Configure TCP Optimizations

For supported software information, click here.

TCP optimizations mitigate the effects of high latency and packet loss on the performance of TCP-based applications. The optimizations are based on a TCP proxy architecture, in which one or more Versa Operating System^{TM} (VOS^{TM}) devices in a network path between a client and server split the TCP connection into two. One connection faces the client and the other one faces the server, and the VOS devices act as TCP proxies for each of the split network segments. The optimizations can be done in dual-ended mode, in which the TCP connection is split between two peer VOS devices in the network path that discover each other, or in single-ended mode, in which one or more VOS devices can independently have single-ended TCP optimization configured without being aware of or coordinating with the other VOS devices in the network path.

TCP Optimization Overview

TCP is a connection-oriented protocol with traffic management capabilities that focus on reliably delivering traffic to its destination. However, a number of factors can negatively impact the performance of TCP connections, including:

- Latency—Latency is the delay between the time that a device sends TCP packets and the time that it receives an acknowledgment that the packets were received. Latency increases as network distances increase. When latency is high, the sending device spends more time idling, which reduces throughput and translates to a poor application experience.
- TCP slow-start—When a device begins sending TCP traffic, TCP slow-start probes network bandwidth and gradually increases the amount of data transmitted until it finds the network's maximum carrying capacity. On a high-latency path, it can take time to determine the maximum carrying capacity and to react to changes in bandwidth. TCP optimization reduces the time to reach the maximum available bandwidth.
- Last-mile network problems—Last-mile networks can suffer from a number of problems, such as random packet loss and buffer bloat. For example, an LTE network may experience random packet loss because of poor signal coverage or access-point handovers. In addition, the large shaping buffers that are often present in last-mile networks can fill up, causing correspondingly large end-to-end latencies.
- Packet reordering—Packet reordering can cause a TCP sender to mistakenly detect packet loss and retransmit data needlessly.
- Burst losses—Shallow buffers and packet-transmission strategies on wireless networks can cause burst losses, in which the idle time between bursts of traffic can be misinterpreted as signs of congestion, causing unnecessary TCP backoffs.
- End-host TCP limitations—TCP performance is affected by limitations in the TCP implementation on the client and server. For example, the client or server may not enable the selective acknowledgment (SACK), window scaling, and timestamp options; or, the client or server may limit the size of TCP send and receive buffers to a value that is

insufficient to accommodate the bandwidth delay product (BDP) of a high-latency network.

The TCP optimizations mitigate the effects of these problems. Because the optimizations are performed at the transport level, they work even if the end-to-end connection is encrypted.

The following TCP optimization features are available:

- Latency splitting—TCP proxies are enabled at one or more points in the path between the client and server, splitting the end-to-end latency into smaller segments. Each segment performs loss recovery independently, significantly improving both slow-start convergence and loss recovery times.
- Congestion control—TCP optimization implements bottleneck bandwidth and round-trip (BBR) propagation time, a congestion-control algorithm based on the published IETF specification. BBR measures both the largest amount of recent bandwidth available to a connection and the connection's smallest recent round-trip delay. BBR then uses these metrics to control how fast it sends data and how much& data it allows to be sent at any given time.
- Loss detection and recovery—Versa software implements the recent acknowledgment (RACK) loss-detection algorithm and the supplemental tail loss probe (TLP) algorithm based on the published IETF specification. RACK and TLP help solve the problems caused by lost retransmissions, tail drops, and packet reordering, which are problems that standard TCP loss-detection mechanisms do not mitigate well.
- High-performance TCP options—In environments where performance is limited by the end hosts instead of the network (for example, the end hosts have small buffers or do not support SACK, window scaling, timestamp options, an so forth), Versa software enables these options on their behalf.
- HTTP/HTTPS proxy—Versa HTTP/HTTPS proxy can leverage TCP optimizations to further improve performance in certain use cases, such as selective local internet breakout for applications normally accessed through a proxy and SSL decryption.

TCP Optimization Modes

You can implement TCP optimization in one of the following modes, depending on the kind of traffic being transmitted:

- Auto mode—Auto mode is the simplest way to configure TCP optimization. You use auto mode when there are two or more VOS devices in the network (dual-ended mode). Auto mode can detect which VOS device is closest to the client and which is closest to the server (peer discovery). It then splits the connection between these two devices. You must configure auto mode on all VOS devices in the network path so they can participate in peer discovery. The VOS devices on the WAN side of a TCP segment automatically enable, on the end hosts, any options that are needed for better performance but that are not advertised by the end hosts. Auto mode is useful when when the traffic is sometimes upload-heavy and sometimes download-heavy.
- Splice mode—Splice mode locally splits the TCP connection after an application or URL category is identified or when the VOS device receives the first data packet after the three-way handshake completes. Similar to peer discovery, the three-way handshake completes end to end. However, if the end hosts did not advertise high-performance TCP options, such as SACK or timestamps, these options cannot be enabled during the split. This mode is useful when you want to enable only single-sided optimization, but cannot do a proxy or peer discovery because not enough information is available in the first synchronization (SYN) packet to match the correct rule. Splice mode is mainly useful for optimizing traffic that is destined to a web proxy at a hub location. On the hub, because all application traffic is destined to the web proxy address, the SYN packet alone is not sufficient to determine the application and match the correct rule.
- Proxy mode—In proxy mode, you configure a single VOS device to be a proxy for the TCP connection instead of doing end-to-end peer discovery. The first SYN packet does not travel end to end, but is terminated locally by the VOS device. Proxy mode is similar to splice mode, but you cannot use proxy mode on a hub when the proxy is

located behind the hub.
- Forward proxy—Forward proxy optimizes data being sent from clients to servers. You configure it on the VOS device closer to the client.
- Reverse proxy—Reverse proxy optimizes data being sent from servers to clients. You configure it on the VOS device closer to the server.

Note that TCP optimizations are mainly optimizations on the transmission side: they improve performance for bulk data transfer across high-latency and lossy WANs. In these cases, auto mode is often the simplest and best configuration, because a TCP proxy is created close to the client, to optimize uploads, and a TCP proxy is also created close to the server, to optimize downloads. However, there may be cases in which the overhead of two TCP proxies in the path offsets the benefit of TCP optimization. In these cases, and especially if you understand the nature of the application well, you can use one of the single-ended TCP optimization modes (forward/reverse proxy or splice mode). For example, if you know that a particular application is always download-heavy, it may be sufficient to configure a reverse proxy close to the server. On the other hand, if it is clear that the application is always upload-heavy, it may be sufficient to configure a forward proxy close to the client.

Even though most of the optimizations are on the transmission side, some of them also help the data receiver. For example, consider a client behind a poor last-mile link accessing an application across a high-latency network. Even if the application only downloads large amounts of data from the server, the throughput may be limited by one of the following:

- The client uses only a small receive buffer, and so the throughput is limited to the buffer size bytes per round-trip time.
- The client does not support the SACK option, which means that loss recovery is inefficient.

You can address both these problems by using a forward proxy, which can advertise the SACK option to the server and supports a much larger receive buffer.

Configure TCP Optimizations

To configure the TCP optimization features, you do the following:

- Configure TCP profiles
- Configure TCP optimization policies

Configure TCP Profiles

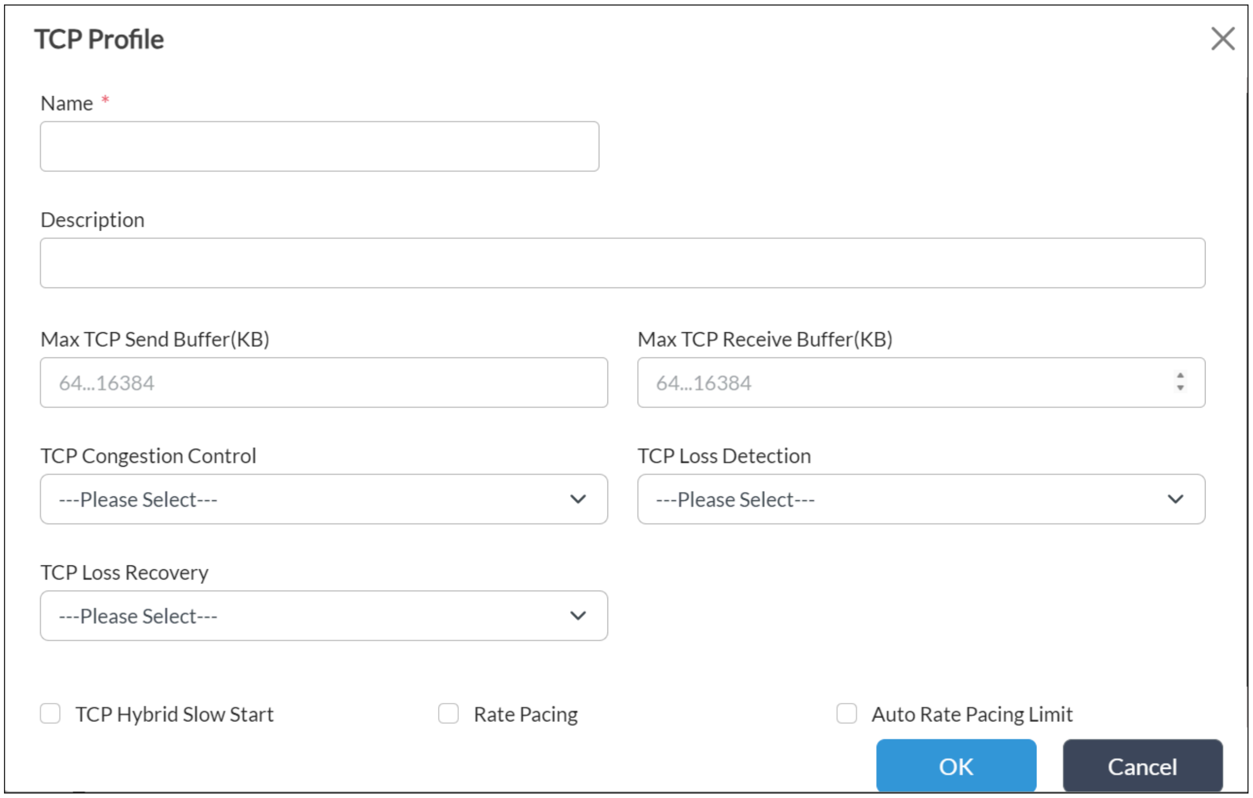
In a TCP profile, you configure settings that apply to a proxy or a split connection.

To configure a TCP profile:

1. In Director view:
    a. Select the Configuration tab in the top menu bar.
    b. Select Templates > Device templates in the horizontal menu bar.

    c. Select an organization in the left menu bar.
    d. Select a post-staging template in the main pane. The view changes to Appliance view.
2. Select the Configuration tab in the top menu bar.
3. Select Objects & Connectors > Objects > TCP Profile in the left menu bar. The main pane displays a table of TCP profiles.

4. Click the + Add icon. In the TCP Profile screen, enter information for the following fields.



Field	**Description**
Name (Required)	Enter a name for the TCP profile.
Description	Enter a description for the TCP profile.
Maximum TCP Send Buffer	Enter the maximum size of the TCP send buffer. Setting the buffer size limits TCP memory consumption. *Range:* 64 through 16777216 bytes (16 MB) (for Releases 21.1.1 and later); 64 through 8192 bytes (8 KB) (for Releases 21.1.0 and earlier)

Field	Description
	Default: 4096 bytes (4 KB)
Maximum TCP Receive Buffer	Enter the maximum size of the TCP receive buffer. Setting the buffer size limits TCP memory consumption. *Range:* 64 through 16777216 bytes (16 MB) (for Releases 21.1.1 and later); 64 through 8192 bytes (8 KB) (for Releases 21.1.0 and earlier) *Default:* 4096 bytes (4 KB)
TCP Congestion Control	Select the congestion control algorithm to use: ◦ New Reno congestion control algorithm—An algorithm that responds to partial acknowledgments. ◦ Cubic congestion control algorithm—An algorithm that uses a cubic function instead of a linear window increase function to improve scalability and stability for fast and long-distance networks. This is the default algorithm. ◦ BBR congestion control algorithm—Bottleneck bandwidth and round-trip propagation time (BBR) measures both the largest amount of recent bandwidth available to a connection and the connection's smallest recent round-trip delay. BBR then uses these metrics to control how fast it sends data and how much data it allows to be sent at any given time. *Default:* Cubic congestion control algorithm
TCP Loss Detection	Select the method of TCP loss detection to use: ◦ Duplicate ACKs—Loss detection based on duplicate ACKs. This is the default method. ◦ Recent acknowledgment (RACK) *Default:* Duplicate ACKs
TCP Loss Recovery	Select the method of TCP loss detection to use:

Field	Description
	◦ Proportional rate reduction—Perform TCP loss detection as described in RFC 6937. ◦ Pipe algorithm—Perform TCP loss detection as described in RFC 6675.
TCP Hybrid Slow Start	Click to enable TCP hybrid slow start. By default, hybrid slow start is disabled.
Rate Pacing	Click to enable rate pacing. Rate pacing injects packets smoothly into the network, thereby avoiding transmission bursts, which could lead to packet loss. By default, rate pacing is disabled.

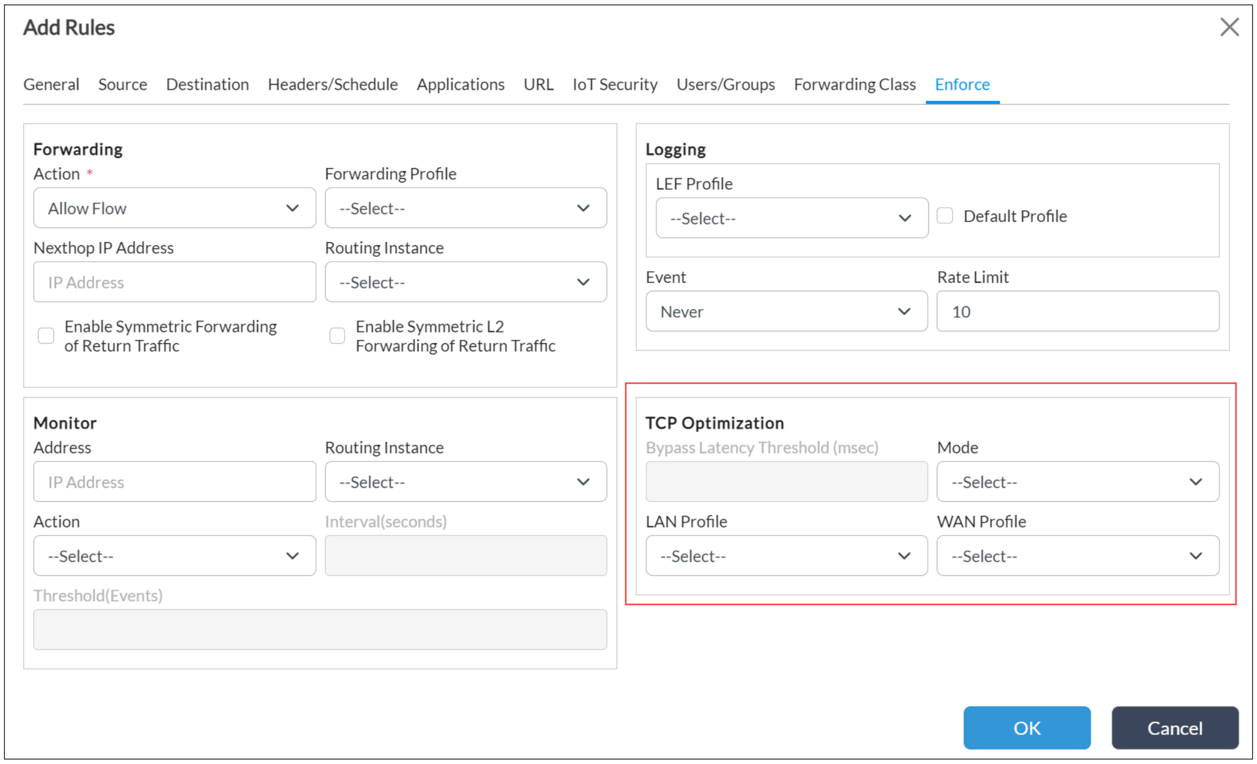
5. Click OK.

Configure TCP Optimization Policies

TCP optimization policies use standard policy-match language to determine which sessions to optimize and how to optimize them.

To configure a TCP optimization policy:

1. In Director view:
    a. Select the Configuration tab in the top menu bar.
    b. Select Templates > Device templates in the horizontal menu bar.
    c. Select an organization in the left menu bar.
    d. Select a post-staging template in the main pane. The view changes to Appliance view.
2. Select the Configuration tab in the top menu bar.
3. Select Services > SD-WAN > Policies in the left menu bar. The main pane displays a table of policies.
4. Select Rules in the horizontal menu bar.
5. Click the + Add icon. In the TCP Profile screen, click the Enforce tab and enter information for the following fields.


Add Rules

Forwarding

Action *: Allow Flow
Forwarding Profile: --Select--
Nexthop IP Address: IP Address
Routing Instance: --Select--
Enable Symmetric Forwarding of Return Traffic
Enable Symmetric L2 Forwarding of Return Traffic

Logging

LEF Profile: --Select--
Default Profile
Event: Never
Rate Limit: 10

Monitor

Address: IP Address
Routing Instance: --Select--
Action: --Select--
Interval(seconds)
Threshold(Events)

TCP Optimization

Bypass Latency Threshold (msec)
Mode: --Select--
LAN Profile: --Select--
WAN Profile: --Select--

Field	**Description**
TCP Optimization (Group of Fields)	
◦ Bypass Latency Threshold	Enter how much latency must be measured before TCP optimizations begin. *Range:* 0 through 60000 milliseconds *Default:* 10 milliseconds
◦ Mode	Select a TCP optimization mode: ◦ Auto ◦ Bypass—Disable TCP optimizations. ◦ Forward proxy ◦ Proxy ◦ Reverse proxy ◦ Splice
◦ LAN Profile	Select a LAN profile.

Field	Description
	For proxy mode, you must configure a TCP profile. If you do not select TCP profiles, a system default LAN profile is applied that uses the cubic congestion control algorithm and duplicate ACK loss detection.
◦ WAN Profile	Select a WAN profile. For proxy mode, you must configure a TCP profile. If you do not select TCP profiles, a system default WAN profile is applied that uses the BBR congestion control algorithm and RACK loss detection.

6. Click OK.

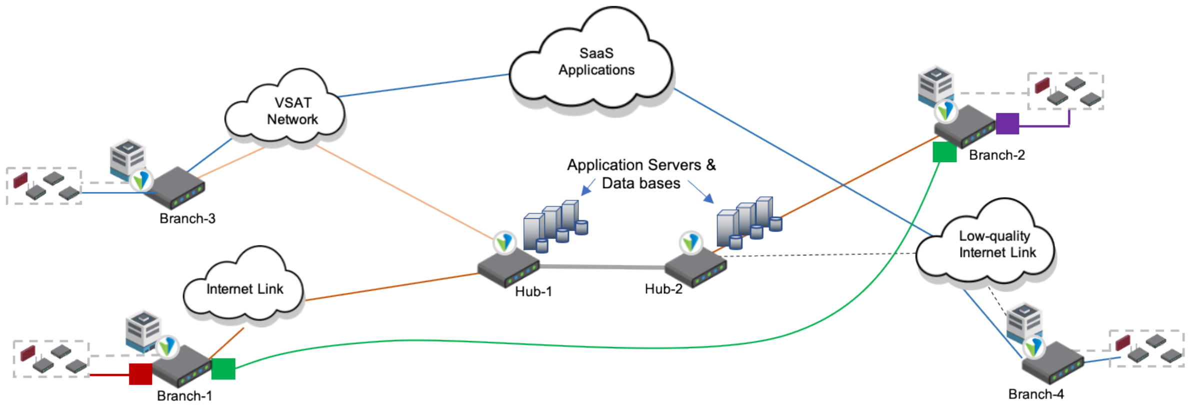
Use Case Scenarios

This section describes examples of how to use TCP proxies in different scenarios.

Scenario 1: Client Traversing Two Hubs To Access an Application

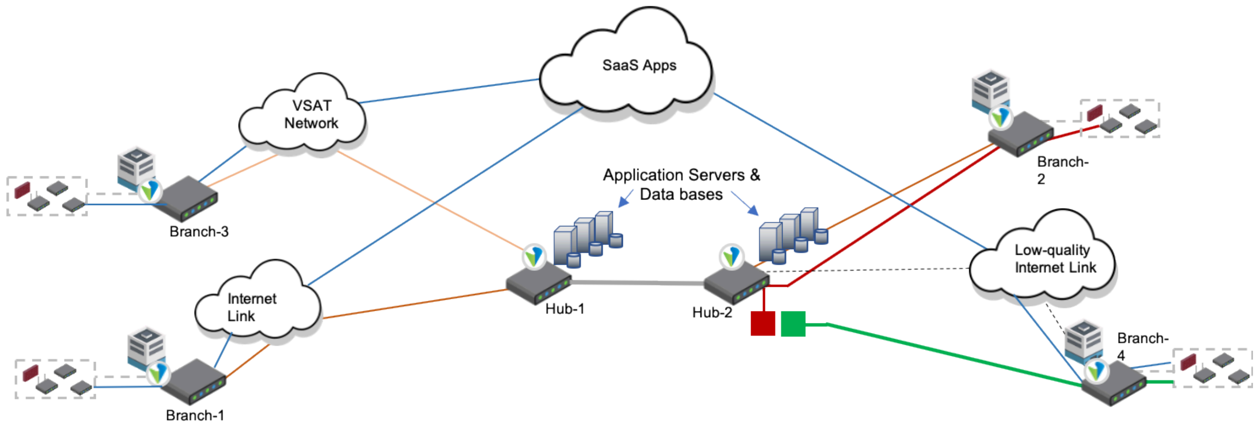
In this scenario, an application behind Branch-1 is communicating with an application behind Branch-2. The resulting TCP connections experience large round-trip times, and the number of hops increases the chances of random drops in the network, which reduces the effective rate at which TCP can communicate.

With TCP optimization, peer discovery places TCP proxies in Branch-1 and Branch-2, which creates three independent TCP connections. The TCP connection between the client and server is split into three TCP connections: Client to Branch-1, Branch-1 to Branch-2, and Branch-2 to server. TCP optimizations, including BBR congestion control and RACK loss detection, are used on the Branch-1 to Branch-2 connection to improve throughput.



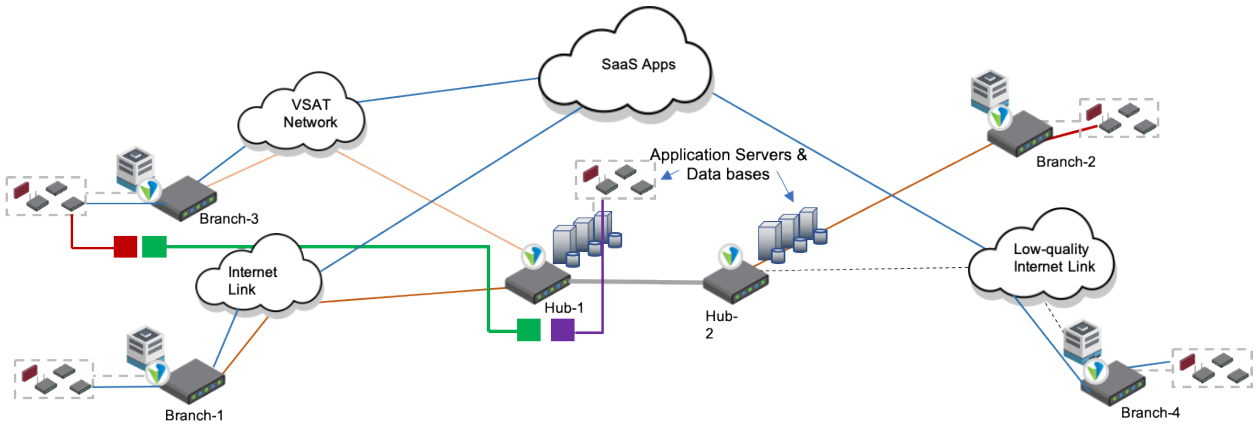
Scenario 2: Client Using a Low-Quality Internet Link To Access an Application Hosted in Another Branch

In this scenario, the Branch-4 link is a low-quality internet link that is likely to drop packets. When a user behind Branch-4 tries to download a file from an application server in Branch-2, the goodput is low because of frequent packet drops, resulting in a poor user experience. This problem is solved by implementing proxy mode optimization on Hub-2. Proxy mode optimization splits the end-to-end latency between Branch-4 and Branch-2 at the hub, improving loss recovery and goodput. Alternatively, you can configure a reverse proxy at Branch-2, where BBR and RACK loss detection improve goodput without latency splitting.



Scenario 3: Client Using a VSAT Link To Access a Hub-Hosted Application

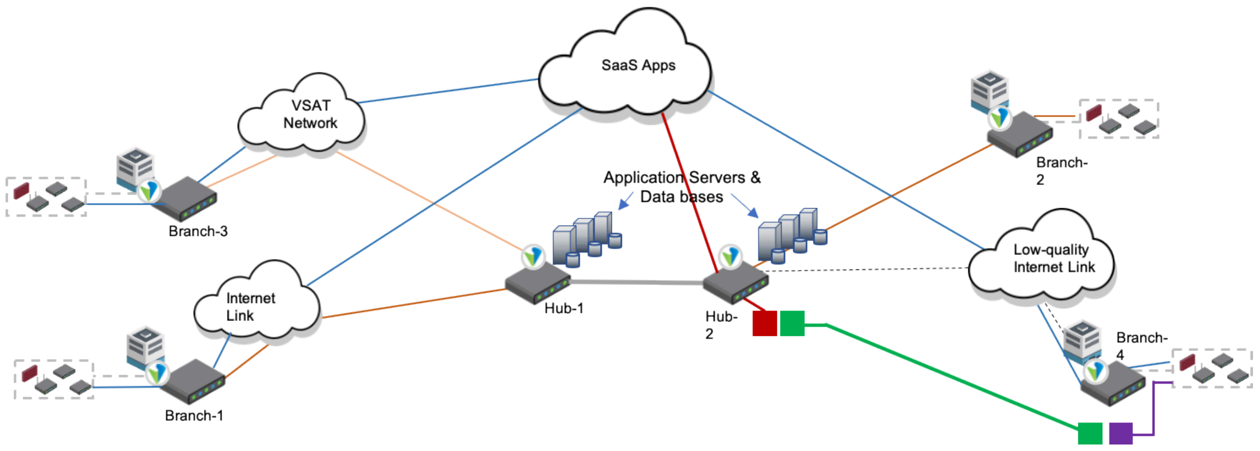
In this scenario, Branch-3 is connected to rest of the network over a very small aperture terminal (VSAT) link, which inherently has long delays, in the range of 500 to 1000 milliseconds, and experiences random losses. A user connecting to an application behind Hub-1 experiences a slow connection. TCP optimization in auto mode splits the end-to-end connection into three segments, and the TCP connection between Branch-3 and Hub-1 uses BBR and RACK loss detection to improve throughput.



Scenario 4: Client Using a Low-Quality Internet Link To Access a Cloud-Based Application

TCP optimization is useful when users behind a lossy link try to access cloud-based applications. When a local breakout is used, the resulting user experience is degraded because of losses introduced on the access link.

In this scenario, it is important for the TCP connection to have a short round-trip time so that it can recover more quickly from the lost packets. Splitting the TCP connection between Branch-4 and Hub-2 (short round-trip time) and between Hub-2 and the cloud-based SaaS applications (longer round-trip time) allows the losses on the lossy internet connection to be recovered faster and improves the end-to-end user experience.



Supported Software Information

Releases 21.1 and later support all content described in this article, except:

- In Releases 21.1.1, increase the maximum send and receive buffer sizes from 8 MB to 16MB; add support for forward proxy and reverse proxy TCP optimization modes.