELAMは1つのパケットのみをキャプチャするため、キャプチャの開始時にトラフィックがアクティブに流れていることを確認します。
- フィルタは、無関係なトラフィックのキャプチャを避けるために正確である必要があります。
- vPC環境では、両方向でトラフィックのハッシュが異なる可能性があるため、両方のスイッチでELAMを実行します。
- 出力には、入力インターフェイス、出力インターフェイス、および転送の決定がハードウェアで表示され、データプレーンに対する正式な可視性が提供されます。

参考

Cisco Nexus 9000クラウドスケールASIC ELAMガイド

インターフェイスレベルの検証

インターフェイスレベルの検証により、NexusスイッチがTCPトラフィックに影響を与える制約や異常を導入していないことが確認されます。焦点は、設定、動作状態、およびハードウェアカウンタが、高パフォーマンスのデータプレーン転送で予想される動作と一致していることを確認することです。

設定の検証

- インターフェイスに制限ACLが適用されていないことを確認します。

<#root>

```
switch#

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

- 意図しないQoSポリシーがトラフィックに影響を与えていないことを検証します(インターフェイスレベルおよびグローバルQoS (キューイング、ポリシング、シェーピングを含む))。

<#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メンバーシップ、STPステート、IPアドレッシングを含む、レイヤ2またはレイヤ3の設定（スイッチポートとルーテッドインターフェイス）を確認します。

<#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

動作状態の検証

- MTUの一貫性を確認し、予期される設定（1500または9000バイトなど）に一致することを確認します。

<#root>

switch#

show interface ethernet1/1-2 | include MTU

- インターフェイスの速度とデュプレックスの設定を確認します。

<#root>

switch#

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

- インターフェイスの安定性を検証する（フラッピングやリンクの頻繁な遷移がない）。

<#root>

switch#

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

エラーカウンタの検証

- テスト前にカウンタをクリアします。

<#root>

switch#

```
clear counters interface all
```

- エラーカウンタを監視する（0以外の値のみ）:

<#root>

switch#

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

テスト後の検証

- TCPトラフィックテストを再実行し、カウンタを再度確認します。

<#root>

switch#

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

- カウンタの値は増えません。増えていても、レイヤ1または物理リンクエラー、CRC/FCSエラー、バッファオーバーラン/ドロップなどのハードウェア関連の問題が発生している可能性があります。

ルーティングとARPの安定性

ルーティングとARPの安定性を確保することは、Nexusスイッチが一貫したレイヤ3到達可能性を

持ち、TCPのパフォーマンスに影響を与えるような断続的な解決問題を引き起こさないことを確認するために重要です。ルーティングエントリまたはARP解決の不安定性は、パケット損失、遅延の増加、またはトラフィックのブラックホールの原因となる可能性があります。

検証基準

- 送信元と宛先のルーティングエントリが存在し、安定しており、頻繁に変更されない必要があります。
- ARPエントリは解決する必要があり、継続的に更新されたり欠落したりすることはありません。

<#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

トラフィックがCPUにパントされないことの確認

Cisco Nexus 9000スイッチでは、転送はハードウェア(ASIC)で実行され、CPUは通常のデータプレーン操作には関与しません。したがって、コントロールプレーンでホスト間TCPトラフィックを観察することは異常であり、パケットが例外または誤設定のためにパントされていることを示します。トラフィックをCPUで処理する必要が生じると、トラフィックはコントロールプレーンポリシング(CoPP)の対象になり、許可されたコントロールプレーンレートをトラフィックが超過した場合にドロップが観察されることが予想されます。

検証方法

- Ethanalyzerを使用してコントロールプレーンに到達するトラフィックをキャプチャします。

<#root>

switch#

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

予想される動作

- ホスト間TCPデータプレーントラフィックはCPUで確認できません。

予期しない動作

- フローと一致するパケットが表示されている場合、トラフィックはパントされています。これは次の原因で発生する可能性があります。
  - 優れたパケット処理 (TTL期限切れ、ACLロギング、リダイレクト)
  - 設定ミスまたはサポートされていない機能
  - 誤ったハードウェアプログラミング

パケット転送遅延の判別

Nexus 9000スイッチのパケット転送遅延は、パケットサイズ、転送モード、有効な機能によって異なります。シスコの仕様では、カットスルー転送での64バイトパケットの遅延が一般的に参照されています。

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

- カットスルーフォワーディング（Nexus 9000のデフォルト）:
  - パケット全体が受信される前に転送を開始します。
  - 遅延を最小限に抑えます（マイクロ秒未満～ 1 µs）。
- ストアアンドフォワード:
  - 転送する前にパケット全体を受信する必要があります。
  - パケットサイズに比例した遅延が追加されます。

機能が追加されると、遅延が増加する可能性があります。

- VXLANカプセル化/カプセル化解除
- ACLルックアップ（TCAM処理）
- QoS分類およびキューイング
- テレメトリ(NetFlow、ERSPAN、sFlow)
- 輻輳時のバッファリング

ただし:

- これらの操作は、ハードウェアパイプラインで実行されます。

遅延が著しく増加する現実的なシナリオは、輻輳のみです。

- パケットは出力キューにバッファリングされます。
- 遅延は次の要因によって異なります。
  - Queue Depth
  - インターフェイス使用率
  - QoS ポリシー

このような場合でも、次のことを行います。

- 遅延は通常、マイクロ秒から数百マイクロ秒の範囲です。
- ミリ秒レベルの遅延が続くと、次のようになります。
  - 重度の輻輳
  - パフォーマンスの問題
  - QoSまたはバッファリングの誤設定

SPANからCPU（データプレーンのパケットキャプチャ）

これにより、データプレーントラフィックをコントロールプレーンにミラーリングしてパケットキャプチャを行い、.pcapngファイルにエクスポートできるため、Wiresharkで詳細な分析を行うことができます。

コンフィギュレーション

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

キャプチャの実行

<#root>

```
switch#
```

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

技術的な考慮事項

- CPUにミラーリングされたトラフィックは、コントロールプレーンポリシング(CoPP)の対象となります。
- トラフィックがCoPPを超える場合：
  - パケットはコントロールプレーンでのみドロップできる
  - これにより、解析中に誤検出が発生します。
- 低から中程度のトラフィックのシナリオでは、SPANからCPUへの移行を推奨します。
- 高スループット環境：
  - ローカルSPAN（外部アナライザ）を使用
  - リモートキャプチャにERSPANを使用

メソッド	利点	制限
SPAN	正確、カプセル化なし	物理的な接続が必要です。
ERSPAN	リモートキャプチャ機能	ネットワーク輻輳の影響を受けやすい

コントロールプレーンのレート制限の検証

SPANからCPUへのキャプチャの信頼性を確保するには、レート制限が原因でコントロールプレーンでミラーパケットがドロップされていないことを検証する必要があります。

検証コマンド

```
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
```

検証方法

- コマンドを3秒以内の間隔で実行します。
- SPAN関連のドロップカウンタを確認します。

解釈

- SPAN行のドロップカウンタの増分は、信頼できるキャプチャを示しません。
- ドロップカウンタの増加はコントロールプレーンでのパケット損失を示し、キャプチャの信頼性を低下させます。

ドロップが確認された場合、キャプチャ方式をSPANまたはERSPANに変更する必要があります。

TCPの前のICMPベースの検証

ICMPテストは、複雑なTCP分析を実行する前に、データプレーンの整合性のベースライン検証を提供します。ICMPはステートレスでシンプルなため、パケット損失、重複、またはパスの不整合をすばやく検出できます。

SPANキャプチャで想定される動作

- 各ICMPパケットは2回表示される場合があります。
  - 入力に対して1回
  - 出力時に1回
- 標準pingの場合：
  - 2パケット→エコー要求
  - 2パケット→エコー応答

これにより、データプレーンでパケット損失が発生していないことが正しく転送されていることを確認できます。

異常行動

- 重複パケットまたは非対称パケットのカウントが欠落している場合は、パケット損失またはキャプチャの潜在的な制限を示しています。
- 断続的なタイムアウトは、レイヤ1の問題、輻輳、またはアップストリームの問題を示唆します。

ICMPトラフィックが損失なしで常に転送される場合、TCPトラフィックもレイヤ2/3で正しく転送されている可能性が高くなります。

パケットキャプチャを使用したNexusスイッチの転送遅延の確認

SPAN to CPU（またはSPAN/ERSPAN）を使用してトラフィックをキャプチャすると、各パケットが2回観察されます。1回は入力で、1回は出力で観察されます。 この重複を使用して、同じパケットの両方のインスタンス間の時間差を計算することで、Nexusスイッチによる転送遅延を見積もることができます。

実際には、この遅延は、以前にキャプチャしたICMPトラフィックを使用して、重複したエコー要求パケットとエコー応答パケットの時間デルタを比較することで測定できます。これにより、スイッチの転送パフォーマンスに対するシンプルで効果的なベースラインが提供されます。より詳細な分析が必要な場合は、フローをキャプチャし、重複したTCPパケット間の時間差を測定することで、同じ方法をTCPトラフィックに適用できます。

方法

- パケットとその重複（同じシーケンス番号）を特定します。
- 入力コピーと出力コピーの時間差を測定します。
- このデルタは、ミラーリングとタイムスタンプのオーバーヘッドを含む可能性があるため、スイッチの転送遅延の上限を表します。

Wiresharkの設定

- 時間デルタ表示を有効にする：

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

- 時間デルタ用のカスタム列を追加します。

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

- 関連するトラフィックのフィルタリング（例）：

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

- シーケンス番号またはTCPストリームでパケットをソートする：

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

解釈

- 複製されたパケット間の時間デルタは、マイクロ秒範囲に入ります。
  - この場合、Nexusスイッチではパケット転送に遅延が発生しません。
- 一貫した低いデルタは、ハードウェアベースの転送パフォーマンスを確認します。
- デルタが高いか一貫していない場合は、次の可能性があります。
  - 輻輳またはバッファリング

参照資料

- Cisco Nexus 9000シリーズのデータシート（遅延仕様）
- Cisco Nexus 9000アーキテクチャに関するホワイトペーパー
  Cisco NX-OSのQoSおよびバッファリング動作

送信元ホストからのTCPトラフィック分析のパケットキャプチャ

このセクションでは、前述の架空のケースを使用して、プロファイル設定を含むWiresharkのTCPパケットキャプチャを分析するための詳細な方法論を提供します。表示されている画像はWiresharkから直接取得したものです。このシナリオの特徴は次のとおりです。

あるユーザーが、約6.5 TBのアプリケーション・データセットのバックアップ・プロセスを特定しました。以前は約9時間で完了していましたが、現在では約21時間かかっています。アクセス可能なネットワークデバイスは、送信元サーバ(10.93.19.8)に接続されたCisco Nexus 9300スイッチだけです。 スイッチインターフェイスに設定されているMTUは9000バイト (ジャンボフレーム) ですが、サーバのMTUは不明です。送信元サーバからのパケットキャプチャが利用可能で、以前のすべてのNexus検証手順は異常が検出されることなく完了しています。

主な所見と制約

- Nexusスイッチは除外されています。
  - パケットドロップなし
  - インターフェイスエラーなし
  - QoSまたはACLへの影響なし
  - ハードウェア転送の確認
- インターフェイス 設定
  - アクセス ポート
  - MTU:9000バイト
- 利用可能なデータ：
  - 送信元でのパケットキャプチャ
  - エンドツーエンドMTUの知識
    - 1,472バイトのデータを持つ1500バイトのパケットを使用して、pingはフラグメンテーションなしで正常に完了しました。
- データがありません：
  - 宛先の可視性
  - 宛先サーバーでパケットキャプチャを使用できません。

Wiresharkでは、実行する特定のタイプの分析に合わせてカスタマイズしたカスタムプロファイルを作成できます。

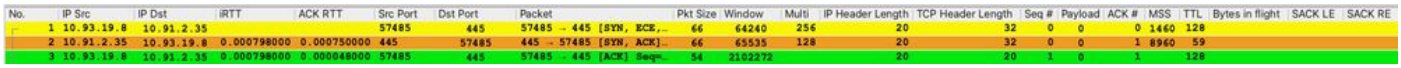
列の説明

- tcp.analysis.initial_rtt(iRTT):TCP 3ウェイハンドシェイクに基づいて、最初のラウンドトリップ時間を推定します。
- tcp.analysis.ack_rtt(ACK RTT):TCPセグメントとそれに対応する確認応答の間の時間を測定します。

- tcp.window_size (Window)：スケーリングが適用される前に、受信側がアドバタイズしたTCPウィンドウサイズを示します。
- tcp.options.wscale.multiplier (Multi)：有効な受信ウィンドウの計算に使用するウィンドウスケーリング係数を表します。
- tcp.seq (Seq#): TCPセグメントの最初のバイトのシーケンス番号を表示します。
- tcp.len（ペイロード）：そのセグメントのTCPペイロードのサイズをバイト単位で示します。
- tcp.ack(ACK#)：送信側からの次の予測バイトを示します（累積確認応答）。
- tcp.options.mss_val(MSS):TCPハンドシェイク中にアドバタイズされた最大セグメントサイズ(MSS)を表示します。
- ip.ttl(TTL):Time To Live（TTL；存続可能時間）の値が表示されます。これは、ホップカウントやルーティング動作の識別に役立ちます。
- tcp.analysis.bytes_in_flight (Bytes in Flight)：現在転送中の確認応答されていないデータの量を表します。

TCP 3ウェイハンドシェイクの分析

TCP 3ウェイハンドシェイクのキャプチャは、セッションの動作を定義するMSS、ウィンドウスケール、SACKなどの重要なパラメータが含まれているため、必須です。
この情報がないと、TCP分析が不完全になり、パフォーマンスに関する誤った結論や根本原因を引き起こす可能性があります。



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			

トラフィックの識別

パケットキャプチャから：

- 送信元IPアドレス：10.93.19.8
- 宛先IPアドレス：10.91.2.35

初期ラウンドトリップ時間(iRTT)分析

初期RTT(iRTT)は次のように計算されます。

- iRTT = 798マイクロ秒

この値は、次の式から導き出されます。

- Packet 2(SYN-ACK)ACK RTT:750 µs →宛先がSYNに応答する時間。
- Packet 3(ACK)ACK RTT:48 µs →送信元がSYN-ACKを確認応答するまでの時間。

遅延の大部分（最大94 %）は転送パス(クライアント→サーバ→クライアント)で発生しますが、送信元からの応答時間は最小で、クライアント上のCPUやアプリケーションの遅延がないことを示します。

TCPポートの識別

- 宛先TCPポート：445

ポート445は、ファイル共有、ネットワークドライブ、およびWindows認証サービスで一般的に使用されるMicrosoft Server Message Block(SMB)に対応します。このプロトコルは遅延とスループットの両方に敏感で、TCPの効率とネットワークの安定性に大きく依存します。

TCPウィンドウサイズ分析

- ソースウィンドウ（拡大/縮小）: 64,240バイト
- 宛先ウィンドウ：65,535バイト

TCPウィンドウは、確認応答を待機する前に送信できるデータ量を表します。この場合、送信元は宛先よりもわずかに制限が厳しくなります。最近の環境では、これらの値は比較的小さく、特にRTTが増加するとスループットが制限される可能性があります。

理論上の最大スループットは、次の式で推定できます。

スループット= TCPウィンドウサイズ/RTT

観察された値を置き換える：

- TCPウィンドウサイズ= 64,240バイト
- RTT = 798マイクロ秒= 0.000798秒

スループット≈ 64,240 / 0.000798 ≈ 80.5 MB/秒(~ 644 Mbps)

これは、次の場合の上限スループットを表します。

- パケット損失なし

- 再送信なし
- 理想的なネットワーク条件

スループット、転送時間、および必要な条件分析

現在の644 Mbpsのスループットでは、6.5 TBのファイルの転送には約23.5時間かかり、観察されたパフォーマンスの低下に合わせられます。9時間の転送ウィンドウを実現するには、スループットが約1.68 Gbpsに増加する必要があり、より大きなTCPウィンドウ（約2.7倍）または大幅に低いRTT（約291 µs）が必要です。

現在の状況（64 KBウィンドウと約798 µs RTT）では、9時間の目標に到達できません。これは、TCPスループットが帯域幅遅延積によって制約されるためです。ウィンドウサイズを増やしたり遅延を減らしたりしなければ、プロトコルは使用可能な帯域幅を増やすことができず、目標を達成できなくなります。

シナリオ	スループット	推定転送時間(6.5 TB)	必要なTCPウィンドウ	必要なRTT
現在の状態	644 Mbps（最大80.5 MB/秒）	~ 23.5時間	64 KB	798マイクロ秒
目標（9時間）	1683 Mbps以下（210 MB/s以下）	9 時間	最大172 KB	~291マイクロ秒

これは以前は正常に動作しており、ネットワーク、アプリケーション、送信元、または宛先で変更が発生したことを示しています。この初期分析だけでは、現在のTCPウィンドウサイズとRTT条件の下では、9時間の目標を達成することは不可能です。

次の表は、RTTとTCPのウィンドウサイズの増減に伴うスループットの変化を比較したものです。

スループットに対するRTTの影響（固定ウィンドウサイズ= 64,240バイト）

RTT	スループット（MB/秒）	スループット（Mbps）
200 µs(0.0002 s)	最大321 MB/秒	最大2,568 Mbps
798マイクロ秒(0.000798マイクロ秒)	最大80.5 MB/秒	最大644 Mbps
2ミリ秒（0.002秒）	最大32.1 MB/秒	最大257 Mbps

RTT	スループット（MB/秒）	スループット（Mbps）
10ミリ秒（0.01秒）	最大6.4 MB/秒	最大51 Mbps

TCPウィンドウサイズの影響（固定RTT = 798 µs）

TCPウィンドウサイズ	スループット（MB/秒）	スループット（Mbps）
16 KB(16,384 B)	最大20.5 MB/秒	最大164 Mbps
64 KB(64,240 B)	最大80.5 MB/秒	最大644 Mbps
256 KB(262,144 B)	最大328 MB/秒	最大2,624 Mbps
1 MB(1,048,576 B)	約1,314 MB/秒	~ 10.5 Gbps

技術解釈

- スループットはRTTに反比例し→遅延が大きいほどパフォーマンスが低下します。
- スループットはTCPウィンドウサイズに正比例し→ウィンドウが大きいほど容量が増加します。
- ウィンドウサイズが小さいと、低遅延の環境でもスループットが著しく制限されます。
- 高速ネットワーク(10G+)では、帯域幅を最大限に活用するためにウィンドウスケーリングが必要です。

これは、RTTとTCPウィンドウサイズの両方がTCPパフォーマンスの重要な要素であり、スループットの問題をトラブルシューティングする際に一緒に分析する必要があることを示しています。

IPおよびTCPヘッダー長

- IPヘッダー長：20バイト
- TCPヘッダー長：32バイト

20バイトのIPヘッダーは、IPオプションが存在しないことを示します。32バイトのTCPヘッダーは、TCPオプションが使用されていることを確認し、基本ヘッダーの12バイトを追加します。これらのオプションには通常、MSS、ウィンドウスケール、およびSACK許可が含まれます。

TCPオプション分析およびTTL

両方のエンドポイントでSelective Acknowledgment(SACK)が有効になっている。これは写真には表示されていません。SACKを使用すると、受信側は非連続データブロックを確認応答し、正常に受信されたセグメントを送信側に正確に通知できます。

たとえば、セグメント1000 ~ 2000および3000 ~ 4000が受信されても、2000 ~ 3000が欠落している場合、受信側はこれを明示的に示すことができます。SACKを使用しない場合、送信者はギャップの後にすべてのデータを再送信します。SACKを使用する場合、欠落した部分のみが再送信されます。これにより、パケット損失が発生する環境のパフォーマンスが大幅に向上します。

パケット1(SYN)分析

- シーケンス番号：0（Wireshark正規化）
- ペイロード：0バイト
- ACK番号：0
- MSS:1460バイト
- TTL:128

Wiresharkは読みやすいようにシーケンス番号をゼロに正規化しますが、実際には大きなランダム値です。接続の確立中にペイロードが存在しないことが予期されます。1460バイトのMSS値は、1500バイトのMTU（20バイトのIPヘッダー+ 20バイトのTCPヘッダー）を示します。 TTLが128の場合はWindowsベースのホストを使用できます。キャプチャでこの値が表示された場合、キャプチャがレイヤ2経由で送信元またはその近くで取得された可能性が高いことを示します。

パケット2(SYN-ACK)分析

- ACK番号：1

SYNフラグは、ペイロードが存在しない場合でも1つのシーケンス番号を消費するため、ACK値は1です。したがって、ACK = SEQ + 1となります。

- TTL:59

確認されたTTLが59の場合は、初期TTLが64であることを示します。これは、パケットがおよそ5つのルーティングホップ(64 − 59 = 5)を通過したことを意味します。 各ルーテッドホップはTTLを1ずつ減らします。

フラグメンテーションリスクとネットワークへの影響

約5つのルーティングホップが存在すると、特にMTUの不一致とフラグメンテーションに関連して、潜在的なパフォーマンスリスクが発生します。

中間リンクのMTUが元のパケットサイズよりも小さい場合は、フラグメンテーションが発生する可能性があります。これにより、次のようないくつかの結果が生じます。

- フラグメンテーションと再構成のオーバーヘッドによる遅延の増大
- 単一のフラグメントを失うとパケット全体の再送信が必要になるため、パケット損失の可能性が高くなります。
- TCPが損失を輻輳と見なし、その送信レートを下げるため、スループットが低下します。
- フラグメンテーションを処理するネットワークデバイスでのCPU使用率の増加。
- ICMPがブロックされ、サイレントパケットドロップが発生する場合の、Path MTU Discovery(PMTUD)障害のリスク。

これらの要因を考慮すると、パス全体でMTUの一貫性を確保するか、必要に応じてMSSクランプを実装することが重要です。

TCP RTT分析：ACK RTTと初期RTT

ACK RTTがiRTTよりも大きい場合は、TCPハンドシェイク中に確立されたベースラインと比較して遅延が増加していることを示しています。

つまり、ネットワークまたはエンドポイントがセッション中にさらに遅延を発生させており、その原因は通常、次のとおりです。

- ネットワークの輻輳またはキューイング
- 受信者またはアプリケーションの処理遅延
- 中間デバイス (ファイアウォール、ロードバランサ)
- 再転送

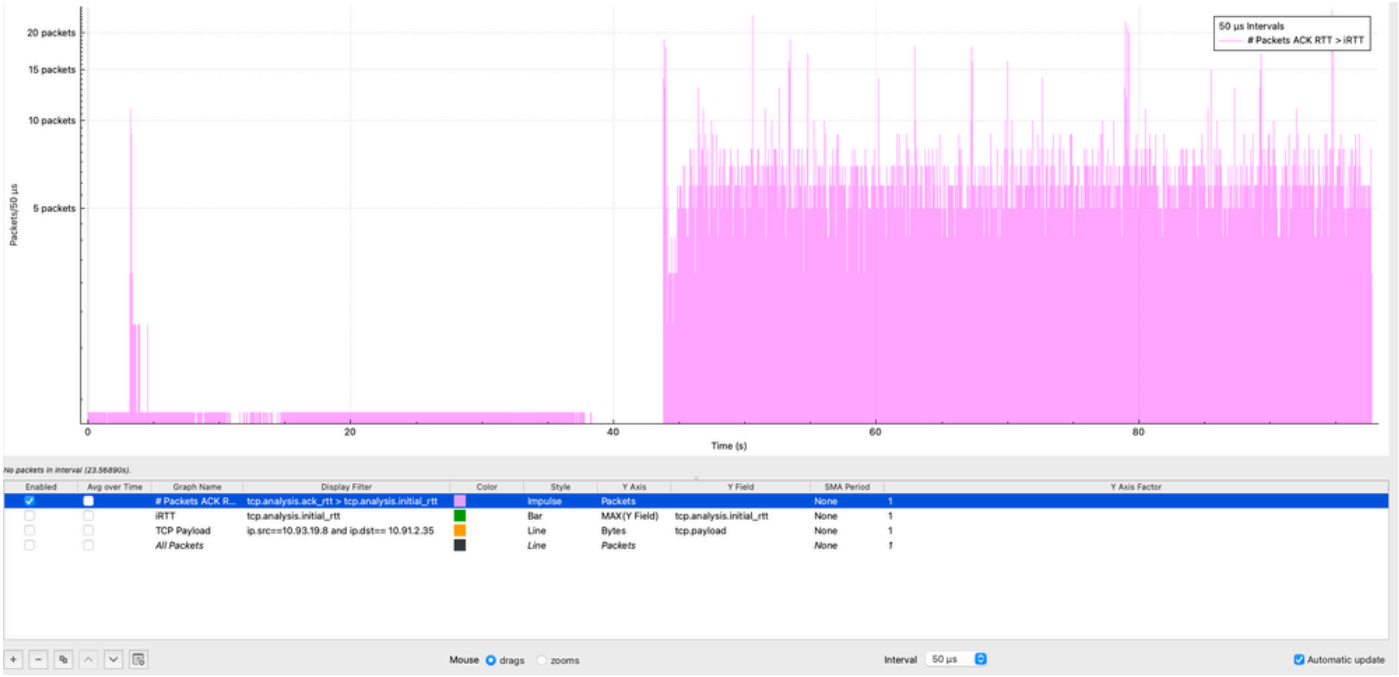
この状態がTCPセッション全体を通して続くと、次の問題が発生します。

- TCPスループットの低下
- 非効率的なウィンドウ使用率
- アプリケーションパフォーマンスの低下

Wiresharkでは、I/O GraphsのI/O Graphs機能のStatistics → I/O Graphsを使用し、表示フィルタ(tcp.analysis.ack_rtt > tcp.analysis.initial_rtt)を適用し、インパルススタイルを選択し、Y AxisをPacketsに設定し、50マイクロ秒の間隔を使用することで、ACK RTT > iRTTの状態の頻度を視覚化できます。

グラフでは、紫色のインパルスが、各50マイクロ秒間隔内でこの条件を満たすパケットの数を表しています。前述したように、この状態はパケットキャプチャ全体を通じて継続し、セッション中の遅延が常に初期ベースラインよりも高いことを示します。この動作は、一時的な状態ではな

く、継続的なパフォーマンスの低下を強く示唆しており、エンドツーエンドのパスにおける輻輳、バッファリング、エンドポイントの処理遅延などの潜在的な原因を調査する必要性を強めます。



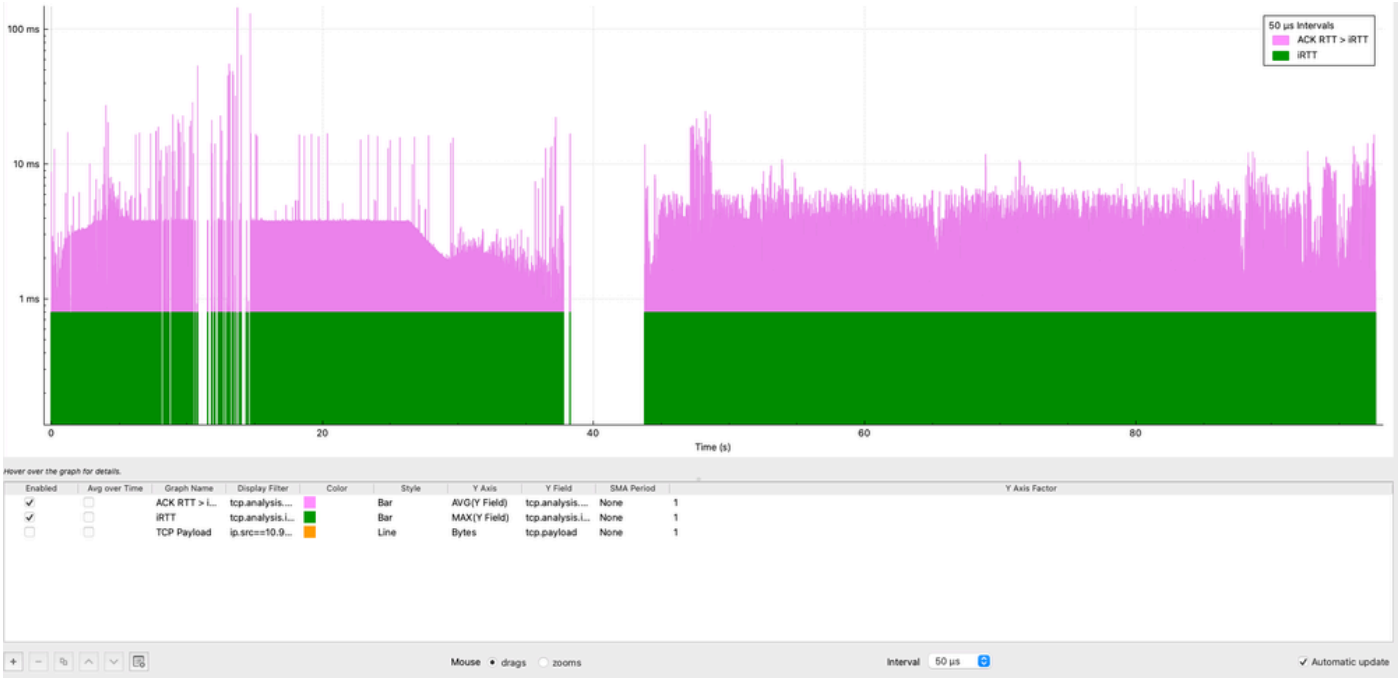
また、iRTTがどのくらいの頻度で超過しているのかを判断することも重要です。Wiresharkではフィールド間の差を直接使用できませんが、I/Oグラフを使用して視覚的に比較できます。

- Statistics → I/O Graphsの順に移動します。
- グラフ1:
  - 表示フィルタ：tcp.analysis.ack_rtt > tcp.analysis.initial_rtt
  - スタイル：バー
  - Y軸：平均
  - Yフィールド：tcp.analysis.ack_rtt
  - 間隔：50マイクロ秒
- グラフ2:
  - 表示フィルタ： tcp.analysis.initial_rtt
  - スタイル：バー
  - Y軸：最大
  - Yフィールド：tcp.analysis.initial_rtt
- 次にグラフを右クリックし、ログスケールを有効にします。

この可視化では、紫色のグラフはACK RTT > iRTTの状態を表しており、これはTCPセッション全体を通じて一貫して存在します。このデータは、複数のピークが11ミリ秒に達し、ベースラインのiRTTの11倍から100倍を表す最大スパイクが100ミリ秒を超える持続的な遅延の上昇を示しています。

この動作により、遅延の増加が一時的ではなく永続的であることが確認されます。これは、時間

の経過とともにセッションに影響を与えるシステム上の問題があることを示します。このような継続的な偏差は、ネットワークの輻輳、バッファリング(bufferbloat)、エンドポイントの処理遅延などの要因を強く示唆します。



TCP再送信とスプリアス再送信分析

このセクションでは、再送信を経時的に分析してTCPの信頼性を評価します。これにより、パケット損失がパフォーマンスの低下に寄与しているかどうかを検証できます。

時間の経過に伴うTCPの再送信

このグラフは、時間の経過に伴うTCP再伝送の分布を示しています。合計42の再送信が確認され、トラフィック全体のわずか0.00125 %でした。

このレベルの再送信は無視できる程度であり、パケット損失がこのシナリオの要因ではないことを明確に示します。

Wiresharkの設定（TCP再送信）

```
Statistics → I/O Graphs
```

- 表示フィルタ:

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

- [スタイル]: [インパルス]または[バー]
- Y軸：パケット
- 間隔：1秒

TCPスプリアス再送信

このグラフは、送信元10.93.19.8で生成されたTCPスプリアス再送信数を1秒インターバルで示しています。

Wiresharkでは、TCPスプリアス再送信は、実際には失われていないセグメントをホストが再送信したことを示します。元のパケットは受信側に正常に到達しましたが、送信側は不正確なタイミング推定のために損失を誤って認識しました。この動作は実際のパケット損失を示すものではなく、送信側での非効率的な再送信ロジックです。

このキャプチャの内容：

- 送信元10.93.19.8は、わずか8マイクロ秒後にパケットを再送信します。
- 通常の再送信タイマーは約200ミリ秒です。

これにより、再送信動作がネットワークではなく、送信元TCPスタックによって完全に制御されていることが確認できます。

観察されたスプリアス再送信の総数は1,112で、キャプチャされたトラフィック全体の0.0332 %を表します。

Wiresharkの設定（TCPスプリアス再送信）

```
Statistics → I/O Graphs
```

- 表示フィルタ:

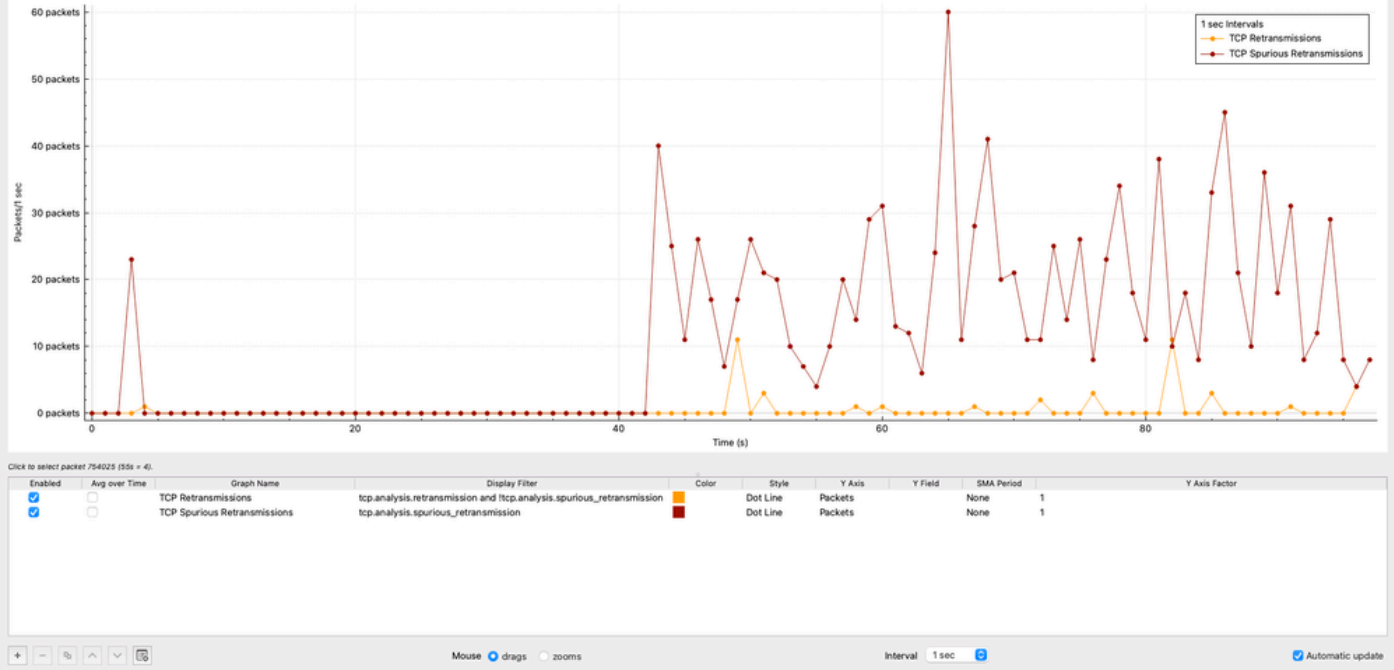
```
tcp.analysis.spurious_retransmission and ip.src==10.93.19.8
```

- [スタイル]: [インパルス]または[バー]
- Y軸：パケット
- 間隔：1秒

技術解釈

- 実際の再送信の割合が非常に低いことから、ネットワークでパケット損失が発生していないことがわかります。
- スプリアス再送信が存在する場合は、発信元ホストによる再送信の決定が早すぎる可能性があります。
- この動作は効率にわずかに影響を与える可能性がありますが、深刻なスループット低下の主な原因ではありません。

この分析により、問題がネットワークの信頼性ではなく、TCPの動作、遅延、またはエンドポイントのパフォーマンスに関連していることが一層明らかになります。

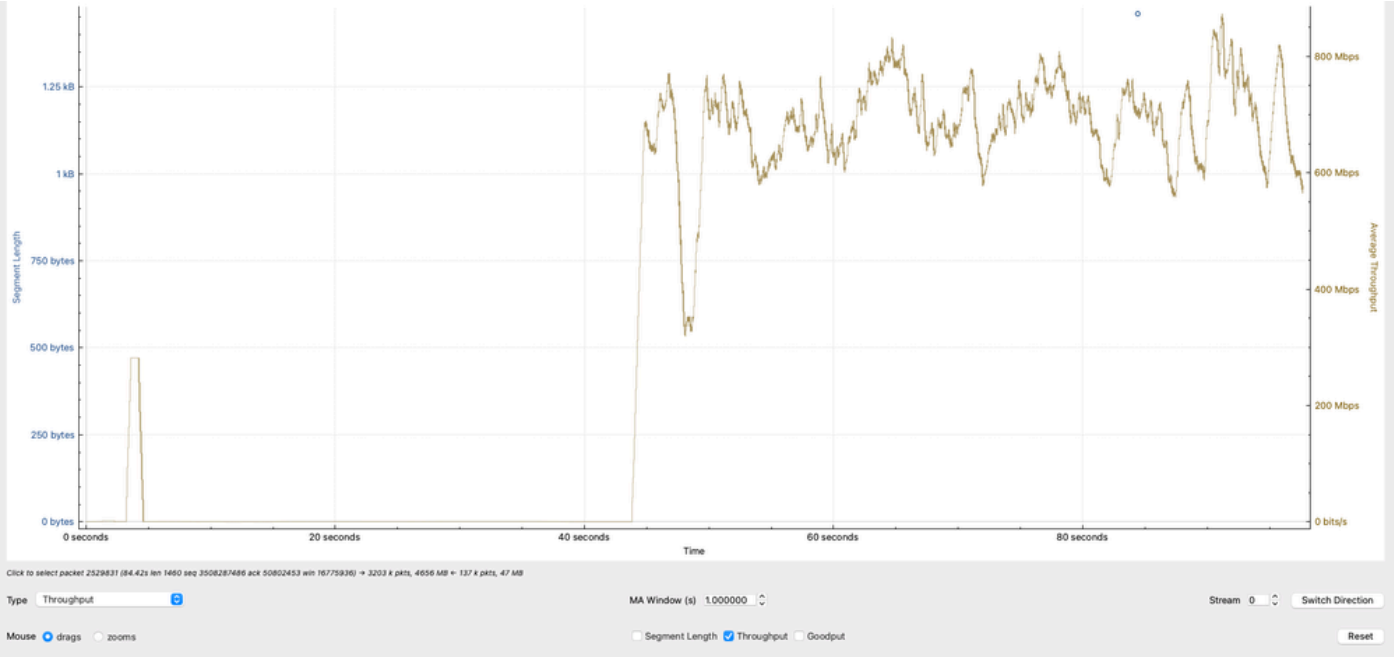


効果的なスループット分析

このグラフは、TCPペイロード（実際に転送されるデータ）に基づいて計算された実効スループットをメガビット/秒で示しています。観測されたスループットは主に600 ~ 800 Mbpsの間で変動しており、ネットワークでデータ転送が活発に行われている間は、帯域幅のポテンシャルが高くなっていないことを示しています。

Wiresharkの設定（実効スループット）

Statistics → TCP Streams Graphs → Throughout



技術解釈

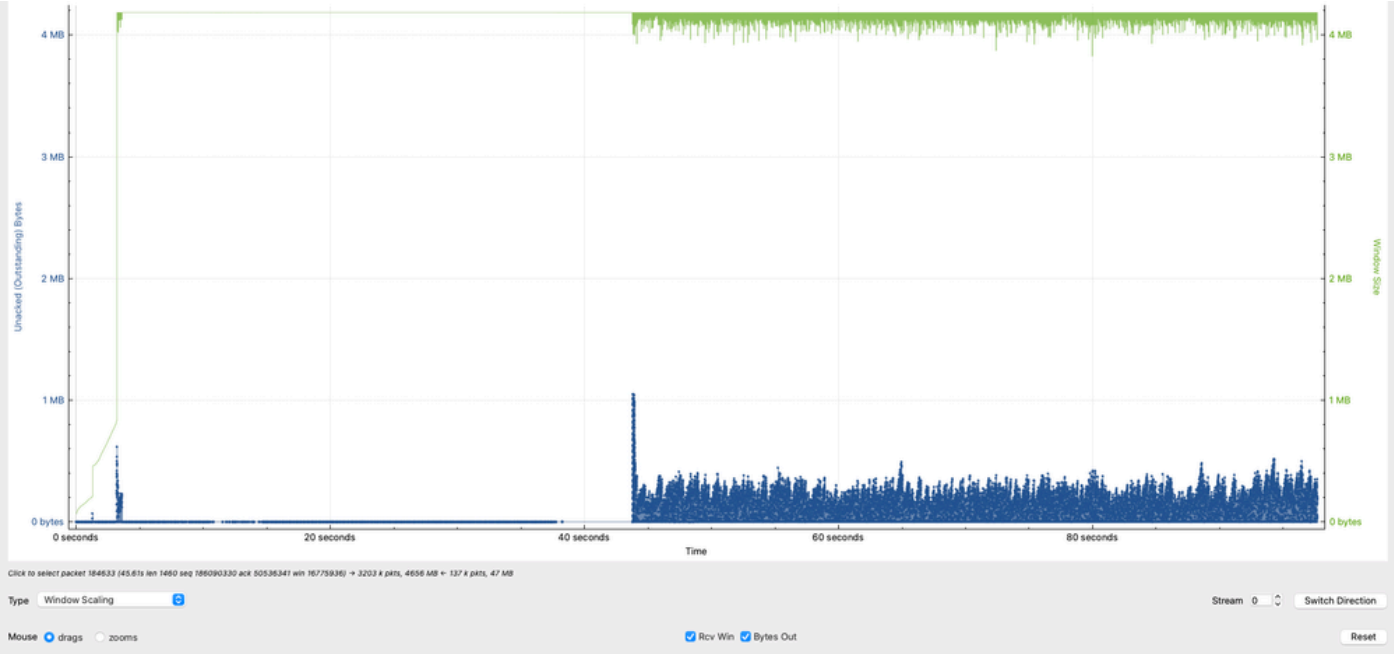
- 600 ~ 800 Mbpsのスループット範囲は、TCPウィンドウサイズとRTTに基づく以前の計算と一致します。
- スループットの変動は次を反映します。
  - RTTの変動
  - TCP輻輳制御の調整
  - アプリケーションペーシングまたはバッファリング
- スループットはラインレート（10Gなど）に近づいていないため、制限は物理的な帯域幅ではなく、TCP効率の制約です。
- この分析により、観察されたスループットがTCPの制限（ウィンドウサイズと遅延）と一致することが確認され、ボトルネックがパケット損失やインターフェイス容量によるものではなく、トランスポート層の動作とエンドポイントの状態によるものであることが補強されます。

移動中のデータ（TCPウィンドウ）の分析

このグラフは、受信側の容量と実際の伝送中のデータ（伝送中のバイト数）を比較することによって、TCPセッションのクリティカルな動作を示しています。

- 緑色の線は、10.91.2.35（レシーバ）が受け入れ可能なTCPデータの量（実効受信ウィンドウ）を表しています。

- 青い線は、10.93.19.8 (送信者) から現在転送されているTCPデータの量を表します。



Flightで観測されたデータは約1 MBで、その他のピークは約8 KBと5 KBですが、主に1 KBと250 KBの間で集中しています。

これは、受信側は大量のデータを処理できますが、送信側は常に使用可能なウィンドウを使用していないことを示します。

Wiresharkの設定 (送信中のデータとウィンドウのデータ)

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

技術解釈

- 受信側(10.91.2.35)は、より多くのデータを受信できることを示す、非常に大きなウィンドウをアドバタイズします。
- 送信側(10.93.19.8)は、使用可能なウィンドウを十分に活用していません。これは、値が低く一貫性のない「Data in Flight」で示されています。
  - 送信者は、スループットを最大化するために、受信側のアドバタイズされたウィンドウ(~ 1 MB)に近いData in Flight(DIF)値を維持するのが理想的です。
  - 通信中の高いデータレベルを維持できないと、スループットが直接制限され、ネットワーク容量の問題ではなく、発信元でのTCPの非効率性を示す強力なインジケータとなります。

TCPペイロードとMSSの経時的分析

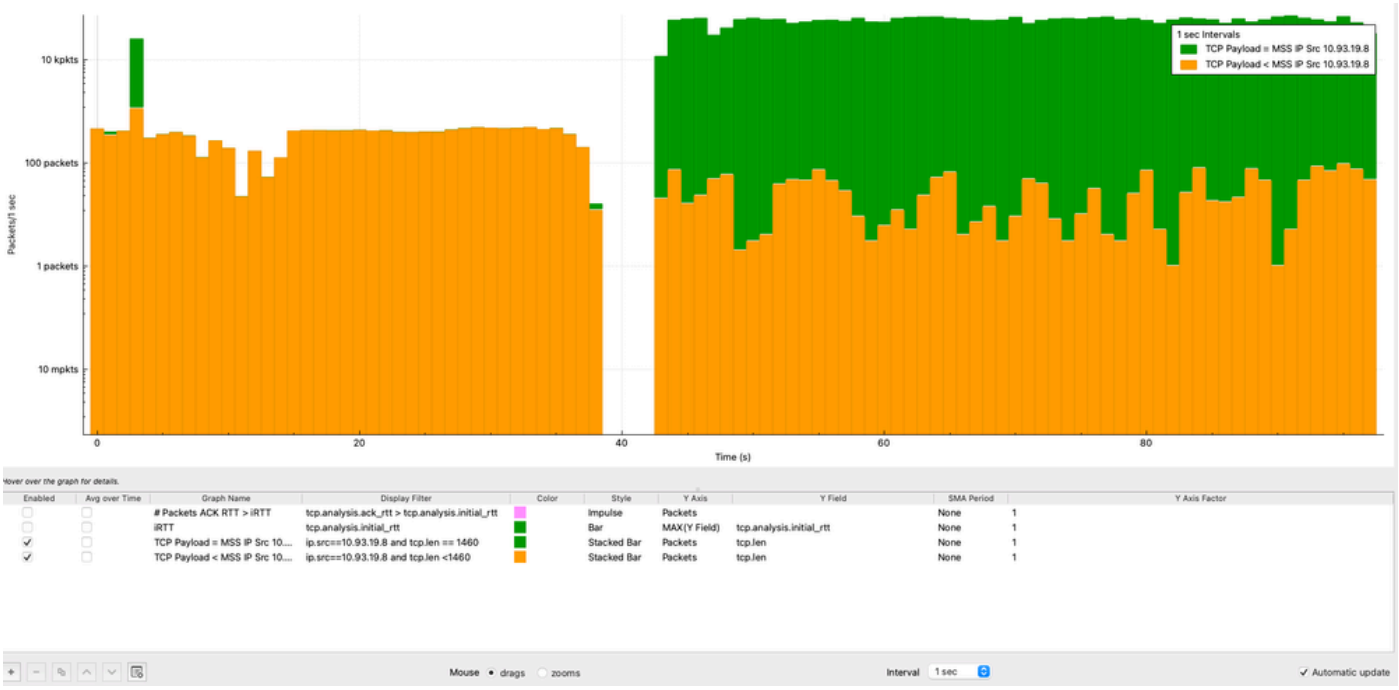
MSSに対するTCPペイロードサイズの経時的分析は、送信側が各TCPセグメントを効率的に使用しているかどうかを判断するのに役立ちます。この分析は、送信元IPアドレス(10.93.19.8)の観点から実行されます。

Wiresharkでは、グラフは次のように設定されます。

- グラフ1（MSSサイズパケット）:
  - 表示フィルタ：ip.src==10.93.19.8、tcp.len == 1460
  - スタイル：積み上げ横棒
  - Y軸：パケット
  - 間隔：1秒
- グラフ2(すべてのパケット≤MSS):
  - Display Filter（表示フィルタ）:ip.src==10.93.19.8 and tcp.len <= 1460
  - スタイル：積み上げ横棒
  - Y軸：パケット
  - 間隔：1秒
- より良い視覚化のために対数スケールを適用

分析から：

- パケットの大部分（1秒あたり10,000パケット以上）は、常に1460バイトのMSS値に達します。
- 通常のTCPの動作（ACK、セグメンテーション、またはストリーム終了データ）により、パケットの一部分ほどペイロードを伝送しません。



根本原因分析(RCA):TCPパフォーマンスの低下

この分析により、TCPパフォーマンス問題の根本原因を特定するには、ネットワークがパフォーマンス低下の主な原因であると仮定するのではなく、全体的なエンドツーエンドのアプローチが必要であることが明らかになりました。

Cisco Nexus 9300スイッチに対して、インターフェイスカウンタ、QoSポリシー、ルーティングとARPの安定性、CPUパント検証、SPANベースのパケットキャプチャ、およびELAMを使用したASICレベルの転送検証など、さまざまな検証が実施されました。すべての結果で、スイッチが期待されるパラメータの範囲内で動作していることが確認されました。

- パケットドロップなし
- 異常遅延なし（マイクロ秒範囲）
- QoSまたはコントロールプレーンへの影響なし
- 正しいハードウェア転送

さらに、TCP分析では次のことが明らかになりました。

- 再送信が無視できる(0.00125 %)
- パケット損失の証拠がない
- ソースでの一貫したMSS使用率
- TCPウィンドウとRTTの制約に合わせたスループット
- 利用可能なTCPウィンドウの使用率が低い（Data in Flight分析）
- ネットワークがボトルネックではない
- ソースサーバーがパフォーマンスを制限しています

結論

このパフォーマンスの低下は、ジャンボ対応環境でMTU 1500で動作する送信元サーバによって引き起こされ、使用可能なネットワーク容量の効率的な使用を妨げます。

ソリューション

宛先およびネットワークインフラストラクチャに合わせて、発信元サーバのMTUを1500バイトから9000バイトに増やします。利点：

- より大きなTCPセグメントを有効にする
- パケットオーバーヘッドの削減
- 全体的なスループットの向上

技術的な復習

この分析の重要なポイントは、ネットワークのパフォーマンスをトラブルシューティングする際に、結論を早めないようにすることです。最初に問題をネットワークに関連付けることは一般的ですが、この事例から、ネットワークがデータプレーンパス全体で正常に機能していることが明らかになります。 ハンドシェイクパラメータ、RTT動作、ウィンドウ使用率、再送信、ペイロード効率など、送信元と宛先の両方の観点から詳細なTCP分析を実行することによってのみ、真のボトルネックを正確に特定することができました。

時間をかけてTCPの動作を詳細に分析することで、誤診断を防止し、不要なネットワーク変更を減らし、修復の作業を実際の根本原因に向けることができます。

翻訳について
シスコは世界中のユーザにそれぞれの言語でサポート コンテンツを提供するために、機械と人による翻訳を組み合わせて、本ドキュメントを翻訳しています。ただし、最高度の機械翻訳であっても、専門家による翻訳のような正確性は確保されません。シスコは、これら翻訳の正確性について法的責任を負いません。原典である英語版（リンクからアクセス可能）もあわせて参照することを推奨します。