



Part 1: PLAN

Implement WAN Optimization

- **Describe How WAN Optimization Works**
- **Describe the Solution Architecture**
- **Manage Your Appliances**

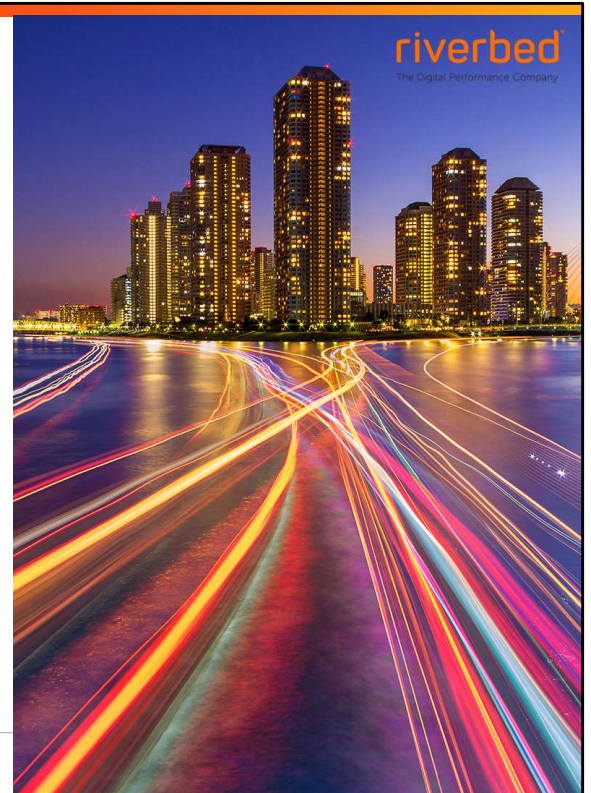


Learning Objectives

After completing this module, you will be able to:

- Describe how optimization addresses network performance challenges.
- Alleviate bandwidth restrictions.
- Increase TCP performance.
- Mitigate RTT at the application layer.
- Visualize the effects of optimization.
- Describe elements of a complete WAN optimization solution.

© 2020 Riverbed Technology, Inc. All rights reserved.





Key Points



Optimization has a multifaceted approach – Bandwidth, TCP and the application layer.

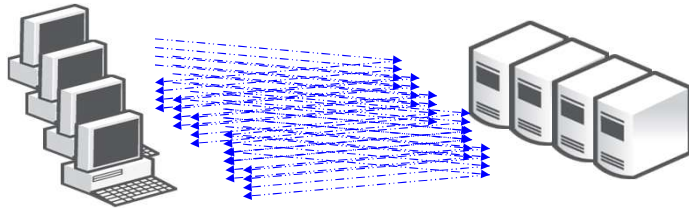


WAN Bandwidth appears to be increased yet is largely irrelevant to the user; their experience comes from the LAN throughput.



With application layer optimization, the effective latency is reduced because the user suffers the WAN RTT less often.

Why Optimize – in a non-WAN era



Local Servers have
almost zero RTT and
almost limitless
bandwidth

Chatty and bulky
applications still perform
well

© 2020 Riverbed Technology, Inc. All rights reserved.

riverbed 6

The SteelHead was invented to solve the problem of applications running over the WAN. There are many operational and economic advantages to moving the servers from remote branches to Datacenters. Many of the protocols over which the applications work were designed with LAN performance in mind.

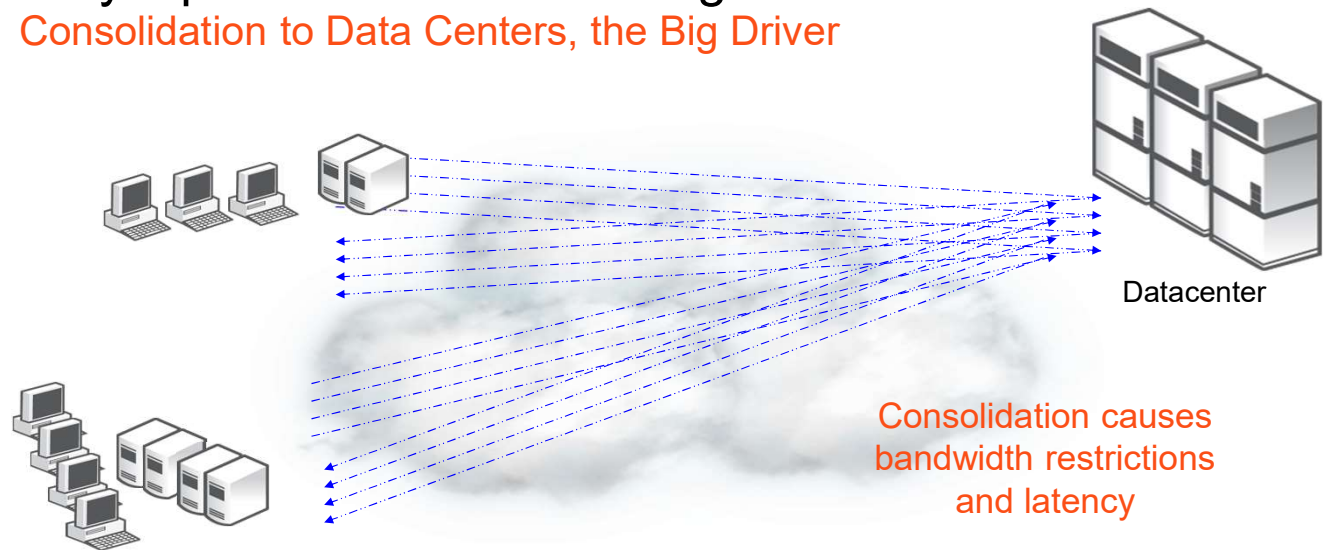
If a developer works from the premise of limitless bandwidth and zero round trip delay, the protocol of course will be written with that in mind.

When we add bandwidth restrictions and extend the RTT to sometimes hundreds of milliseconds, the performance will suffer badly. The SteelHead makes the WAN look like a LAN by mitigating for the RTT and the bandwidth.

It is a multi-faceted approach, involving the Network, Transport and Application layers.

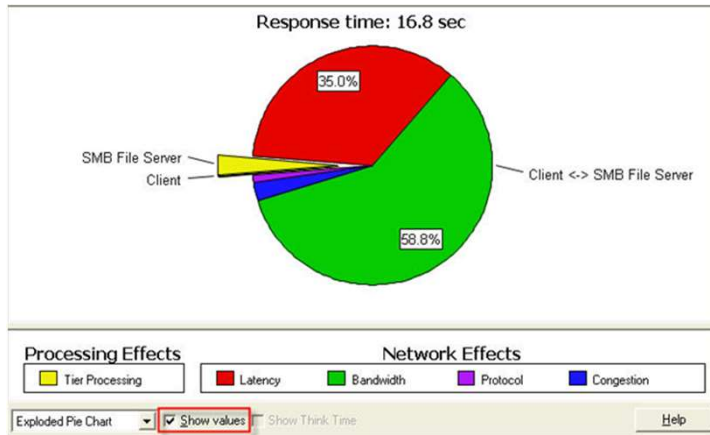
Why Optimize – Consolidating across a WAN

Consolidation to Data Centers, the Big Driver



Elements of Network Delay – Network Perspective

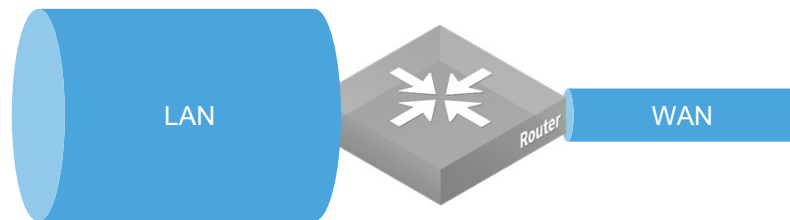
Summary of the Elements of Delay – Network Perspective



Statistics	No WAN Acceleration
Response Time (sec)	16.8
WAN Application Turns	115
Effect of Latency (sec)	5.9
Effect of Bandwidth (sec)	9.9
Effect of Protocol (sec)	0.16
Effect of Congestion (sec)	0.42

Elements of Network Delay – Transmission Time

- Clocking speed - Time to Wire
 - Bandwidth restriction
 - Packetization delay
 - Fragmentation & reassembly



Elements of Network Delay - Latency

■ Latency

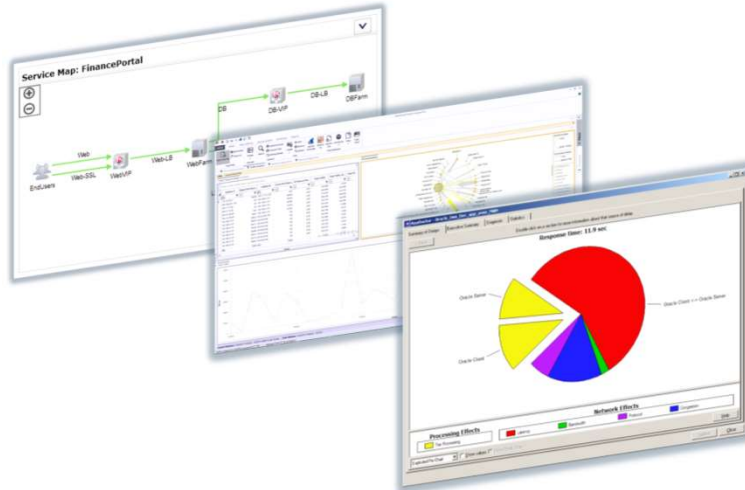
- Switching time
- Geography
- Application Turns
- Transport Turns
- Flow Control
- Window delays



Elements of Network Delay – Not network-based

Delays Other than Those on the Network

- Server delay
- Tier Processing
- External Lookups
- Client Delay
- User Delay



Increasing the Performance with Optimization

Goals

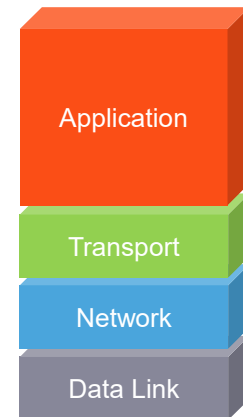
Mitigation for:

- Bandwidth restrictions
 - Deduplication
 - Compression
- Round Trip Time
 - Transport Layer
 - Application Layer
- Application Awareness
 - Deep Packet Inspection
 - QoS, Path Selection

Where Can We Make the Biggest Difference? TCP & UDP

TCP and UDP behavior

- TCP is a streaming protocol
 - Transfers large amounts of data segmented into smaller datagrams
 - Is reliable and connection orientated
 - Carries most of the traffic
- UDP is a connectionless protocol
 - Unidirectional protocol
 - Carries single messages
 - Assumes the application layer achieves reliability if it is required



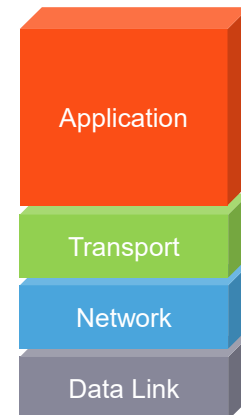
Where Can We Make the Biggest Difference? Layer-4

Layer 4 Is the Obvious Choice

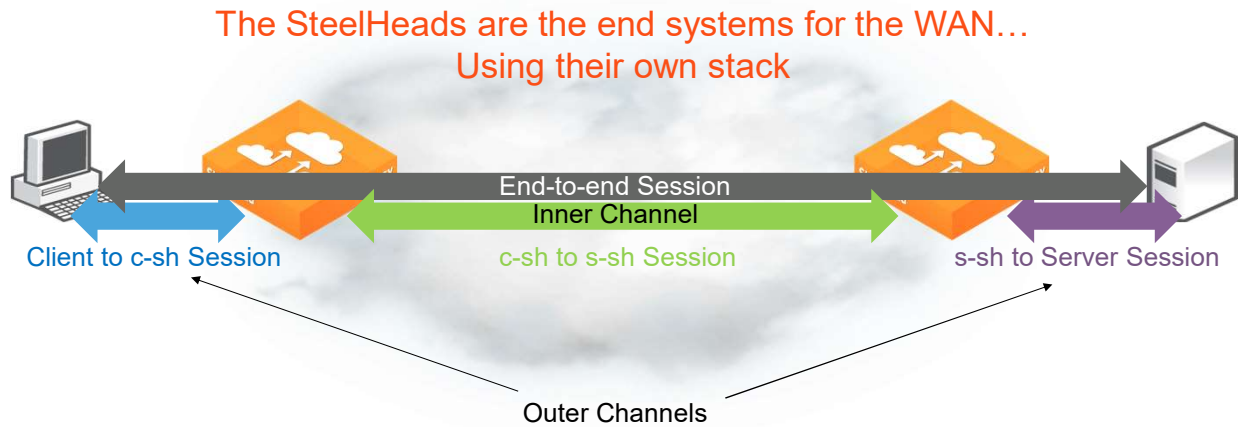
- We cannot change what we don't get involved in
 - If we terminate and regenerate Layer 4 we can use our own stack
 - If we understand the application:
 - We cannot change the RTT, but...
 - We *CAN* change the number of times the user suffers it
 - Obviously this all depends on seeing the data in the clear (no encryption)

So...

- A TCP Proxy device, aware of the major applications is the answer



Where is best to Optimize? – TCP Proxy



© 2020 Riverbed Technology, Inc. All rights reserved.

riverbed 15

Note:

c-sh = client SteelHead
s-sh = server SteelHead



SteelHead Deduplication – SDR Concept

Scalable Data Referencing (SDR)

- SteelHeads employ a sophisticated deduplication algorithm
- It detects repeatable patterns of bytes
- 60% – 90% reduction is commonplace, sometimes it is more

With SDR, a data pattern traverses the WAN
once between any two SteelHeads

SteelHead Deduplication – SDR Basics

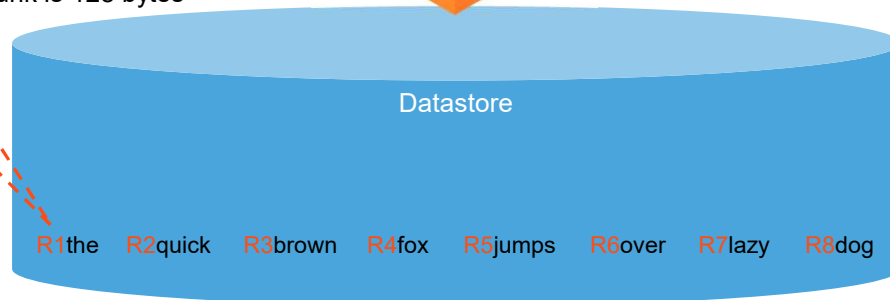
Scalable Data Referencing

- Incoming LAN data is sliced into chunks
- A reference is added to each chunk
- References are 16 bytes
- $8 < \text{chunk} < 512$ bytes
- Average chunk is 128 bytes



the
quick
brown
fox
jumps
over
the
lazy
dog

Note: "the"
was recorded
only once.



© 2020 Riverbed Technology, Inc. All rights reserved.

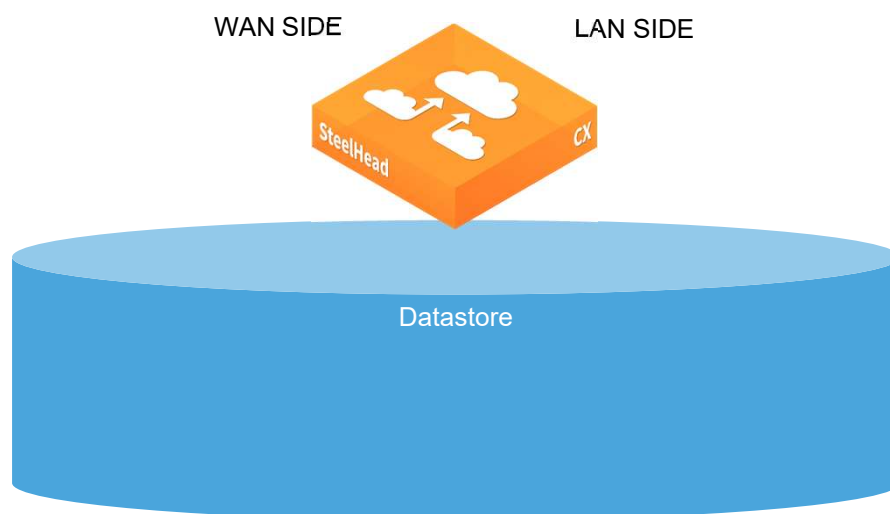
riverbed 18

References are 16 bytes. The segments themselves are between 8 and 512 bytes, on average 128. Their length depends entirely upon the way the algorithm works and we have no control over it... and even less interest. All we need to know about how it works, is, it works very well! It is worth remembering that all SteelHeads, of all revisions will do exactly the same, with exactly the same data.

Don't forget, whatever goes across the WAN as optimized is compressed using Lempel Ziv. There are 9 levels of this, for adjustment of the tradeoff between cpu resource and compression efficiency. The SteelHeads are configurable for this, but the default is 6 (out of 1-9).

SteelHead Deduplication – SDR “cold pass”

First Pass “Cold”



© 2020 Riverbed Technology, Inc. All rights reserved.

riverbed 19

In addition to traditional techniques like data compression, RiOS also uses a Riverbed proprietary algorithm called Scalable Data Referencing (SDR).

RiOS SDR breaks up TCP data streams into *unique data chunks* that are stored on the hard disks (*RiOS data store*) of the device running RiOS (a SteelHead or SteelCentral Controller for SteelHead Mobile host system).

Each data chunk is assigned a unique integer label (*reference*) before it is sent to a peer RiOS device across the WAN.

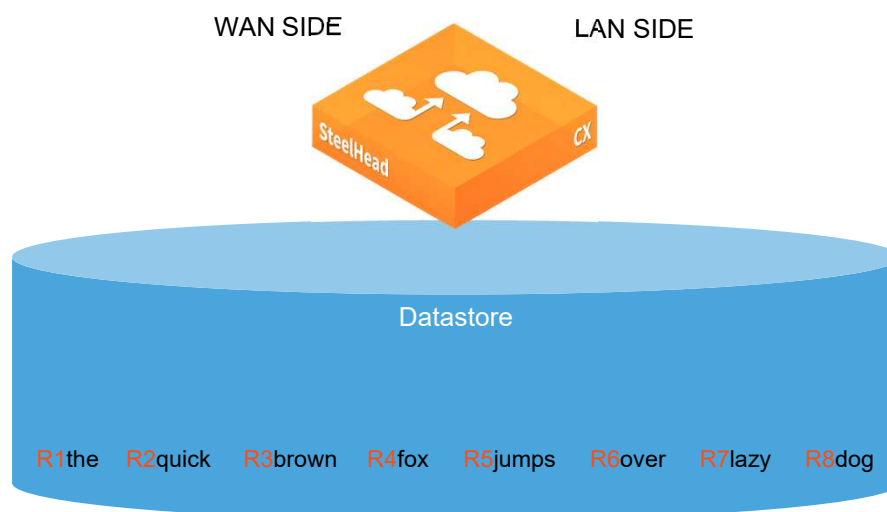
When the same byte sequence occurs in future transmissions from clients or servers, the reference is sent across the WAN instead of the raw data chunk.

The peer RiOS device (a SteelHead or SteelCentral Controller for SteelHead Mobile host system) uses this reference to find the original data chunk on its RiOS data store and reconstruct the original TCP data stream.

Files and other data structures can be accelerated by data streamlining even when they are transferred using different applications. For example, a file that is initially transferred through CIFS is accelerated when it is transferred again through FTP.

SteelHead Deduplication – SDR “warm pass”

Subsequent Passes “Warm”



© 2020 Riverbed Technology, Inc. All rights reserved.

riverbed 20

In addition to traditional techniques like data compression, RiOS also uses a Riverbed proprietary algorithm called Scalable Data Referencing (SDR).

RiOS SDR breaks up TCP data streams into *unique data chunks* that are stored on the hard disks (*RiOS data store*) of the device running RiOS (a SteelHead or SteelCentral Controller for SteelHead Mobile host system).

Each data chunk is assigned a unique integer label (*reference*) before it is sent to a peer RiOS device across the WAN.

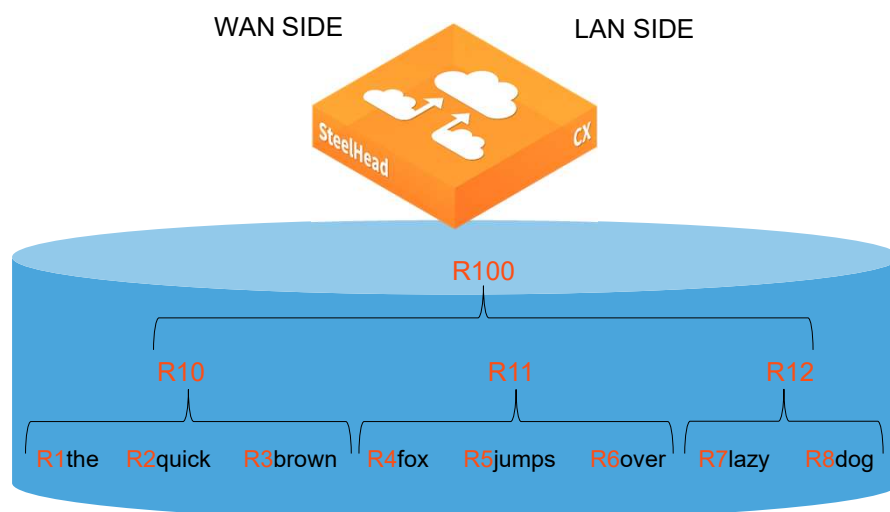
When the same byte sequence occurs in future transmissions from clients or servers, the reference is sent across the WAN instead of the raw data chunk.

The peer RiOS device (a SteelHead or SteelCentral Controller for SteelHead Mobile host system) uses this reference to find the original data chunk on its RiOS data store and reconstruct the original TCP data stream.

Files and other data structures can be accelerated by data streamlining even when they are transferred using different applications. For example, a file that is initially transferred through CIFS is accelerated when it is transferred again through FTP.

SteelHead Deduplication – Meta data, the “S” in “SDR”

It Does Not Stop There: The Meta Data



© 2020 Riverbed Technology, Inc. All rights reserved.

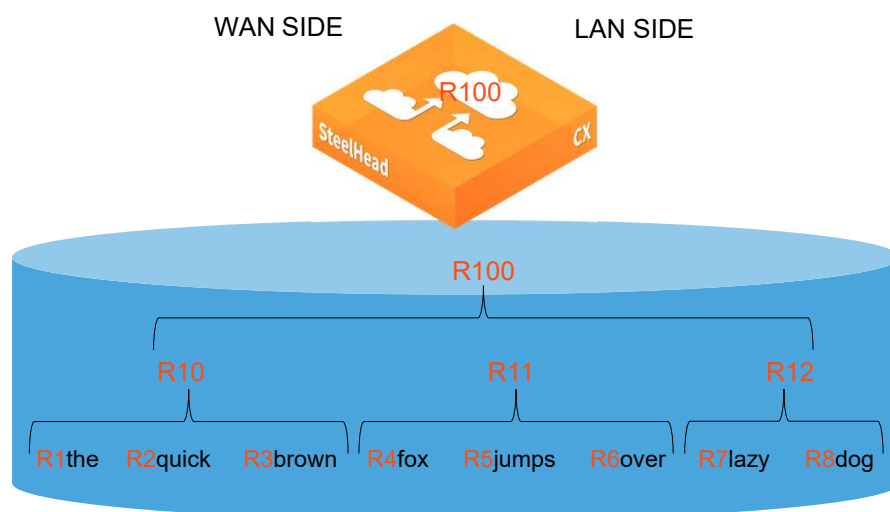
riverbed 21 kg

The various datum retrieved by users (from a server) are intercepted on the LAN interface of a Server-side SH (S-SH), initiating the SDR process.

- Implementing a randomizer engine, SDR creates 8 to 512 byte 'chunks' of unique byte patterns from the incoming data stream (or, "outer channel") and writes them to the Segstore partition (a.k.a. Datastore partition).
- SDR then assigns a unique 16-byte reference number to each byte pattern; these are 1st Level reference numbers that are linked directly (and permanently) to their respective byte patterns. In this slide, the reference numbers r1 through r12 are 1st-level 16-byte reference numbers.
- The 'scalable' aspect of SDR refers to its ability to recognize 'popular' 1st level reference numbers, for which it generates 2nd-level reference numbers to represent the aggregate of 1st-level reference numbers and their assigned byte patterns.
- This process repeats as necessary up to 3 more iterations. References r13 through r19 represent these higher-level reference numbers.
- Once this initial SDR process is complete, the S-SH then sends the byte patterns and their assigned reference numbers out the WAN interface; this is a cold transfer between the two SH's (the "inner channel").
- Should this S-SH intercept subsequent LAN-sourced data streams containing previously processed patterns of zeroes and ones (from same or similar files), SDR will NOT 'chunk' or write this same data to disk. Instead, SDR directs the S-SH to send across the WAN just the reference numbers to which the original byte patterns had been assigned – no actual byte patterns will traverse the WAN, substantially reducing WAN load. These subsequent transfers are known as a warm transfers and constitute a major component of SH Bandwidth (or Data) Streamlining.

SteelHead Deduplication – SDR in a warm pass

So What Really Goes to Line



© 2020 Riverbed Technology, Inc. All rights reserved.

riverbed 22

Applications that encode data in a different format when they transmit over the WAN can also be accelerated by data streamlining.

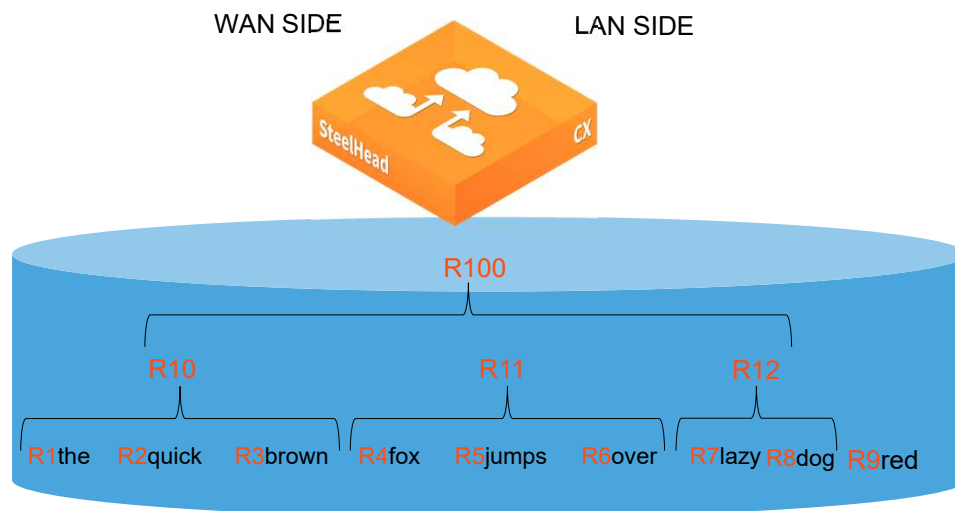
For example, Microsoft Exchange uses the MAPI protocol to encode file attachments prior to sending them to Microsoft Outlook clients.

As a part of its MAPI-specific optimized connections, the RiOS decodes the data before applying SDR.

This decoding enables the SteelHead to recognize byte sequences in file attachments in their native form when the file is subsequently transferred through FTP or copied to a CIFS file share.

SteelHead Deduplication – SDR in ‘partial warm’ pass

A Hybrid Transfer



© 2020 Riverbed Technology, Inc. All rights reserved.

riverbed 23

Applications that encode data in a different format when they transmit over the WAN can also be accelerated by data streamlining.

For example, Microsoft Exchange uses the MAPI protocol to encode file attachments prior to sending them to Microsoft Outlook clients.

As a part of its MAPI-specific optimized connections, the RiOS decodes the data before applying SDR.

This decoding enables the SteelHead to recognize byte sequences in file attachments in their native form when the file is subsequently transferred through FTP or copied to a CIFS file share.

SteelHead Deduplication – WAN as a rate limiter

Without Optimization, the WAN sets the End-to-End Speed



SteelHead Deduplication – WAN with SDR

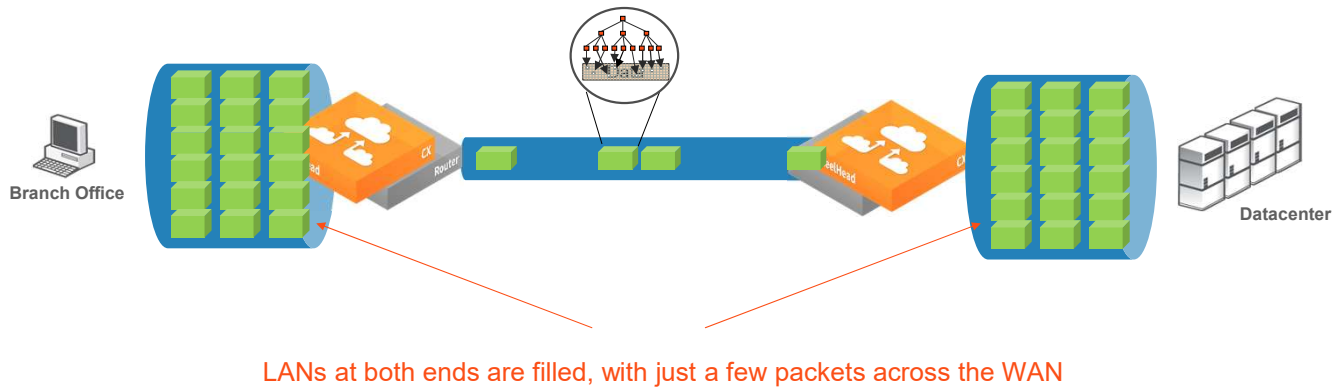
Add Optimization



SteelHead Deduplication –WAN unleashed

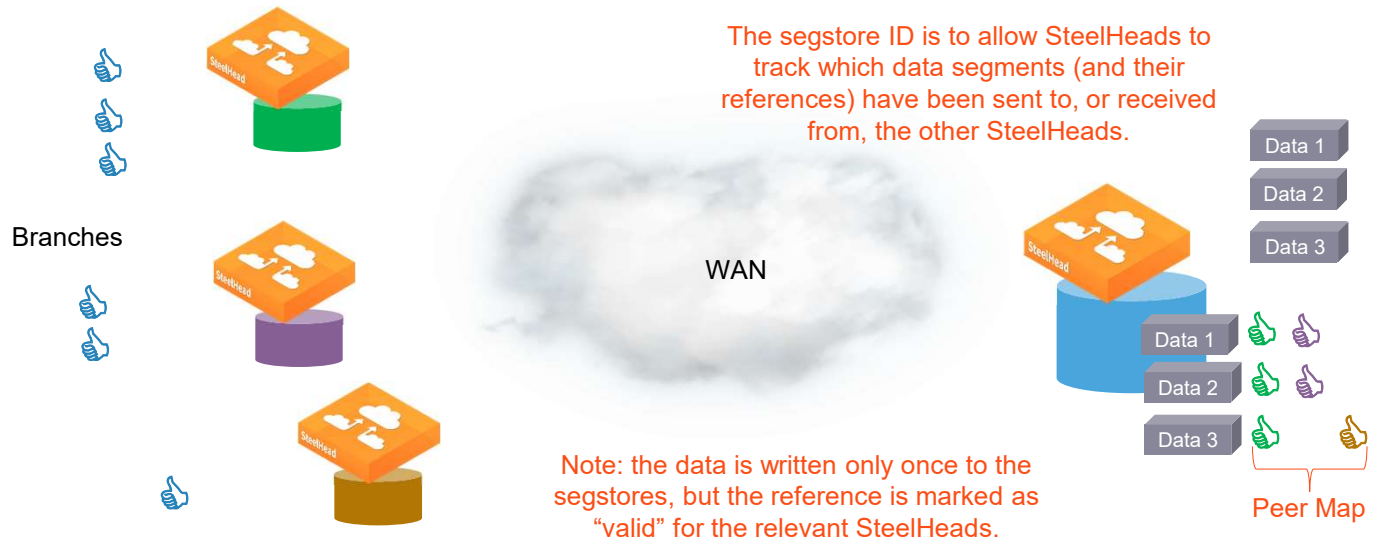
No Bottleneck

Commonly 60-98% reduction in bandwidth



The Data Store Identifier – a Unique Segstore ID

What is the Data Store ID (or Segstore ID)? What is it For?

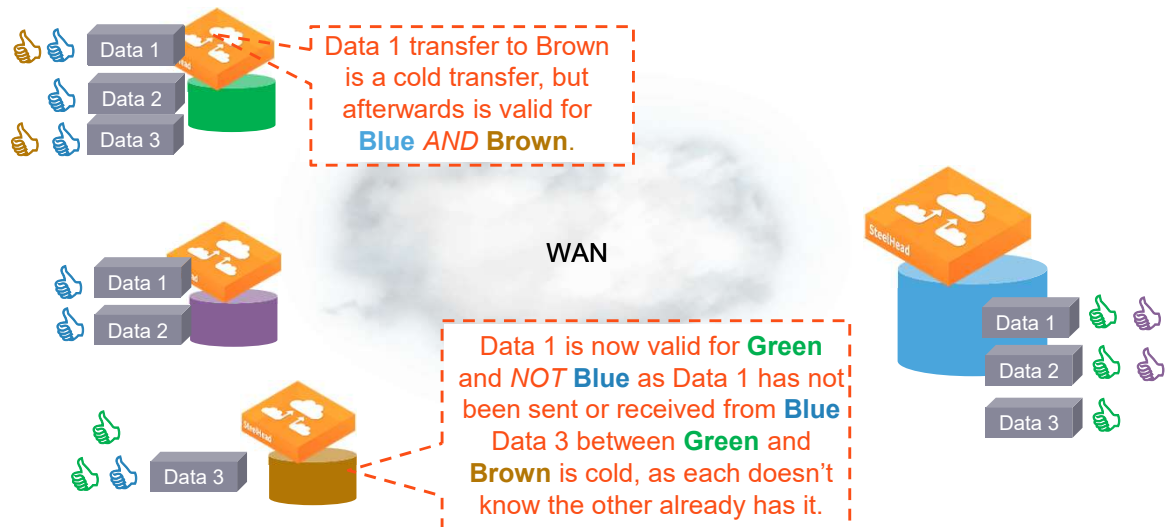


© 2020 Riverbed Technology, Inc. All rights reserved.

riverbed 27

The Data Store Identifier – Usage & considerations

Now from Branch to Branch



Automated Data Store Synchronization – the Why What it is and How it Works

- Used to increase the transparency to users of a failover
- SDR data is bi-directionally shared between *exactly two* SteelHeads
- If one or the other of the pair fails ... the same warm performance is achieved following the users reset
- Communication and synchronization occurs on the Primary or Auxiliary interfaces (best practice Aux)
- They both assume the same segstore id. (the one of the master)

Automated Data Store Synchronization – Segstore HA

High Availability for Your Warm Data



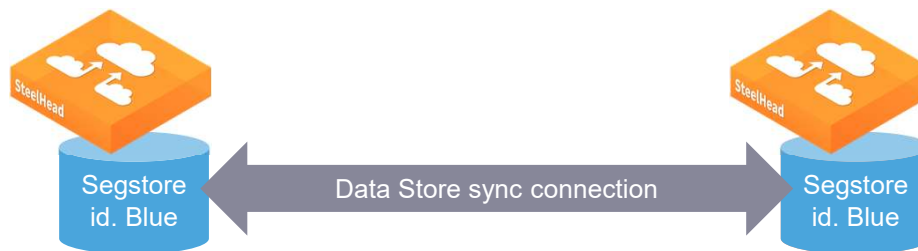
© 2020 Riverbed Technology, Inc. All rights reserved.

riverbed 30

Automated Data Store Synchronization – Details

The Master & Backup Devices Data Store Identifiers

- Upon connection, the Backup assumes the segstore id of the Master, which means:
 - The backup segstore id is never seen again
 - All other SteelHeads see only one segstore id, Blue; therefore ...
 - References are valid on *ALL* remote SteelHeads for both Master & Backup
 - Synchronization is bi-directional



© 2020 Riverbed Technology, Inc. All rights reserved.

riverbed 31

Automated Data Store Synchronization – Consideration

The Backup Devices Segstore ID

- If they ever try to optimize between them we will get errors.
- Always ensure it never happens with peering rules for all in-path interfaces.

Syslog: "Peer sport id is the same as mine!"

Peering Rules network Services > Peering Rules

Peering rules allow you to define appliance peering relationships. Note that only the first matching rule will be applied.

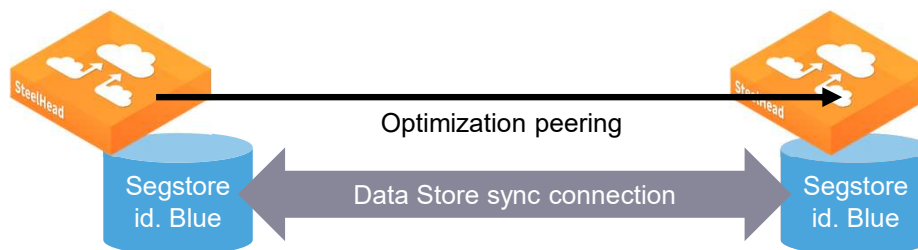
Settings

- ☒ Enable Enhanced IPsec Auto-Discovery
- ☒ Enable Enhanced IPsec Auto-Discovery
- ☒ Enable Forwarded Peer Table

Apply

Add a New Peering Rule Remove Selected Rules (31 More Selected Rules...)

Number	Type	Source	Destination	Port	Peer	SSL	Cloud Acceleration
1	Pass	All-IP	All-IP	All	All-IPv4	Incapable	Auto
Description: Default rule to pass-through connections destined to currently bypassed SSL client server pairs							
2	Auto	All-IP	All-IP	443	All-IPv4	Capable	Auto
Description: Default rule to auto-discover and attempt to optimize connections destined to port 443 as SSL							
default	Auto	All-IP	All-IP	All	All-IPv4	No Check	Auto



Automated Data Store Synchronization – Configure Configuration

Remember:

- The same version of code is needed on both SteelHeads
- Both Steelheads must be the same model
- This is the only way to preserve a Data Store



RD	NETWORKING	OPTIMIZATION	REPORTS	AD
	NETWORK SERVICES	DATA REPLICATION	SSL	
	General Service Settings	FCIP	SSL Main Settings	
	In-Path Rules	SRDF	Secure Peering (SSL)	
	Peering Rules	SnapMirror	Certificate Authorities	
	Transport Settings	Data Store	CRL Management	
	Service Ports	Performance	Advanced Settings	
			Secure Peering IPsec	

Optimization
Data Store


By the way, encrypting the Data Store can affect performance and in most cases is not necessary.

Data Store Data Replication > Data Store ?

General Settings

Data Store Encryption Type: None  

☒ Enable Automated Data Store Synchronization

Current Appliance: Backup 

Peer IP Address:

Synchronization Port:

Reconnection Interval (seconds):

☒ Enable Branch Warming for SteelHead Mobile Clients

☐ Enable Data Store Wrap Notifications

Threshold: days

Apply

Related Topics: [Secure Vault](#), [Data Store Status](#)

Automated Data Store Synchronization – RMA use case

Replacing a SteelHead *Without* Losing the Data Store

- The SteelHead configured as Master is the segstore id that all others know
- To replace a backup, just do it, nothing else to do with DSS
- To replace a Master we need to be careful:
 - Change the old Backup to be the new Master (this retains the segstore id)
 - Replace the old master with a new one configured as Backup
 - Wait for it to sync up - it could be awhile as potentially it is TB of data
 - Optimization will be working straight away

1. For more details on Data Store synchronization, see S12964 and S17927 on the support Knowledge base

Automated Data Store Synchronization – Caveats

Some Caveats

- The Steelheads must be the same model and the same code.
- Any topology is supported (in-path or otherwise) but:
 - They should be LAN speed and RTT away from each other (max supported is 10ms).
- They must be prevented from *ever* optimizing to each other (peering rules).
- Effectively there is only one Data Store for the pair.
- During an outage of one, e.g., upgrade, everything still works except DSS. It will catch up (re-sync) when DSS connects again.



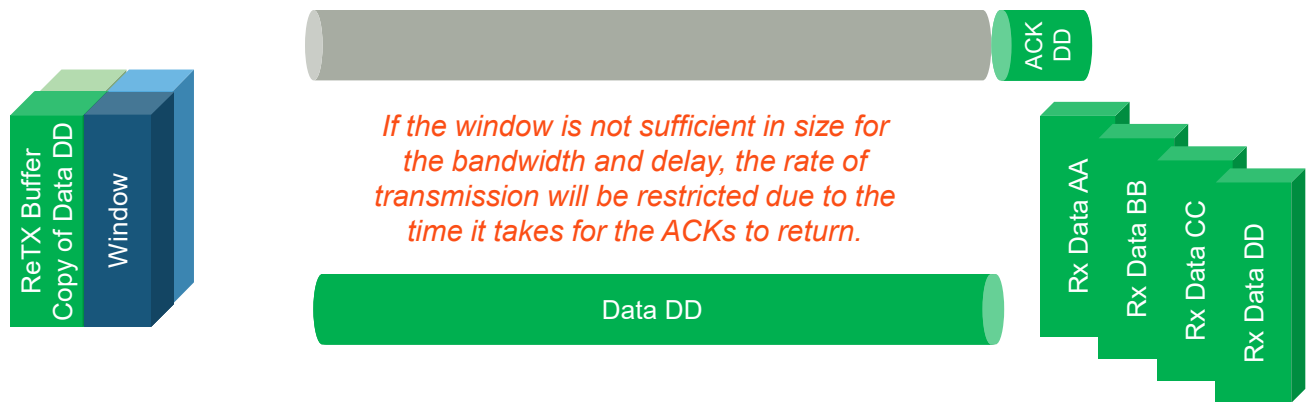
Sliding Windows

Key Concepts

- The **Tx Window** represents the number of bytes the transmitter is allowed to have in-flight (unacknowledged).
- The **Rx Window** (known as the advertised window) describes the number of bytes that the receiver is capable of accepting at any point in time.
- Used for:
 - Flow control
 - Congestion control
 - Delivery in order

Window Scaling – Overview and limitation

Size is Important

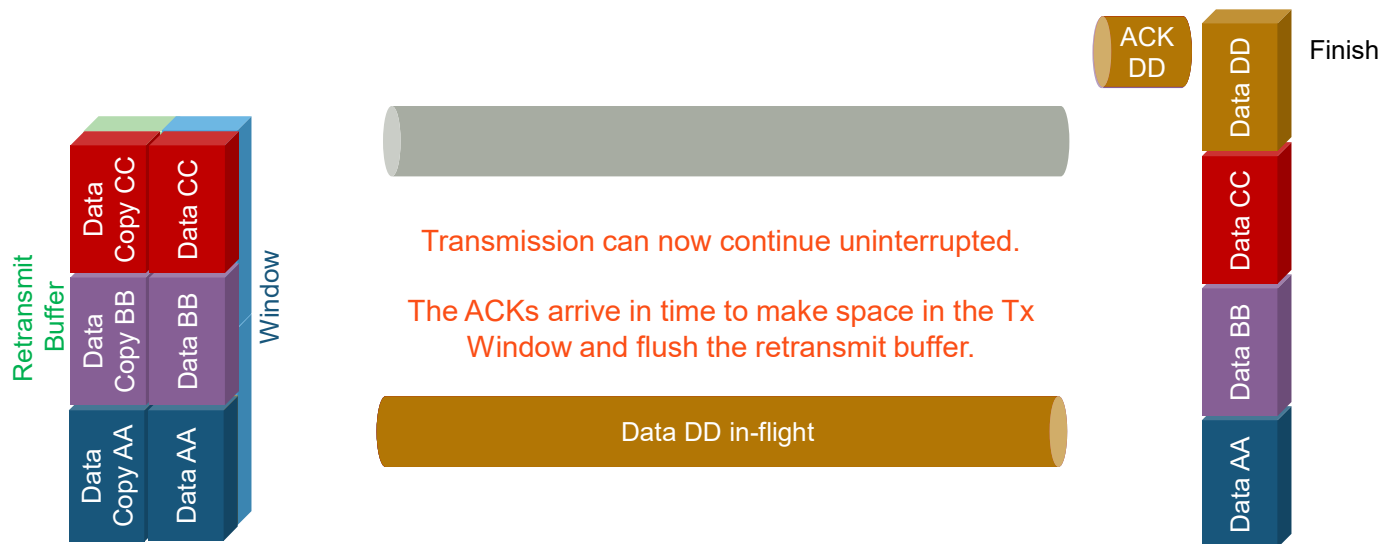


© 2020 Riverbed Technology, Inc. All rights reserved.

riverbed 39

If the time taken for the ACK to return from the other end is greater than the time it takes to send the contents of the retransmit buffer (the Tx Window), then the transmission **MUST** stop until it hears that it got there safely. As such the transmission will be interrupted.

Window Scaling – When window size matches need With Sufficient Window



© 2020 Riverbed Technology, Inc. All rights reserved.

riverbed 40

Window Scaling – RFC1323 basics

Described in RFC 1323

- The Window field in TCP allows for 2^{16} which equates to 64KB
- Counted in segments, so MSS is of interest here
- Window scaling allows this to be increased subject to negotiation
- Negotiation is only carried out in SYN and SYN/ACK messages
- Renegotiation “on the fly” is not possible

tcp.options.wscale.multiplier						
No.	Time	HTTP code	Source	Destination	Protocol	Length
241	6...		10.1.3.20	10.1.30.252	TCP	74
242	6...		10.1.30.252	10.1.3.20	TCP	74
286	7...		10.1.3.20	10.1.30.252	TCP	90
295	8...		10.1.3.25	10.1.30.25	TCP	74
296	8...		10.1.30.25	10.1.3.25	TCP	74
297	8...		10.1.3.25	10.1.30.25	TCP	74
298	8...		10.1.30.25	10.1.3.25	TCP	74
307	8...		10.1.3.25	10.1.30.25	TCP	74
308	8...		10.1.30.25	10.1.3.25	TCP	74
310	8...		10.1.3.25	10.1.30.25	TCP	74
311	8...		10.1.30.25	10.1.3.25	TCP	74
319	8...		10.1.3.25	10.1.30.25	TCP	74
320	8...		10.1.30.25	10.1.3.25	TCP	74
334	8...		10.1.3.25	10.1.30.25	TCP	74
335	8...		10.1.30.25	10.1.3.25	TCP	74

[Checksum Status: Unverified]						
Urgent pointer: 0						
Options: (20 bytes), Maximum segment size, SACK permitted, Timestamps, No-Operation (NOP), Window scale						
> Maximum segment size: 1460 bytes						
> TCP SACK Permitted Option: True						
> Timestamps: TSval 4134855, TSecr 163410						
> No-Operation (NOP)						
> Window Scale: 3 (multiply by 64)						
Kind: Window Scale (3)						
Length: 3						
Shift count: 6						
[Multiplier: 64]						
> [SEQ/ACK analysis]						

0000	a0 b1 d4 89 88 77 00 0c	29 f1 ec 46 08 00 45 00W..J..F..E.
0010	00 3c 00 00 40 00 40 06	04 ab 0a 01 1e fc 0a 01	<..@.0.
0020	03 14 01 b0 4b ec 99 6c	b2 b9 00 cf 08 c3 a0 12	...K..l
0030	16 a0 c1 84 00 00 02 04	05 b4 04 02 08 0a 00 3f7
0040	17 c7 00 02 7e 52 01 03	03 06~R...

Window Scaling – Sizing calculation

How Much is Enough? Here is How we Calculate it.

- The buffer size required to fill any link is calculated by multiplying the bandwidth by the delay, which is known as the Bandwidth Delay Product (BDP).
- Example:
 - T1 Service
 - 50ms one way latency or 100ms RTT

$$1.536 \text{ Mbps} = 192\text{KB/s}$$

$$(192 \times 10^3 \cancel{\text{bytes/second}}) \times (100 \times 10^{-3} \cancel{\text{seconds}}) = 19,200 \text{ bytes}$$

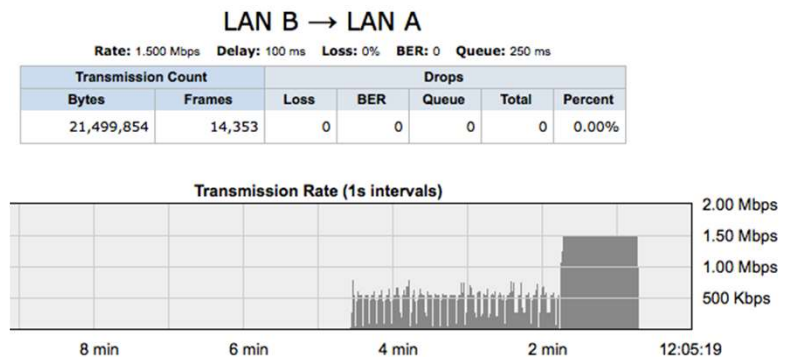
Don't forget the Maximum Segment Size (MSS) is typically 1460.
Therefore, rounding up to 14 x MSS gives us a buffer size of 20KB (20480 bytes).

Window Scaling – traffic example transfer graph

Example of Traffic Patterns Before and After Sufficient Window

- T1 with 200ms RTT Win8 to Server 2008R2 (No Optimization)

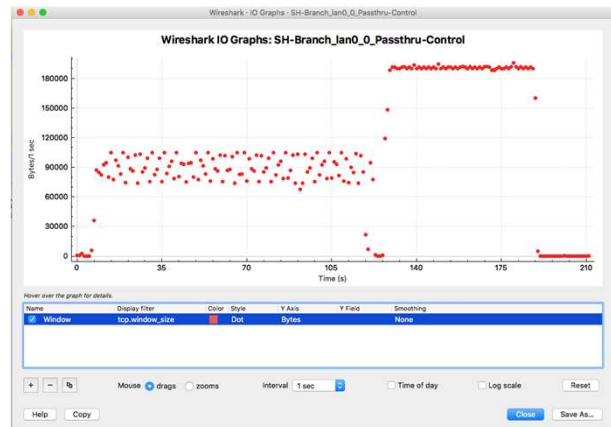
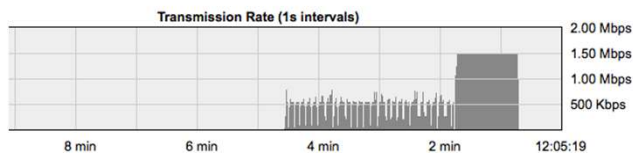
This looks like TCP windowing of some sort. The challenge is to know the cause of the behavior...



Window Scaling – traffic example window size detail

Example of Traffic Patterns Before and After Sufficient Window

Same transfer, with a Wireshark view of the advertised window



TCP Nagle Algorithm

- General rules:
 - If there is unacknowledged data in-flight, new data is buffered
 - If the data to be sent is <MSS, it is buffered until MSS is reached
- More specifically:
 - Send only if one of the conditions below is met...
 - MSS is reached
 - All previously sent data has been acknowledged AND Push flag is set OR buffered data >½ send window
(This last part is recommended but is implementation dependent)
 - Push flag is set AND the Nagle timer has matured

From RFC1122:

Generally, an interactive application protocol must set the PUSH flag at least in the last SEND call in each command or response sequence. A bulk transfer protocol like FTP should set the PUSH flag on the last segment of a file or when necessary to prevent buffer deadlock.

For additional information: <https://www.youtube.com/watch?v=adDC5T-RzR4>

© 2020 Riverbed Technology, Inc. All rights reserved.

riverbed 45

The whole idea of the Nagle algorithm is to avoid many small segments by combining them. Therefore, if the MSS has already been reached, Nagle does not apply and the data can be sent. Otherwise, only if all data has been acknowledged so none is in flight AND the push flag is set, or the data represents over half the buffer size, it can be sent. The push flag here is an excerpt from RFC793:

The sending user indicates in each SEND call whether the data in that call (and any preceding calls) should be immediately pushed through to the receiving user by the setting of the PUSH flag. A sending TCP is allowed to collect data from the sending user and to send that data in segments at its own convenience, until the push function is signaled, then it must send all unsent data.

RFC793 states: “When a receiving TCP sees the PUSH flag, it must not wait for more data from the sending TCP before passing the data to the receiving process”. This is now OPTIONAL as specified in RFC1122.

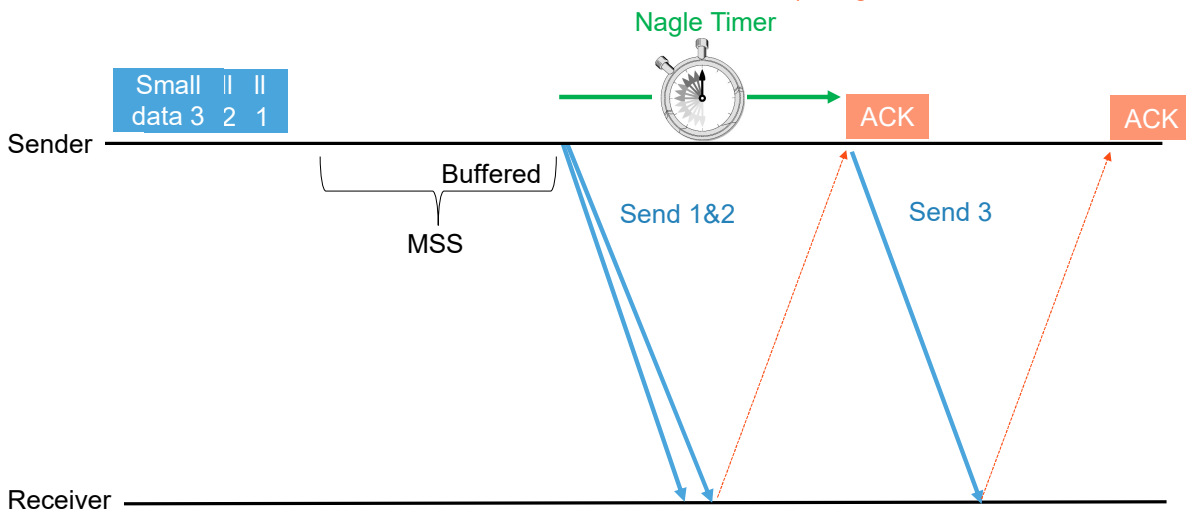
Timeouts range from a few milliseconds to 200 or 300. The SteelHead has a default of 6ms.

TCP Nagle – example transfer

Neural Framing Mode in Operation

In this case, the last data will be sent only when:

- a) ACK Rx for all sent & Push bit set
- b) Nagle timer matures & Push bit set



© 2020 Riverbed Technology, Inc. All rights reserved.

riverbed 46

From RFC 1122:

An application program is logically required to set the PUSH flag in a SEND call whenever it needs to force delivery of the data to avoid a communication deadlock. However, a TCP SHOULD send a maximum-sized segment whenever possible to improve performance.

Nagle on the SteelHeads – Considerations

Nagle setting and considerations

- Enabled by default, as vast majority of applications optimize well with Nagle.
- Set as a per-flow option in SteelHead “In-path Rules”
 - Options are: Never, Always, TCP Hints, Dynamic
 - Much more on in-path rules later
- Interactive applications do not perform well using Nagle, for example:
 - Microsoft RDP
 - FC/IP
 - SQL applications
 - Applications running small, time-sensitive packets
- Citrix blade automatically disables it; no need for additional in-path rule

© 2020 Riverbed Technology, Inc. All rights reserved.

riverbed 48

Enables neural framing in the SteelHead. Enabling neural framing makes your WAN more efficient by gathering data to select the optimal packet framing boundaries for SDR.

If you specify a neural mode, your network will experience a trade-off between the compression and SDR performance, and the latency added to the connection. For different types of traffic, one algorithm might be better than others.

Specify one of the following modes:

Never - Never uses the Nagle algorithm. All the data is immediately encoded without waiting for timers to fire or application buffers to fill past a specified threshold. Neural heuristics are computed in this mode but are not used.

Always - Always uses the Nagle algorithm. This is the default setting (always wait 6ms). All data is passed to the codec, which attempts to coalesce consume calls (if needed) to achieve better fingerprinting. A timer (6ms) backs it up and causes leftover data to be consumed. Neural heuristics are computed in this mode but are not used. This mode is not compatible with IPv6.

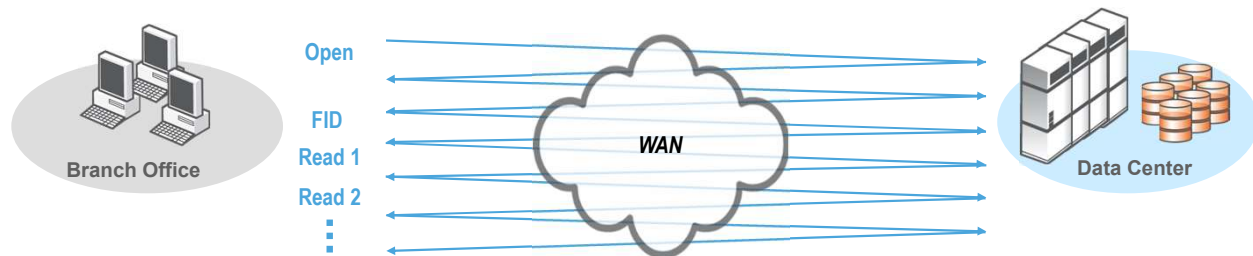
TCP Hints - Bases the setting on TCP hints. If data is received from a partial frame packet or a packet with the TCP PUSH flag set, the encoder encodes the data instead of immediately coalescing it. Neural heuristics are computed in this mode but are not used. This mode is not compatible with IPv6.

Dynamic - Dynamically adjusts the Nagle parameters. The SteelHead picks the best algorithm to use by learning what algorithm is best and adapting if the traffic characteristic changes. This mode is not compatible with IPv6.



Application Streamlining – Applications over WAN

Application Protocol Limitations



*Typical chatty application behavior;
Performance inversely proportional to intervening latency...*

© 2020 Riverbed Technology, Inc. All rights reserved.

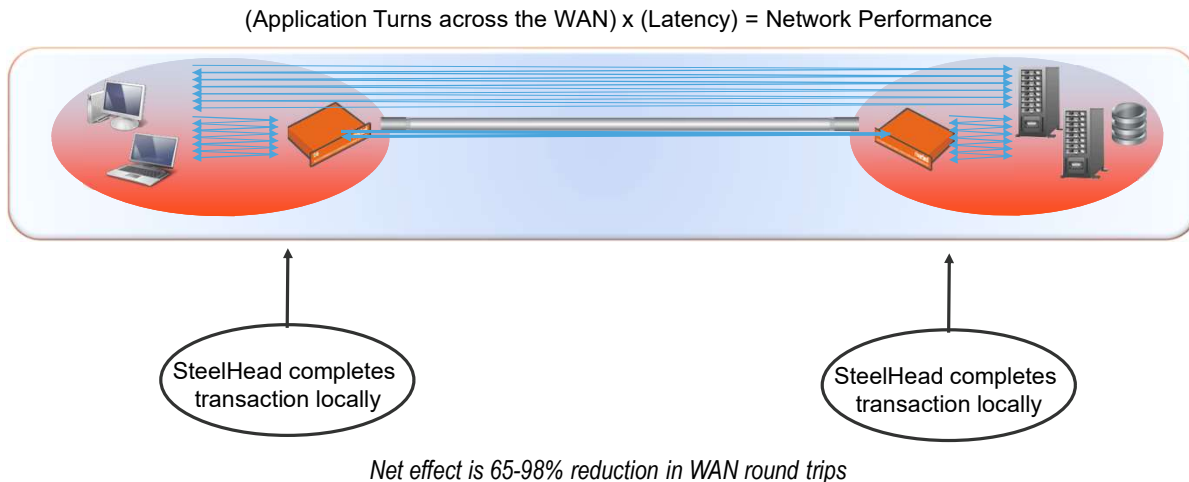
riverbed 50

This really works together with the previous solution to further speed up the transfer of application layer protocols.

The problem with application-layer protocols is that they have a 'conversation' across the WAN. Each request-response cycle is known as a 'turn' and the more turns required to complete a transaction, the longer it will take to complete. For example, 10 turns across a 100ms link will add a second of delay to the transaction.

Application Streamlining – ‘Localizing’ app transactions

Also Known As “Transaction Prediction”



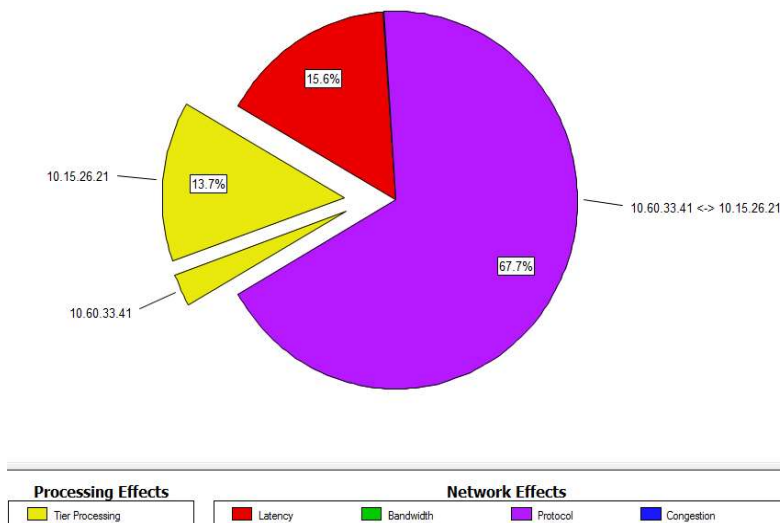
© 2020 Riverbed Technology, Inc. All rights reserved.

riverbed 51

Now, the SteelHeads understand a number of application-layer protocols which means that they can terminate them locally, act as the client/server, depending on location and restrict the chatty WAN traffic to the LAN, with a massive reduction in WAN round trips as shown.

Application Streamlining – Use Case

Opening a SharePoint object using HTTP from Hong Kong to London



© 2020 Riverbed Technology, Inc. All rights reserved.

riverbed 52

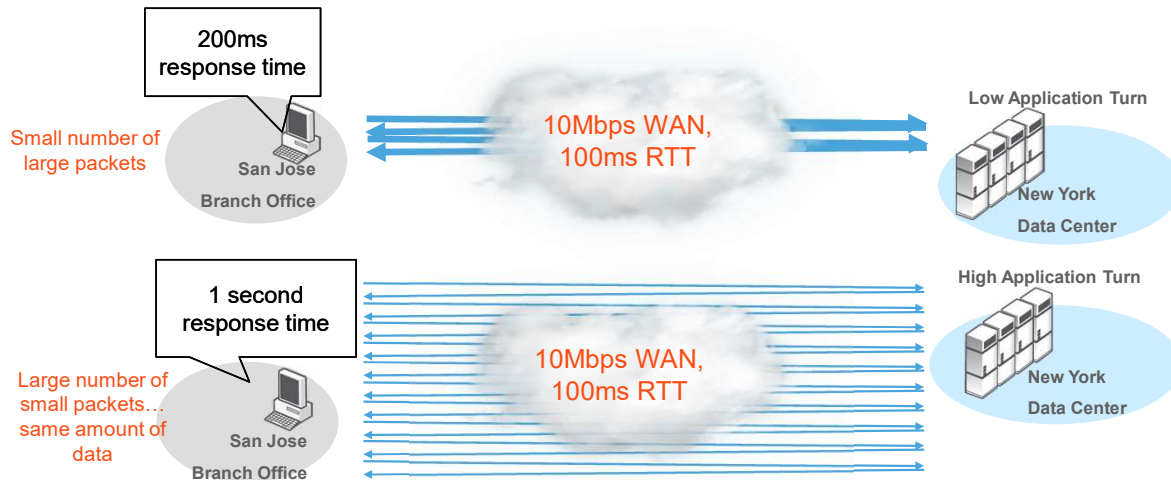
Transaction Analyzer, another one of Riverbed's products is a very useful tool for analyzing the delays of individual transactions. The above is an example of opening a SharePoint object (HTTP) from Hong Kong to Europe.

We can see that the protocol (layer 7) and latency (layer 4) make up more than 80% of the delay, causing each application transaction to take 14 seconds. With SteelHead optimization, this time was brought down to 6 seconds, a saving of 8 seconds - more than a half and nearly all down to latency! Now, that particular site had 278 users. Imagine 10 transactions a day by each user, that's 2780 transactions with 2780 * 8 seconds lost, or 6 hours of lost productive time!!! And that is just for one site.

Effect of Latency on Traffic – Application efficiency

Application Turns

Every turn suffers the network RTT chatty apps cost time on a WAN



© 2020 Riverbed Technology, Inc. All rights reserved.

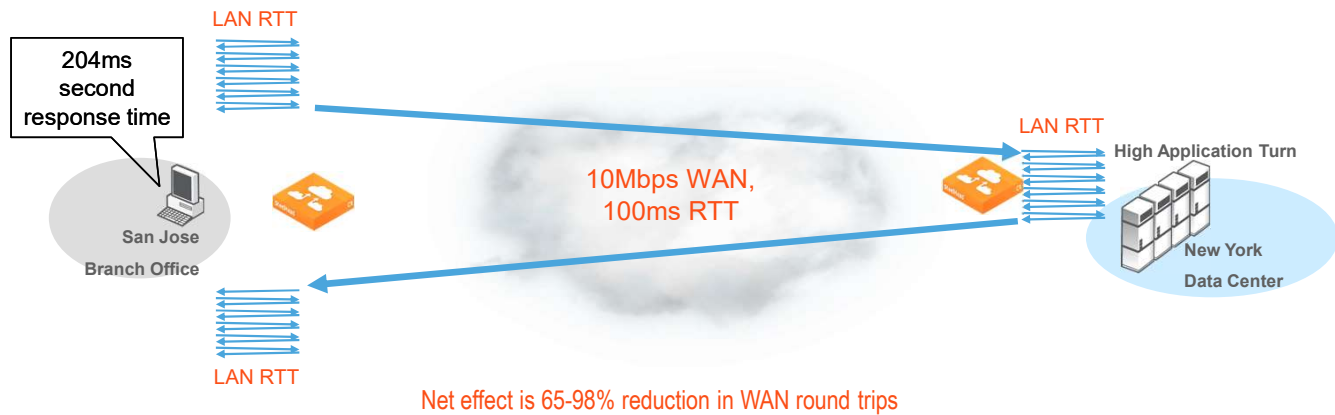
riverbed 53

Each application turn adds a delay. For example, on a 100ms link, if a transaction requires 10 turns to complete then we have added 1 sec of delay, over and above any delay associated with bandwidth. The fewer the numbers of turns the faster the application will perform.

Effect of Latency on Traffic – ‘Localization’ benefit

Application Turns

SteelHeads can complete the transaction locally – chattiness stays LAN side



© 2020 Riverbed Technology, Inc. All rights reserved.

riverbed 54

Each application turn adds a delay. For example, on a 100ms link, if a transaction requires 10 turns to complete then we have added 1 sec of delay, over and above any delay associated with bandwidth. The fewer the numbers of turns the faster the application will perform.



Key Points



Optimization changes many things; it generally makes everything work harder.



Bandwidth usage is insignificant for warm passes, but can *increase* the throughput for cold ones.



We cannot change the Round Trip Time (RTT), but we *can* change the number of times the user suffers it.

The Effect of Optimization

SteelHeads Make Everything Work Harder - even the WAN

- LAN and WAN traffic characteristics differ enormously with WAN-OP
- Cold transfers increase dramatically in throughput due to:
 - Application layer RTT optimization
 - TCP adjustments
 - Compression
 - SDR “on the fly” which works to de-duplicate even in a single transfer!
- Warm transfers increase the LAN traffic dramatically
- The extra workload can make other devices suffer, such as:
 - Servers
 - Load balancers
 - Firewalls
 - Web proxies

Visualize the Effects – SteelHead perspective

SteelHead View

CT	Notes	Source:Port	Destination:Port	LAN kB	WAN kB	Reduction	Start Time	Application
		10.1.1.51:49310	10.1.30.102:445	20,240	2,144	89%	2018/02/22 12:11:50	SMB2-SIGNED

- Do not rely entirely on the reduction percentage
- Always look at the contributing numbers as well
 - 100% of nothing is still nothing
- The arrow icon expands the view
 - This is still not the whole story because RTT acceleration and TCP adjustments are not shown here

CT	Notes	Source:Port	Destination:Port	LAN kB	WAN kB	Reduction	Start Time	Application
		10.1.1.51:49310	10.1.30.102:445	60,636	2,151	96%	2018/02/22 12:11:50	SMB2-SIGNED

Connection type: BIOS Connection age: 2 minutes, 1 second Application: SMB2-SIGNED Transport: TRANSPORT_ID_NONE Notes: In-path SDR optimized LZ compressed Client side	Source: 10.1.1.51:49310 10.1.1.25:7801 SteelHead 250 9.1.3 (YOU ARE HERE) 10.1.1.25:11085 Transparency mode: Correct Addressing Congestion control: New Reno 10.1.30.25:7800 SteelHead 550 9.1.0 Destination: 10.1.30.102:445
---	---

	LAN side	WAN side
Bytes	62,091,173	2,202,656
Packets	43,343	1,393
Retransmitted	0	1
Fast retransmitted	0	0
Timeouts	0	1
Congestion window	134	23

© 2020 Riverbed Technology, Inc. All rights reserved.

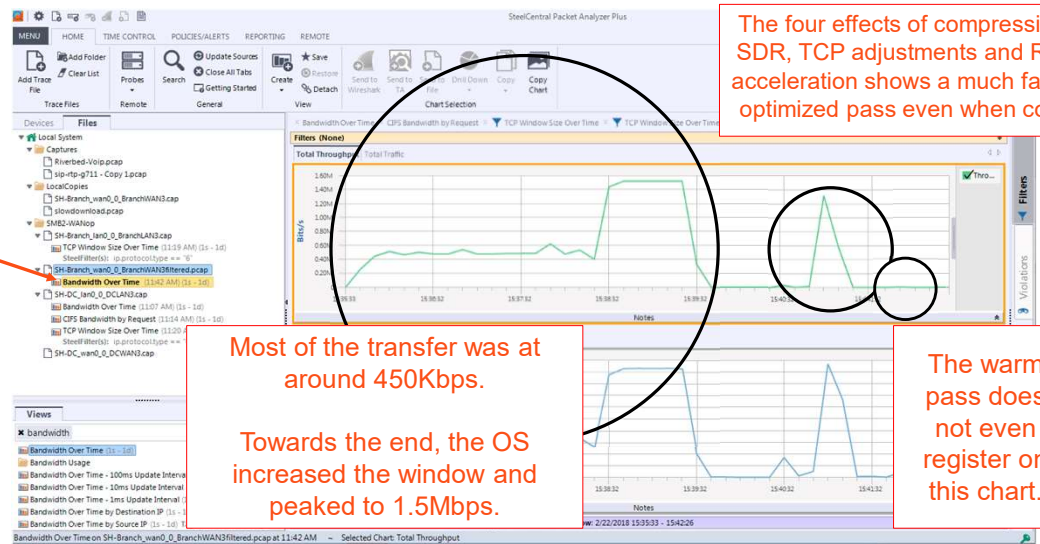
riverbed 58

Visualize the Effects – WAN view of packet behavior

Bandwidth over the WAN – The Same File Transfer, Three Times

1. No WAN-OP
2. Cold
3. Warm

Bandwidth Over Time measured on the WAN.



© 2020 Riverbed Technology, Inc. All rights reserved.

riverbed 59

Visualize the Effects – LAN view of packet behavior

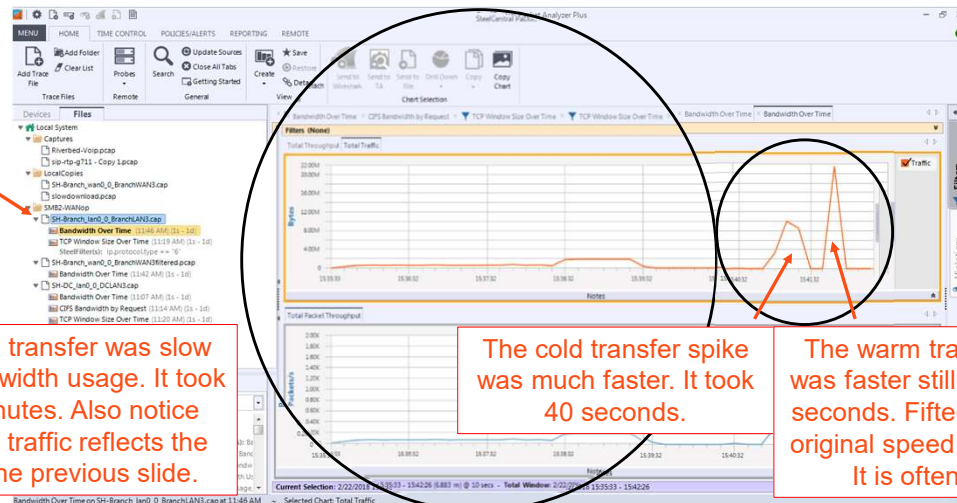
Bandwidth Over the LAN – *Where the User Experience is Measured!*

Bandwidth Over Time measured on the LAN.

Pass through transfer was slow with little bandwidth usage. It took almost 5 minutes. Also notice how the LAN traffic reflects the WAN from the previous slide.

The cold transfer spike was much faster. It took 40 seconds.

The warm transfer spike was faster still and took 20 seconds. Fifteen times the original speed in this case. It is often faster.



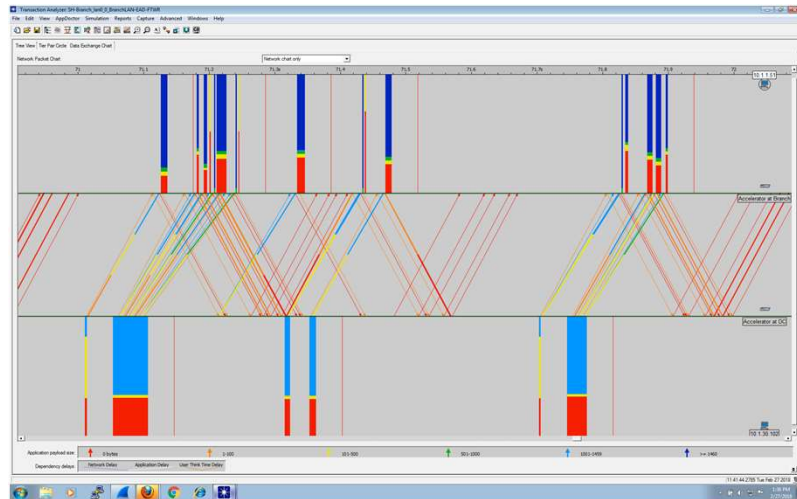
Visualize the Effects – Detailed end-to-end view

SteelCentral Transaction Analyzer View (Warm Transfer)

This shows the difference in the density of the traffic on the three sessions. Firstly from a point of view of the number of messages (App layer Acceleration). Secondly the bandwidth in use (SDR).

Also note the packet size on the server-side LAN at the bottom, compared with that of the client at the top. The color key is below.

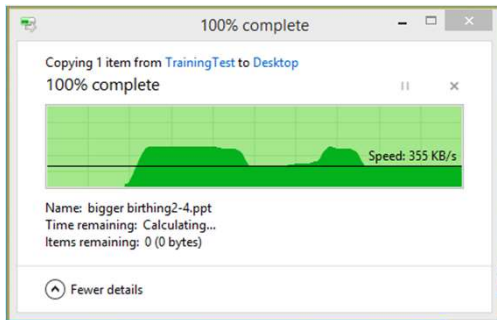
0 Bytes
1-100 Bytes
101-500 Bytes
501-1000 Bytes
1001-1499 Bytes
>=1460 Bytes



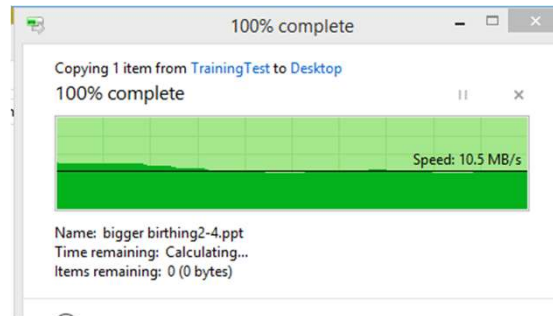
Visualize the Effects – End user view

Transfer Over the WAN – A Windows View

No Optimization

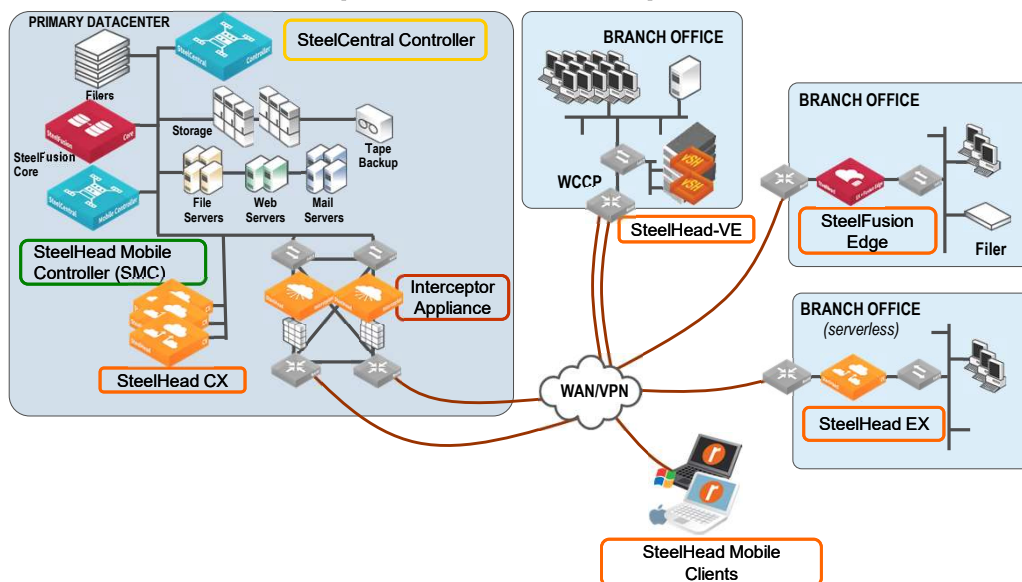


SteelHeaded!





Elements of a Complete WAN Optimization Solution



© 2020 Riverbed Technology, Inc. All rights reserved.

riverbed 64

A complete optimization solution brings more to the table than solely WAN optimization appliances.

It also includes different manifestations, or models, of the appliances to fit a variety of network topology & infrastructure needs.

The appliances should have a comprehensive management platform and ideally should include interactivity with a mobile solution deployed directly on laptops and/or workstations. Riverbed brings all this to bear and more.

HOL1131

Demonstrate WAN Optimization

In this lab, you will perform the:

- Optimization with CIFS (SMBv1) Demonstration

Duration: **45 minutes**



*eLab system: link and access details
provided in your course confirmation email*

Module Review

You should now be able to:

- Describe how optimization addresses network performance challenges.
- Alleviate bandwidth restrictions.
- Increase TCP performance.
- Mitigate RTT at the application layer.
- Visualize the effects of optimization.
- Describe elements of a complete WAN optimization solution.

© 2020 Riverbed Technology, Inc. All rights reserved.

riverbed
The Digital Performance Company