SRT
SECURE
RELIABLE
TRANSPORT

Secure Reliable Transport Protocol

Deployment Guide

Version 1.3 | Issue 01

est-multi-audio
Reset

ATISTICS

tate	STREAMING
Jp Time	1h29m29s
Sent Packets	1,251,578
Sent Bytes	962,102,880
Bitrate	1,453 kbps

SRT

Reconnections	0
Resent Packets	0
Resent Bytes	0
Received ACKs	342,428
NAKs	0

SRT Last 5 Minutes
Download CSV File

SRTALLIANCE

www.srtalliance.org

SRT
SECURE
RELIABLE
TRANSPORT

DEPLOYMENT GUIDE

Secure Reliable Transport (SRT)

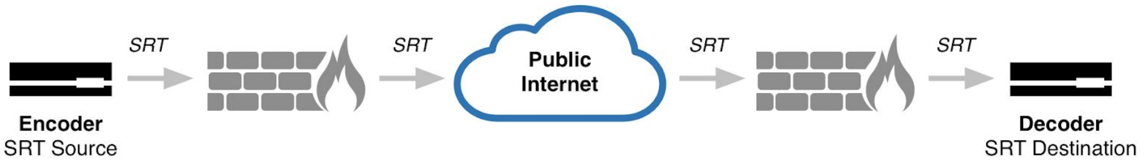
SRT is a transport protocol that enables the secure, reliable transport of data across unpredictable networks, such as the Internet. While any data type can be transferred via SRT, it is particularly optimized for audio/video streaming. This document provides guidance on setting up and deploying SRT technology, which is a feature of an increasing number of Haivision products.

Introduction to SRT

Secure Reliable Transport (SRT) is an open source transport technology that optimizes streaming performance across unpredictable networks, such as the Internet.

Secure	Encrypts video streams
Reliable	Recovers from severe packet loss
Transport	Dynamically adapts to changing network conditions

SRT is applied to contribution and distribution endpoints as part of a video stream workflow to deliver the best quality and lowest latency video at all times.



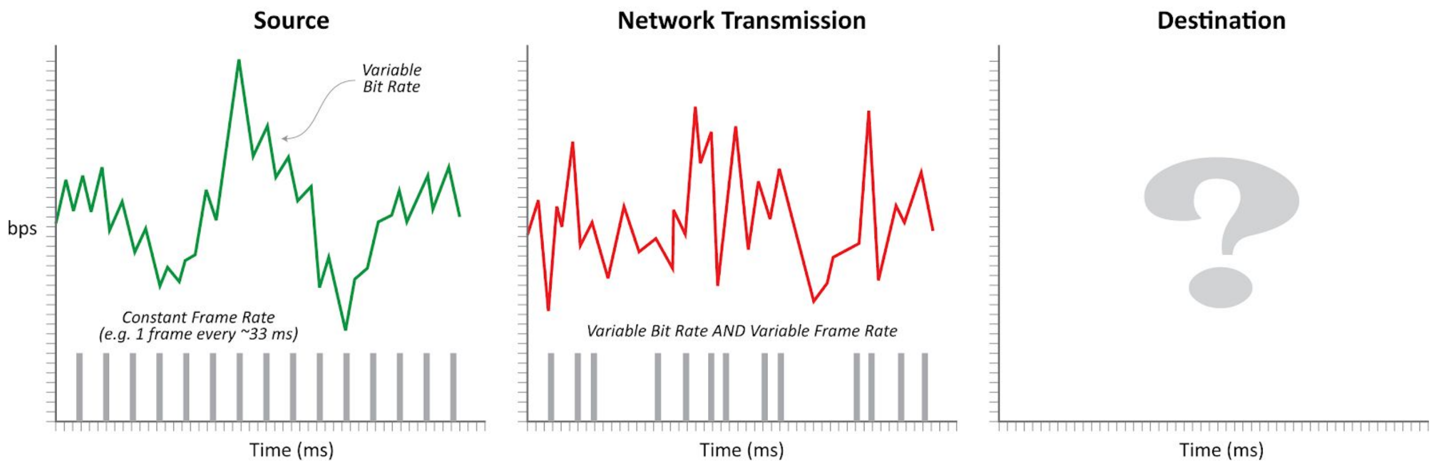
As audio/video packets are streamed from a source to a destination device, SRT detects and adapts to the real-time network conditions between the two endpoints. SRT helps compensate for jitter and bandwidth fluctuations due to congestion over noisy networks, such as the Internet. Its error recovery mechanism minimizes the packet loss typical of Internet connections. And SRT supports AES encryption for end-to-end security, keeping your streams safe from prying eyes.

SRT has its roots in the UDP-based Data Transfer (UDT) protocol. While UDT was designed for high throughput file transmission over public networks, it does not do well with live video. SRT is a significantly modified version that supports live video streaming.

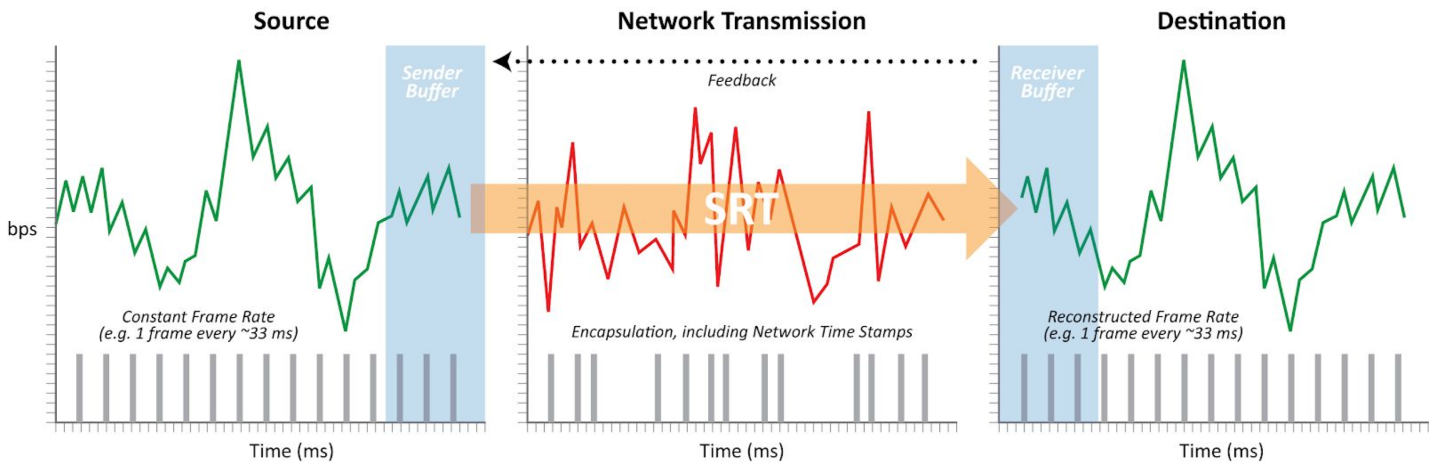
Low latency video transmission across IP based networks typically takes the form of MPEG-TS unicast or multicast streams using the UDP protocol. This solution is perfect for protected networks, where any packet loss can be mitigated by enabling forward error correction (FEC). Achieving the same low latency between sites in different cities, countries or even continents is more challenging. While it is possible with satellite links or dedicated MPLS networks, these are expensive solutions. The use of cheaper public internet connectivity, while less expensive, imposes significant bandwidth overhead to achieve the necessary level of packet loss recovery.

Even though UDT was not designed for live streaming, its packet loss recovery mechanism provided an interesting starting point. The original version of SRT included new packet retransmission functionality that reacted immediately to packet loss to enable live streaming.

To achieve low latency streaming, SRT had to address timing issues. The characteristics of a stream from a source network are completely changed by transmission over the public internet, which introduces delays, jitter, and packet loss. This, in turn, leads to problems with decoding, as the audio and video decoders do not receive packets at the expected times. The use of large buffers helps, but latency is increased.


Source
Network Transmission
Destination
Variable Bit Rate
bps
Constant Frame Rate (e.g. 1 frame every ~33 ms)
Variable Bit Rate AND Variable Frame Rate
?
Time (ms)
Time (ms)
Time (ms)

SRT includes a mechanism that recreates the signal characteristics on the receiver side, dramatically reducing the need for buffering. This functionality is part of the SRT protocol itself, so once data comes out of an SRT connection on the receiver side, the stream characteristics have been properly recovered.


Source
Network Transmission
Destination
Sender Buffer
Feedback
Receiver Buffer
bps
SRT
Constant Frame Rate (e.g. 1 frame every ~33 ms)
Encapsulation, including Network Time Stamps
Reconstructed Frame Rate (e.g. 1 frame every ~33 ms)
Time (ms)
Time (ms)
Time (ms)

Initially developed by Haivision Systems Inc., the SRT protocol was released as open source in April 2017 in partnership with Wowza Media Systems Inc. Open source SRT is distributed under MPL-2.0, which was chosen because it strikes a balance between driving adoption for open source SRT, while encouraging contributions to improve upon it by the community of adopters.

Any third party is free to use the SRT source in a larger work regardless of how that larger work is compiled. Should they make source code changes, they would be obligated to make those changes available to the community.

In May 2017, Haivision and Wowza founded the SRT Alliance (www.srtalliance.org), a consortium dedicated to the continued development and adoption of the protocol.

About the SRT Alliance

The SRT Alliance, founded by Haivision and Wowza, is a commercially funded group dedicated to managing and supporting the open source implementation of SRT, a transport protocol for enabling the delivery of high-quality, low-latency video across the public Internet. This alliance is accelerating interoperability of video streaming solutions and fostering collaboration with industry leaders to achieve lower latency internet video transport. The SRT Alliance is open to new members. For companies who want to participate actively in growing the ecosystem of SRT in low latency video streaming workflows, please contact us at info@srtalliance.org.

About Haivision

Haivision is a global leader in delivering advanced video networking, digital signage, and IP video distribution solutions. Haivision offers complete end-to-end technology for video, graphics, and metadata to help customers to build, manage, and distribute their media content to users throughout an organization or across the Internet. Haivision has specific expertise in the enterprise, education, medical/healthcare, and federal/military markets.

Haivision is based in Montreal and Chicago, with technical centers in Beaverton, Oregon; Austin, Texas; and Hamburg, Germany.

About Wowza

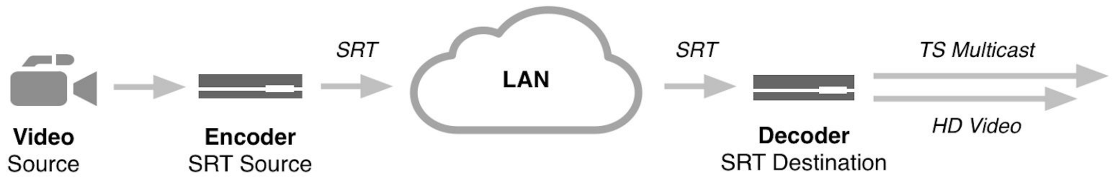
Wowza Media Systems™ is the recognized gold standard of streaming, that enables organizations to expand their reach and more deeply engage their audiences on any device, anywhere in the world.

Table of Contents

SRT Solutions

SRT is a point-to-point, connection-oriented protocol. Each SRT stream is characterized by the transport of multimedia data and control messages, with only one UDP connection used per SRT stream. Nonetheless, it is possible to configure SRT solutions that encompass a variety of situations.

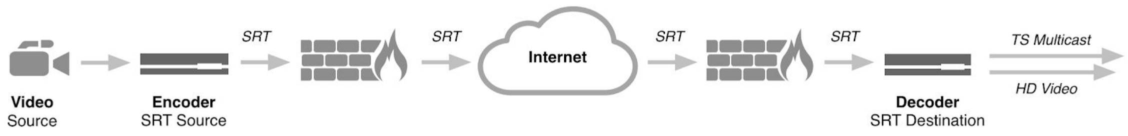
Point to Point & Interactive



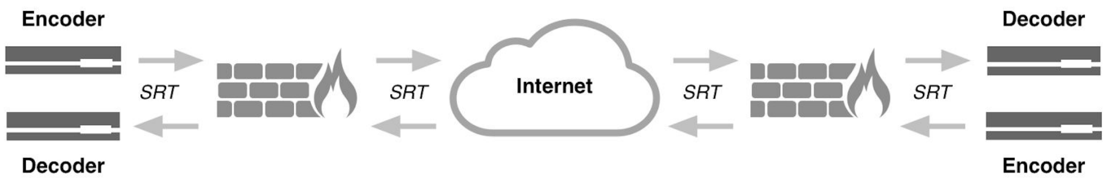
SRT can be easily provisioned for straightforward connections within and between facilities to achieve a low latency, low cost video distribution.

It can be useful over noisy, unreliable local LANs. In large organizations with many users on the same LAN, congestion and packet drops are common — video is very sensitive to this. Even inside the same building on a controlled LAN, SRT can enhance the experience.

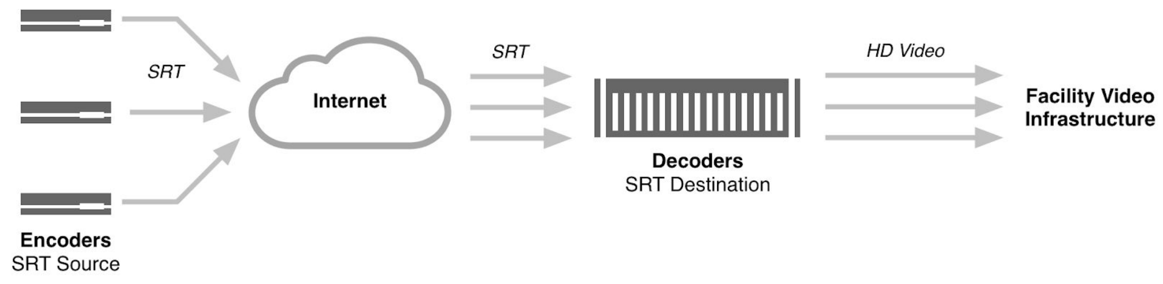
Some networks have VLANs with dedicated bandwidth (where routers prioritize traffic) that require a deep knowledge of routers and switches. SRT removes the need for IT intervention to get video through the network.



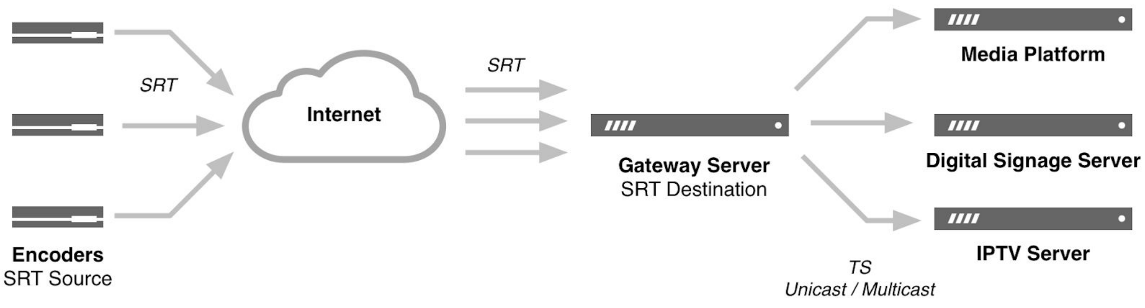
The low latency achievable with SRT is even suitable for interactive applications over the Internet.



Contribution & Aggregation

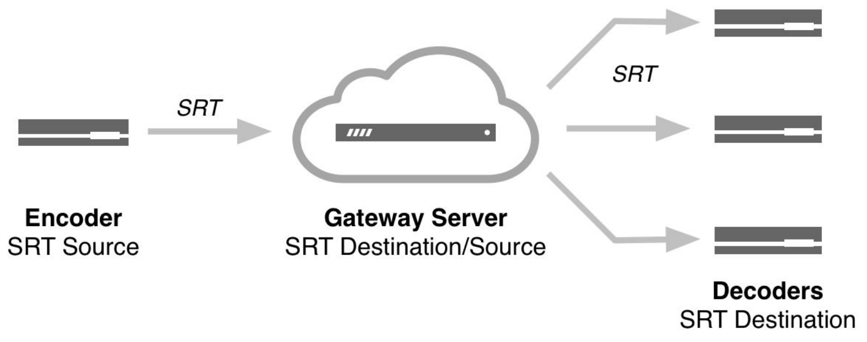


Multiple SRT source and destination devices can be configured to feed high-demand video infrastructures.



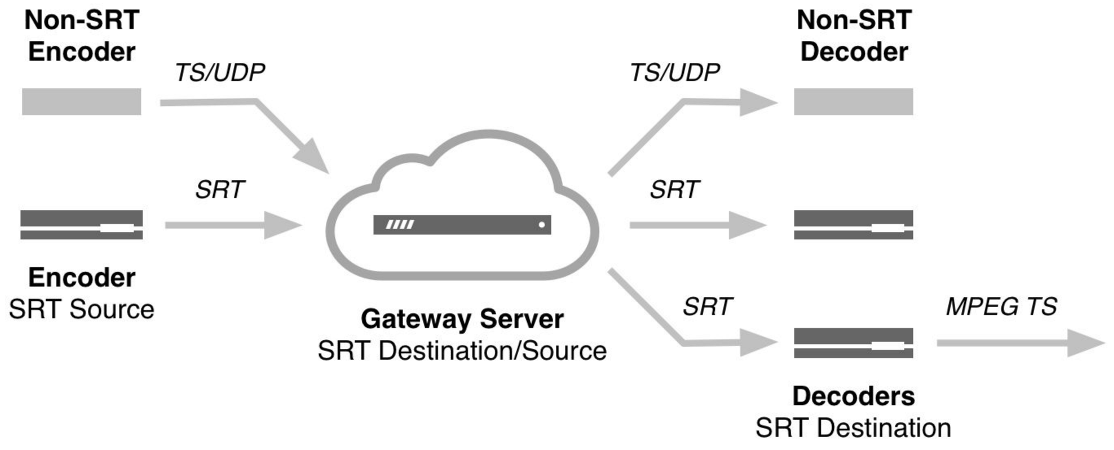
Multiple SRT streams can be aggregated and re-distributed via unicast and multicast into recording, IPTV and digital signage services.

Point to Multi-Point



While SRT operates on point-to-point connections, Haivision’s Media Gateway is designed to support multiple such connections. A single incoming SRT stream can be redistributed via one or more Media Gateways to multiple SRT destination devices.

SRT Stream Flipping



Some SRT destination devices (such as decoders and gateway servers) support "stream flipping" — the ability to convert back and forth between an SRT stream and a standard MPEG Transport Stream (TS). This is accomplished by de-encapsulating an incoming SRT stream and re-streaming it as a TS/UDP stream (and vice versa), and is typically done to allow devices (on a local LAN) that do not support SRT to have access to incoming SRT stream content.

For more information on the numerous ways in which SRT can be deployed across a wide array of solutions, please refer to our members' web sites.

SRT Version History & Compatibility

Since its release as an open source protocol, SRT has been adopted by a large number of vendors. To learn more about interoperability between SRT-enabled products, search for members who have been certified "SRT Ready" on the SRT Alliance website:

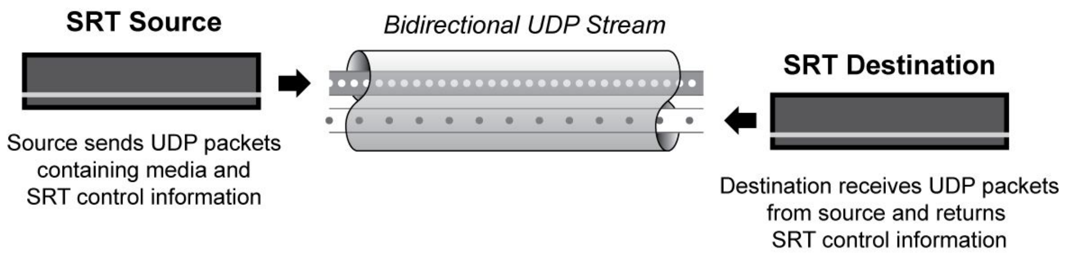
https://www.srtalliance.org/members/

SRT is designed to be backwards compatible, so that new or upgraded products will be able to support SRT interactions with older ones. Note, however, that newer versions of SRT may have features unavailable on older ones.

Basic SRT Concepts

Source & Destination

The SRT protocol relies on bi-directional UDP traffic to optimize video streaming over public networks. In addition to the video data that is sent from a content source device (such as an encoder) to a destination (such as a decoder), there is a constant exchange of control information between the two endpoints, including "keep alive" packets (if needed) approximately every 10 ms, which enable SRT streams to be automatically restored after a connection loss.


SRT Source
Bidirectional UDP Stream
SRT Destination
Source sends UDP packets containing media and SRT control information
Destination receives UDP packets from source and returns SRT control information

NOTE: *It is important to understand that with an SRT stream, the **source** is the device that is sending the content (audio and/or video data), while the **destination** is the device receiving the content. Elsewhere, you may encounter references to an SRT sender and an SRT receiver, but to avoid confusion in this document we will be using the terms **source** and **destination**.*[1]

In some cases, a device can act as both the source and the destination. For example, a gateway server may act as a destination while receiving an SRT stream from an encoder, and then become a source device as it re-streams to a decoder.

[1] There is room for confusion inherent in discussions of SRT, where the terms "caller/listener", "sender/receiver", and "source/destination" often appear. But "caller/listener" has no direct correspondence with "source/destination" — each of the latter can be either of the former. And a destination device can be any intermediate node in an SRT streaming chain, such as a firewall. You can "call" an SRT source or an SRT destination.

SRT Connection Modes

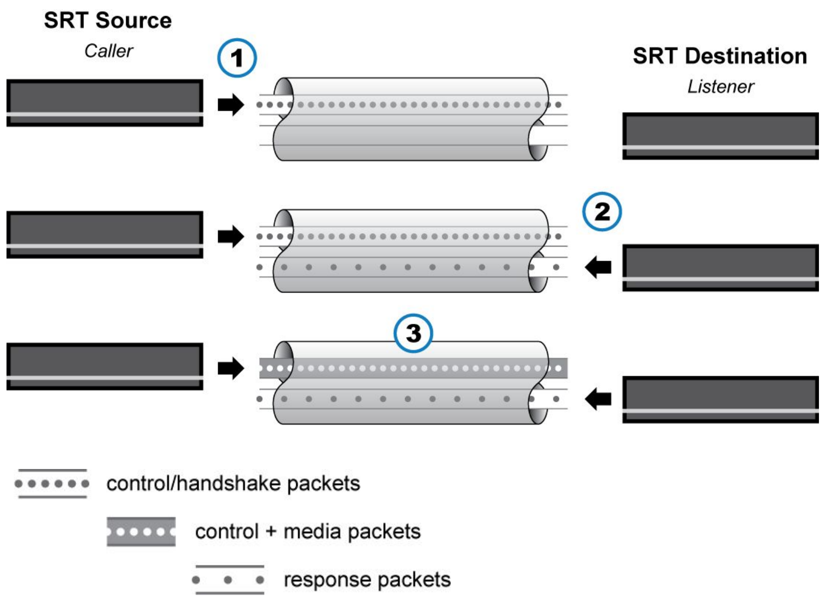
In order to establish a connection, SRT employs a handshake mechanism where each device identifies itself as a *Caller* or as a *Listener*. In certain cases, two devices can simultaneously negotiate an SRT connection in *Rendezvous* mode. As you are configuring SRT streams, you should understand these handshaking modes and when to apply them:

Mode	What it does	When to use it
Caller	Sets a source or destination device as the initiator of an SRT connection. The *Caller* device must know the public IP address and port number of the *Listener*.	To initiate point-to-point streaming: (1) on a source or destination device that is behind a firewall; may require a network administrator to configure the firewall settings. (2) on a source or destination device that is not behind a firewall. (3) on a source or destination device with a dynamic IP address (e.g. a portable encoder using DHCP). **Example**: Set an encoder to *Caller* mode to connect to a decoder over a private network, or vice versa.
Listener	Sets a device to wait for a request to open an SRT connection. The *Listener* device only needs to know that it should listen for SRT packets on a certain port.	To accept a point-to-point connection initiated by a *Caller*: (1) on a source or destination device that is behind a firewall over which you have control and can open a port. (2) on a source or destination device that is not behind a firewall, or exposed directly on the Internet. (3) when you know that another device will initiate the session. **Example**: Set a decoder to *Listener* mode to accept an SRT connection from an encoder.
Rendezvous	Allows two devices to negotiate an SRT connection over a mutually agreed upon port. Both source and destination must be in *Rendezvous* mode.	To establish a point-to-point SRT connection when one or both devices are behind firewalls. Once certain settings are in place on the firewall, SRT connections can be initiated without further intervention by a network administrator.

SRT Connection Mode Examples

In Figure 8 below, the SRT source device is in *Caller* mode, and the SRT destination device is in *Listener* mode. The SRT source (*Caller*) initiates the handshake by sending a series of control packets in a UDP stream (1). When the SRT destination (*Listener*) receives these control packets, it responds by sending its own (2). Once the handshake has successfully completed, the SRT source device begins adding media packets to the UDP stream (3).

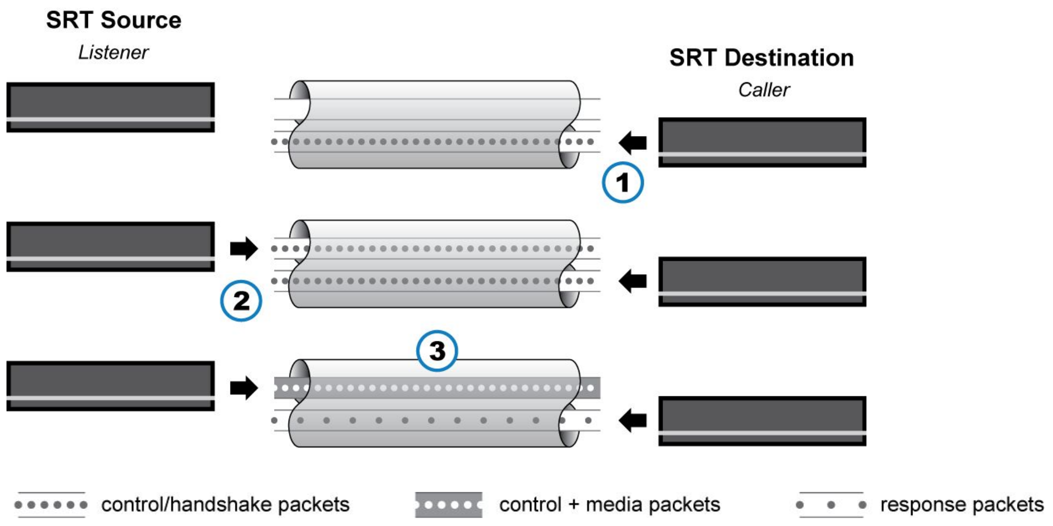
Figure 8 — SRT source device initiates connection



NOTE: *Regardless of which device is in which mode, when the SRT handshake is completed both source and destination continue to exchange control packets containing information about network conditions, dropped packets, etc. Once communication is established, the notion of which device is Caller and which is Listener becomes unimportant. What matters is the source/destination relationship, which is decoupled from the caller/listener relationship.*

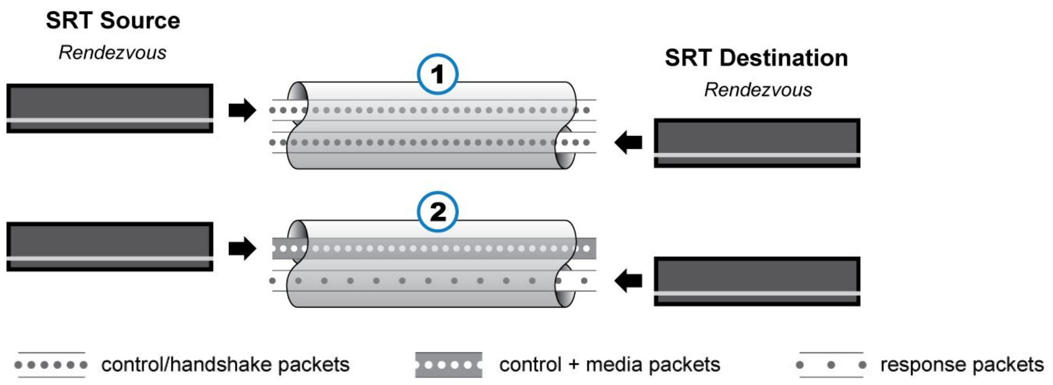
In Figure 9 below, the roles are reversed — the SRT source device is in *Listener* mode, and the SRT destination device is in *Caller* mode. The SRT destination (*Caller*) initiates the handshake by sending a series of control packets in a UDP stream (1). When the SRT source (*Listener*) receives these control packets, it responds by sending its own (2). Once the handshake has successfully completed, the SRT source device begins adding media packets to the UDP stream (3).

Figure 9 — SRT destination device initiates connection



In Figure 10 below, both the SRT source device and the SRT destination device are in *Rendezvous* mode. Both devices send a series of control packets in a UDP stream (1). Once the handshake has successfully completed, the SRT source device begins adding media packets to the UDP stream (2).

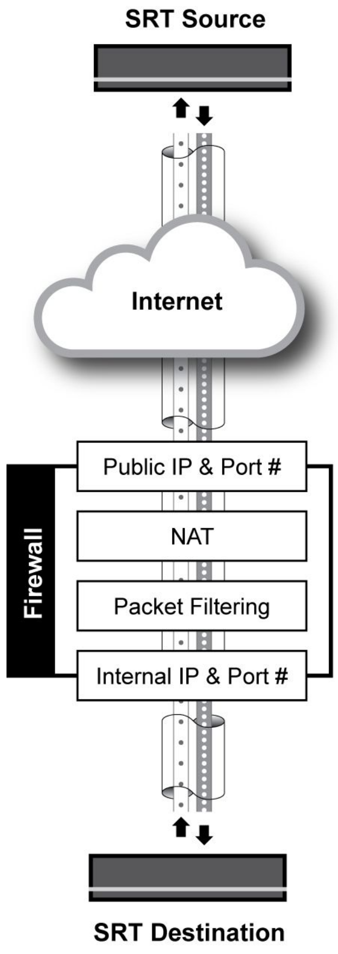
Figure 10 — SRT connection initiated using Rendezvous



SRT and Firewalls

In many real world situations, particularly those involving transmission over the Internet, SRT streams will have to pass through a firewall at the source, at the destination, or at both ends. In order to allow this, a network administrator may have to configure certain settings on the firewall(s), specifically those for Network Address Translation (NAT) and packet filtering. The settings will differ depending on whether the devices behind firewalls are in *Caller*, *Listener*, or *Rendezvous* mode.

Example 1



The figure at right illustrates a simple example, where an SRT source device is attempting to stream across the Internet to an SRT destination behind a firewall. If we consider the case where the SRT source device is in *Caller* mode, and the destination device is in *Listener* mode, then in order for the handshaking process described earlier to be successfully completed (and an SRT streaming session established) certain conditions must be met:

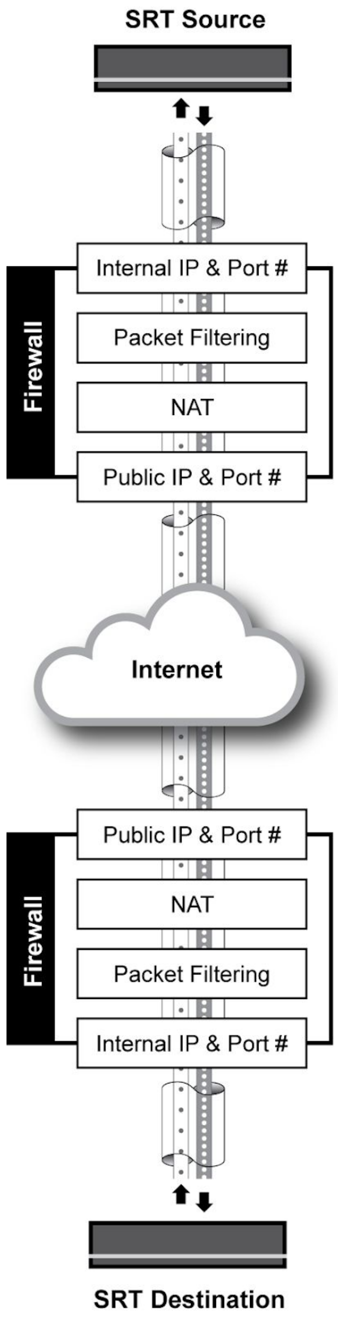
- The SRT source device must “know” the public IP address of the firewall, and the port number on which the SRT destination device is “listening”.
- The firewall must allow the specific destination port used by SRT to be accessible from the Internet.
- The firewall must allow bi-directional UDP traffic.
- Port forwarding must be enabled on the firewall to allow data to flow to the IP address and port of the SRT destination device.
- Packet filtering must be disabled (to allow the SRT packets to pass through).

Example 2

This figure illustrates a more complex example, where an SRT source device behind a firewall is attempting to stream across the Internet to an SRT destination, also behind a firewall. If we consider the case where the SRT source device is in *Caller* mode, and the destination device is in *Listener* mode, then in order for the handshaking process described earlier to be successfully completed (and an SRT streaming session to be established) certain conditions must be met:

- The SRT source device must "know" the public IP address of the destination firewall, and port number on which the destination device is "listening". This information usually comes from the IT Admin responsible for the firewall.
- Both firewalls must allow bi-directional UDP traffic.
- Port forwarding (NAT) must be configured on both firewalls to allow data to flow between the SRT source and destination devices.
- Packet filtering must be configured on both firewalls to allow the exchange of SRT packets between the source and destination devices.

NOTE: *The order in which a firewall performs Network Address Translation and packet filtering will have an impact on how the packet filtering rules are configured.*



Deployment Scenarios

This section describes three common scenarios for deploying SRT:

- **Scenario 1**: Streaming over a private LAN and/or WAN, with no firewall
- **Scenario 2**: Streaming over a public network with both the source and destination behind a firewall (Caller/Listener mode)
- **Scenario 3**: Streaming over a public network with both the source and destination behind a firewall (Rendezvous mode)

In real-world applications, elements of these scenarios can be employed in various ways.

NOTE: *Some of the example values provided in these scenarios (IP addresses, port numbers, etc.) are for illustrative purposes only. For general information on creating and managing streams, please refer to the specific documentation for your product(s).*

NOTE: *The SRT open source project does not contain any web UI elements. For illustrative purposes, this section contains examples of a web interface based on existing SRT-enabled products.*

Scenario 1: Streaming over a LAN or private WAN

In this scenario, an SRT source device (the *Caller*) initiates a point-to-point session with an SRT destination device (the *Listener*) over a LAN (Local Area Network) or a private WAN (Wide Area Network).

Step 1 — Configure the encoder

On the encoder (the SRT source device), do the following:

1. Using the settings in the table below, create and start an output stream:

Setting	Example	Description
Protocol	TS over SRT	SRT is based on the UDP protocol.
Mode	Caller	Encoder will initiate the SRT connection.
Address	198.168.2.20	The target address for the SRT stream, which is the IP address of the decoder.
Source Port	20000	The UDP source port for the SRT stream, which is the unique port over which the encoder will be sending the SRT stream. If you do not specify this UDP source port, an ephemeral source port will be assigned (between 32768 and 61000).
Destination Port	30000	The port over which the decoder will be listening.

Step 2 — Configure the decoder

On the decoder (the SRT destination device), do the following:

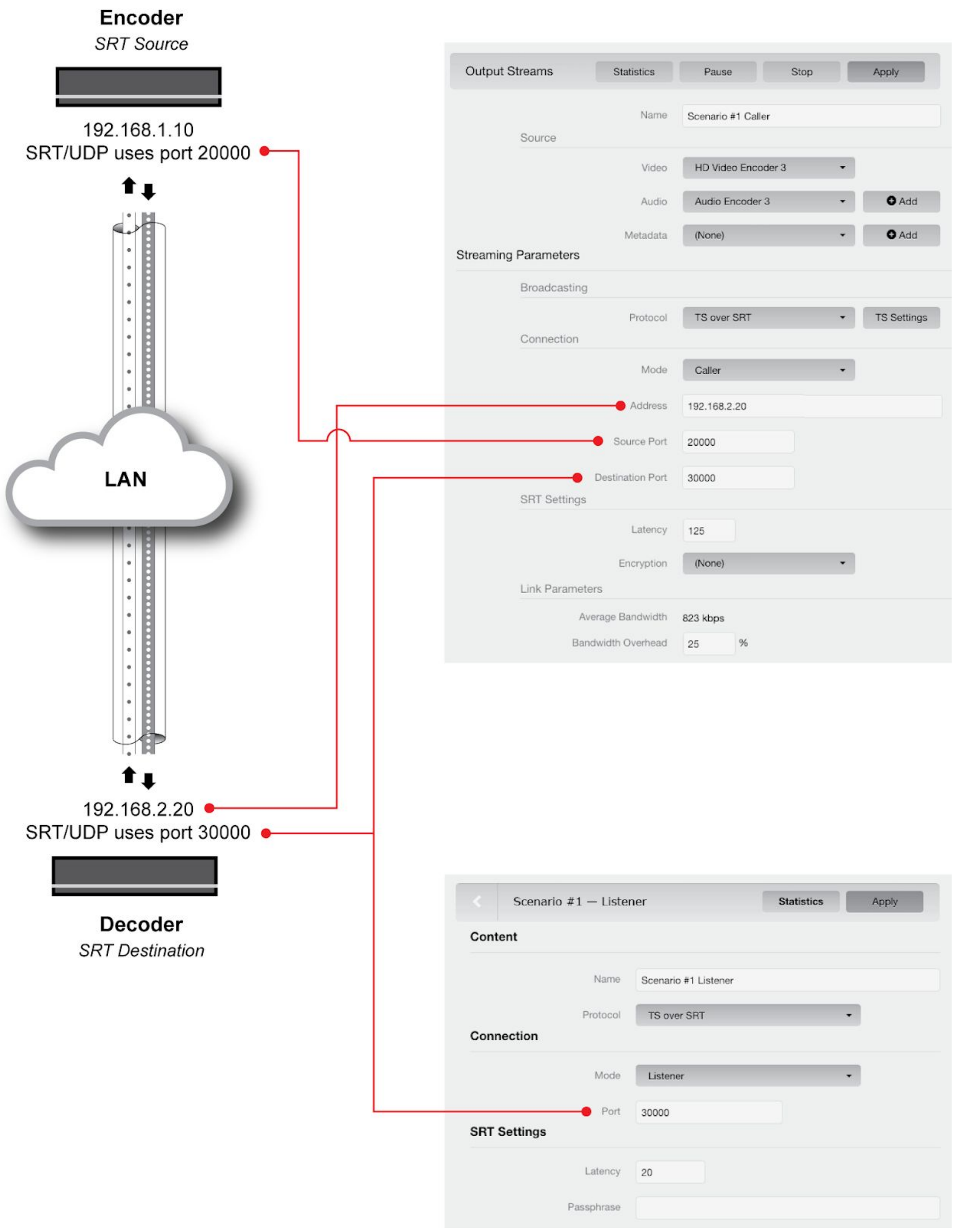
1. Using the settings in the table below, create an input stream:

Setting	**Example**	**Description**
Mode	Listener	The decoder will wait for the source device to initiate the SRT session.
Destination Port	30000	This is the port on which the decoder will be listening.

NOTE: *In the figure below, we are using the same port assignments (20000) for both Caller and its firewall to simplify the scenario. The Caller could, in fact, be using any other port, as long as its firewall had the appropriate mapping to allow return traffic back to the encoder.*

The encoder and Decoder will handshake and establish an SRT session. The encoder will send the video stream to the decoder, which will process the stream and return control packets that includes network throughput, latency and other statistics. The encoder can use this information to adapt its transmission (resend lost packets, adjust bit rate, etc.).

Note that when the SRT handshake is completed both source and destination continue to exchange control packets. Once an SRT connection/session is established, the *Caller* or *Listener* designation becomes unimportant. What matters is the source/destination relationship, or video flow, which is decoupled from the caller/listener relationship.



Output Streams | Statistics | Pause | Stop | Apply

Name	Scenario #1 Caller
Source	
Video	HD Video Encoder 3
Audio	Audio Encoder 3
Metadata	(None)
Streaming Parameters	
Broadcasting	
Protocol	TS over SRT
Connection	
Mode	Caller
Address	192.168.2.20
Source Port	20000
Destination Port	30000
SRT Settings	
Latency	125
Encryption	(None)
Link Parameters	
Average Bandwidth	823 kbps
Bandwidth Overhead	25 %

Scenario #1 — Listener | Statistics | Apply

Content	
Name	Scenario #1 Listener
Protocol	TS over SRT
Connection	
Mode	Listener
Port	30000
SRT Settings	
Latency	20
Passphrase	

Scenario 2: Internet streaming using Caller and Listener modes

In this scenario, an SRT source device (an encoder in *Caller* mode) behind a firewall initiates a point-to-point session over the Internet with an SRT destination device (a decoder in *Listener* mode), also behind a firewall.

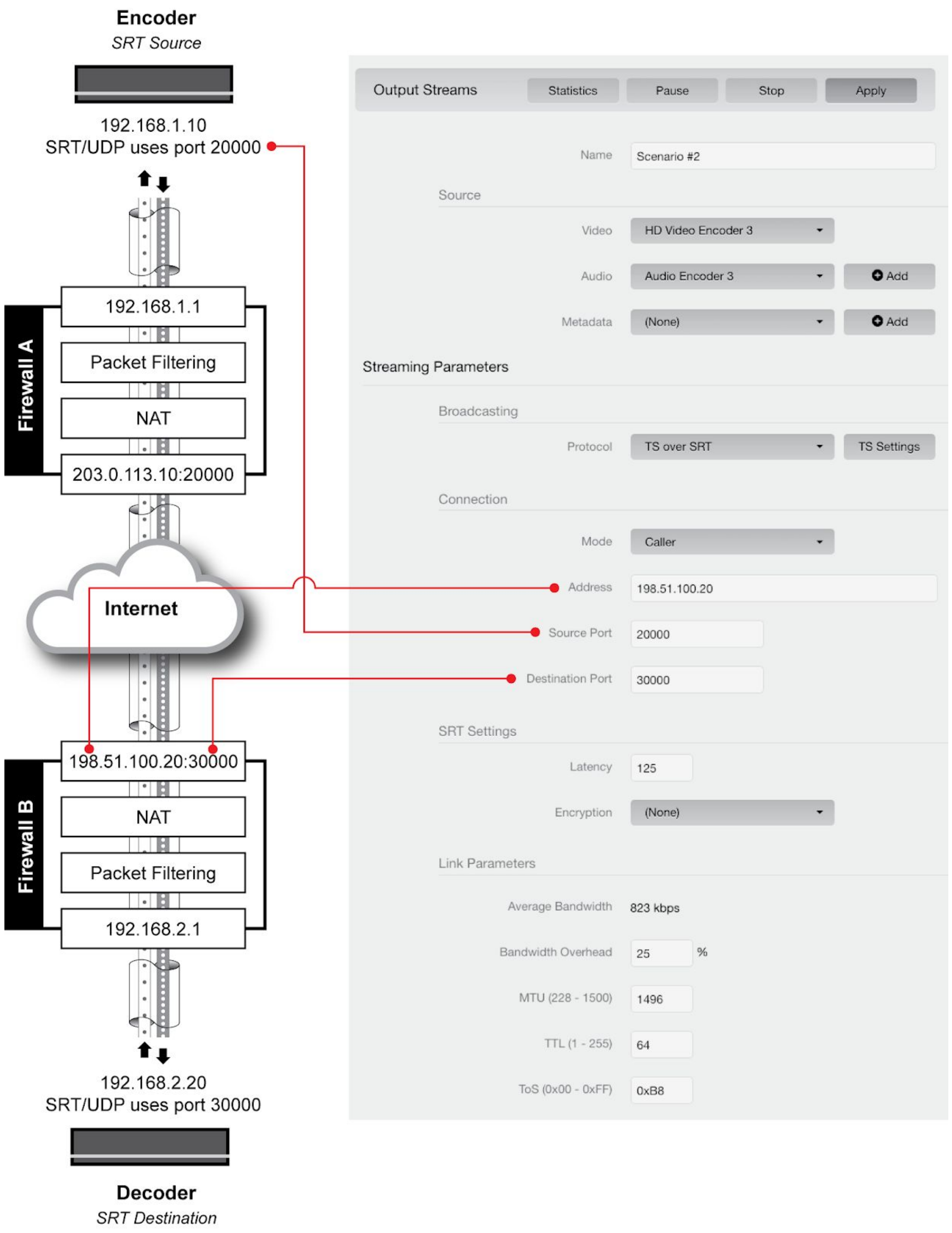
Step 1 — Configure the encoder

On the encoder (the SRT source device), do the following:

1. Using the settings in the table below, create and start an output stream:

Setting	Example	Description
Protocol	TS over SRT	SRT is based on the UDP protocol.
Mode	Caller	Encoder will initiate the SRT connection.
Address	198.51.100.20	The target address for the SRT stream, which is the public IP address of Firewall B (at the destination).
Source Port	20000	The unique UDP source port for the SRT stream; you can leave the default value (Auto-assign), in which case an ephemeral port in the range of 32768 to 61000 is assigned, or, if required by your organization's IT policies, enter a specific (static) port number. If you use Auto-assign, then Firewall A must be configured to map ANY source port from its local side to a specific port on its public side so that return traffic can be directed to the encoder.
Destination Port	30000	The port over which the decoder will be listening. This is the publicly mapped port number for the SRT destination device (i.e. the port that Firewall B opens for the SRT session). This port must be known (it can't be "any").

NOTE: *In the figure below, we are using the same port assignments (20000) for both Caller and its firewall to simplify the scenario. The Caller could, in fact, be using any other port, as long as its firewall had the appropriate mapping to allow return traffic back to the encoder.*



Name: Scenario #2

Source
Video: HD Video Encoder 3
Audio: Audio Encoder 3
Metadata: (None)

Streaming Parameters

Broadcasting
Protocol: TS over SRT

Connection
Mode: Caller
Address: 198.51.100.20
Source Port: 20000
Destination Port: 30000

SRT Settings
Latency: 125
Encryption: (None)

Link Parameters
Average Bandwidth: 823 kbps
Bandwidth Overhead: 25 %
MTU (228 - 1500): 1496
TTL (1 - 255): 64
ToS (0x00 - 0xFF): 0xB8

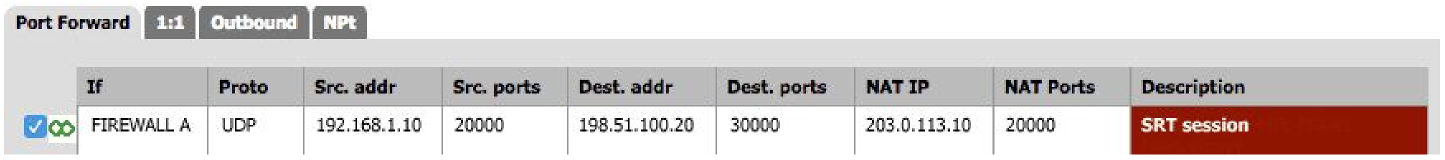
Step 2 — Configure the firewall at the source

On Firewall A (at the source), do the following:

1. Using the settings in the table below, create an outbound NAT rule that allows bidirectional UDP traffic, with a port forwarding entry for incoming traffic on the firewall's public IP address/port that forwards it to the encoder's IP address/port number:

Setting	Example	Description
Protocol	UDP	SRT is based on the UDP protocol.
Source IP	192.168.1.10	The encoder's IP address
Source Port	20000	In this case, we are using a static port assignment. for the source port, but if auto-assigned it can be anything within the ephemeral port range (typically 32768 to 61000 on Linux devices).
Destination IP	198.51.100.20	This is the public IP address of Firewall B
Destination Port	30000	This is the port over which the decoder will be listening.
Outbound NAT Source Port	20000	Your firewall must support Outbound NAT Source Port (which disables outbound NAT port rewrite). Otherwise, Rendezvous mode may be required (see Scenario #3).

Here is an example of an outbound NAT rule for Firewall A:



	If	Proto	Src. addr	Src. ports	Dest. addr	Dest. ports	NAT IP	NAT Ports	Description
✓	FIREWALL A	UDP	192.168.1.10	20000	198.51.100.20	30000	203.0.113.10	20000	SRT session

Step 3 — Configure the firewall at the destination

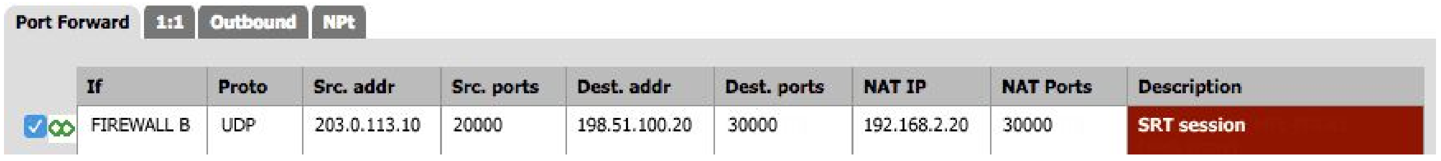
On Firewall B (at the destination), do the following:

1. Using the settings in the table below, create an inbound NAT rule to enable forwarding of SRT traffic to the decoder's IP address/port number:

Setting	Example	Description
Protocol	UDP	SRT is based on the UDP protocol.
Source IP	203.0.113.10	This is the public IP address of the firewall at the source (Firewall A).
Source Port	20000	This must match the Outbound NAT Source Port on the firewall at the source (Firewall A).
Destination IP	198.51.100.20	This is the public (external) IP address of the firewall at the destination (Firewall B).
Destination Port	30000	This is the public (external) port of the firewall at the destination (Firewall B), which in this example is also the port over which the decoder will be listening (dstport).
Redirect Target IP	192.168.2.20	This is the address of the decoder (the internal destination IP).
Redirect Target Port	30000	This is the port over which the decoder will be listening (the internal destination port, or dstport).

NOTE: *The Redirect Target IP and Port are the internal IP and port number that the destination device is using. Destination IP and Port are the firewall's external interface. The port numbers don't have to be the same. For clarity, it may be useful to always use the same port number, but this is up to your Firewall Administrator to decide.*

Here is an example of an inbound NAT rule for Firewall B:

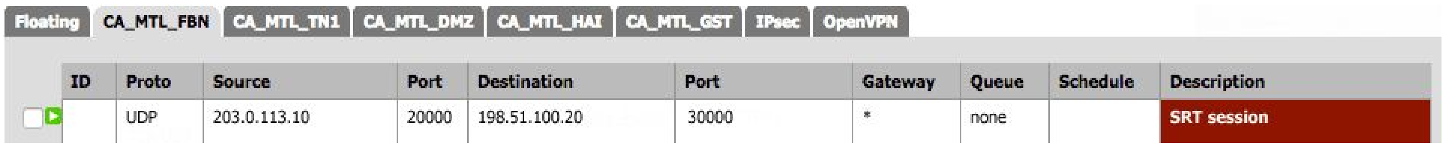


	If	Proto	Src. addr	Src. ports	Dest. addr	Dest. ports	NAT IP	NAT Ports	Description
✓	FIREWALL B	UDP	203.0.113.10	20000	198.51.100.20	30000	192.168.2.20	30000	SRT session

2. Using the settings in the table below, create a packet filtering rule to allow SRT packets to pass freely to and from the decoder's IP address/port number:

Setting	Example	Description
Protocol	UDP	SRT is based on the UDP protocol.
Source IP	203.0.113.10	This is the public IP address of the firewall at the source.
Source Port	20000	This must match the Outbound NAT Source Port on the firewall at the source.
Destination IP	192.168.2.1 or 198.51.100.20	Depending on your firewall, the NAT rules may be applied before or after the packet filtering rules. This will impact the filtering rule definition. If the NAT is applied before, you have to specify the Firewall B internal IP address. If the NAT is applied after, you have to specify the Firewall B public IP address.
Destination Port	30000	This is the port over which the decoder will be listening (dstport).
Policy	Accept/Pass	Allows packets to pass freely between source and destination.

Here is an example of a packet filtering rule for Firewall B:



ID	Proto	Source	Port	Destination	Port	Gateway	Queue	Schedule	Description
	UDP	203.0.113.10	20000	198.51.100.20	30000	*	none		SRT session

Step 4 — Configure the decoder

On the decoder (the SRT destination device), do the following:

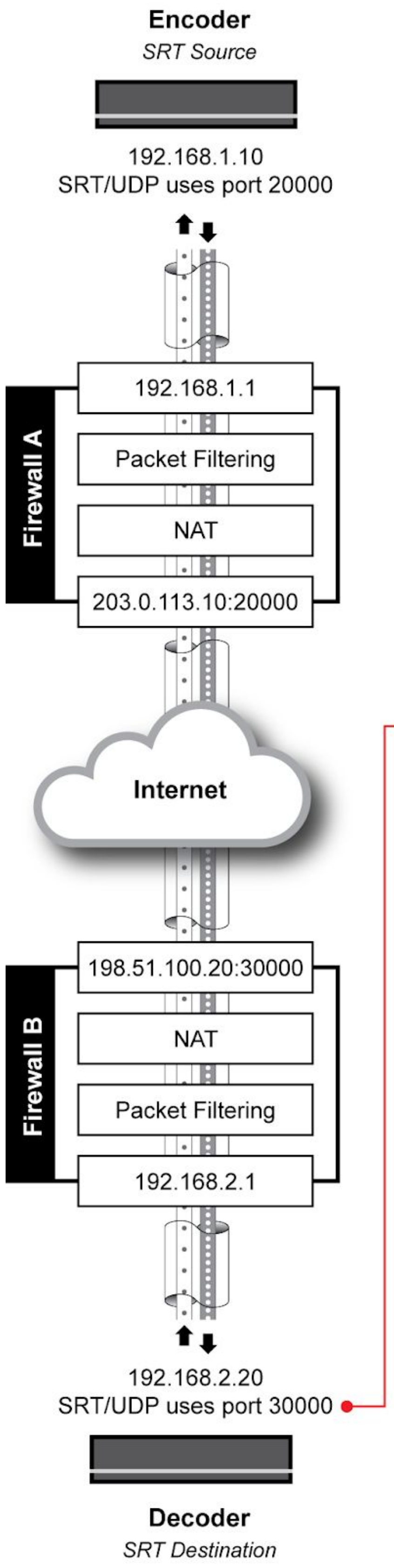
1. Using the settings in the table below, create an Input Stream on the decoder (the SRT destination device):

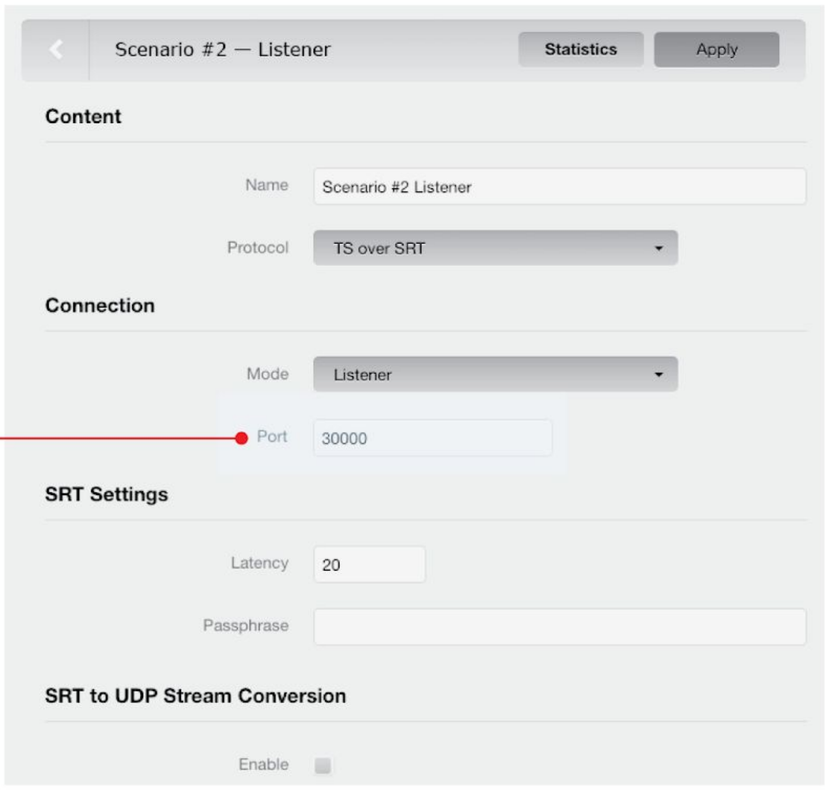
Setting	Example	Description
Mode	Listener	The decoder will wait for the source device to initiate the SRT session.
Destination Port	30000	This is the port on which the decoder will be listening (the port to which Firewall B will be forwarding SRT packets). If you have a NAT translation rule on Firewall B, the Destination Port is the port to which the rule will be forwarding packets.

NOTE: *In the figure below, we are using the same port assignments (20000) for both Caller and its firewall to simplify the scenario. The Caller could, in fact, be using any other port, as long as its firewall had the appropriate mapping to allow return traffic back to the encoder.*

Once all settings have been applied, the encoder and Decoder will handshake and establish an SRT session. The encoder will send the video stream to the decoder, which will process the stream and return control packets that include network throughput, latency and other statistics. The encoder can use this information to adapt its transmission (resend lost packets, adjust bit rate, etc.).

Note that when the SRT handshake is completed both source and destination continue to exchange control packets. Once an SRT connection/session is established, the *Caller* or *Listener* designation becomes unimportant. What matters is the source/destination relationship, or video flow, which is decoupled from the caller/listener relationship.





Scenario #2 — Listener

Content

Name	Scenario #2 Listener
Protocol	TS over SRT

Connection

Mode	Listener
Port	30000

SRT Settings

Latency	20
Passphrase	

SRT to UDP Stream Conversion

Enable	☐

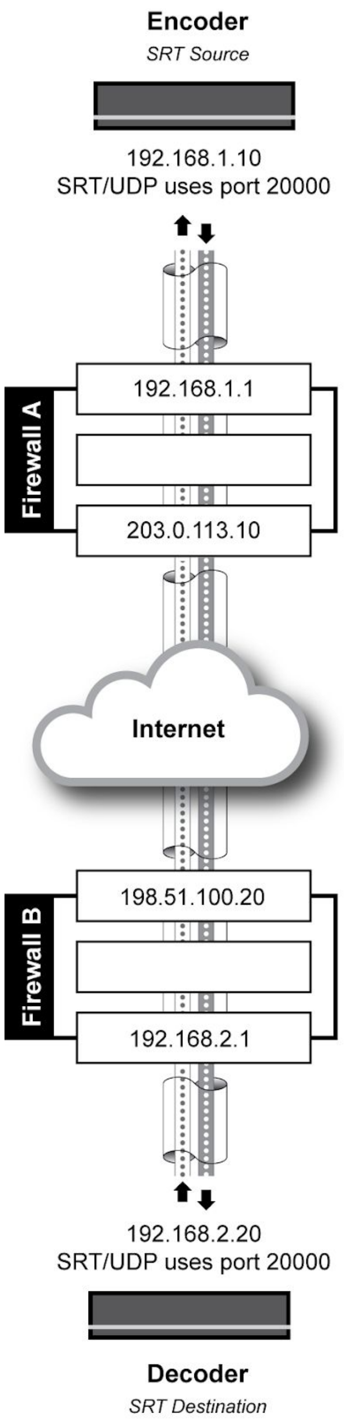
Firewall Notes

- If the source device’s port is auto-assigned, the firewall at the source must have an outbound NAT rule for [source port] set to “any”.
- If the source device’s port is specified, then the same value should be used in the outbound NAT rule.
- If a destination firewall has a filtering rule that matches a source port with a source IP, you must disable outbound port rewrite on the source firewall. Disabling this option allows the source firewall to map any port from the SRT source device to a unique, predefined port after the NAT rules have been applied.
- Depending on your firewall, the NAT rules may be applied before or after the packet filtering rules. This will affect the filtering rule definition. If the NAT rules are applied before, you have to specify the firewall’s internal IP address. If the NAT rules are applied after, you have to specify the firewall’s public IP address.

IMPORTANT: *Point-to-point sessions through firewalls can be done in reverse, with the SRT source device in Listener mode and the SRT destination device in Caller mode.*

Scenario 3: Internet streaming using Rendezvous mode

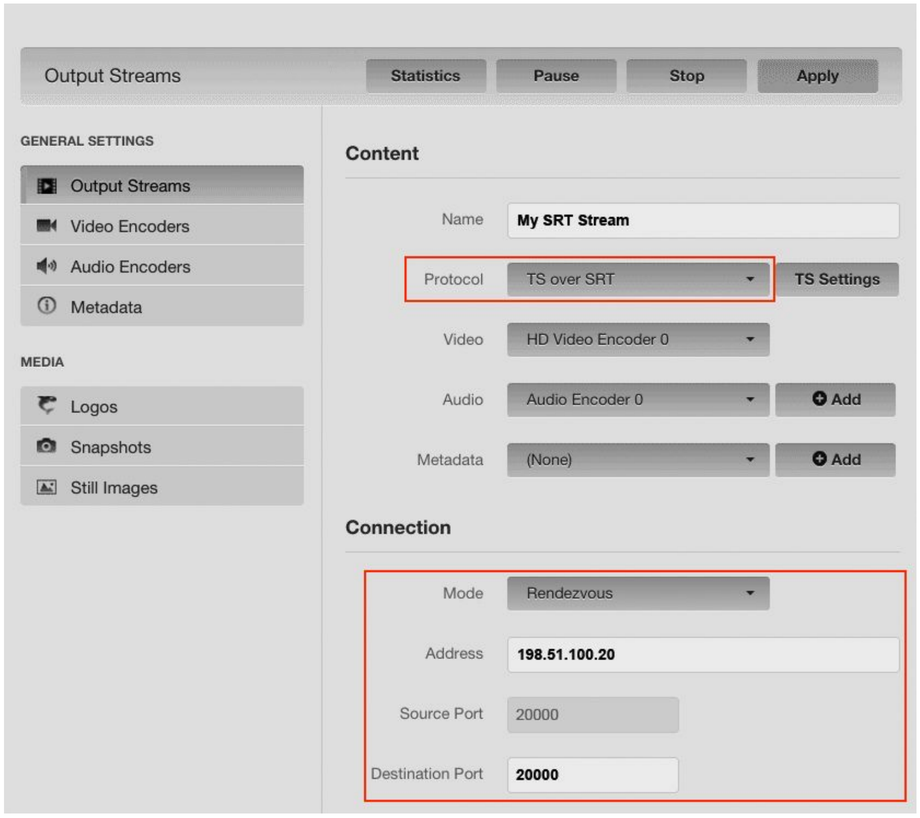
In this scenario, an encoder in Rendezvous mode behind a firewall and a decoder, also in Rendezvous mode and behind a firewall, mutually establish a point-to- point SRT streaming session over the Internet.



Step 1 — Configure the encoder

1. Using the settings in the table below, create and start an Output Stream on the encoder (the SRT source device):

Setting	Example	Description
Protocol	TS over SRT	SRT is based on the UDP protocol.
Mode	Rendezvous	Encoder attempts to initiate SRT session, and listens for incoming SRT connection requests.
Address	198.51.100.20	This is the target address for the SRT stream, which is the public IP address of the firewall at the destination.
Source Port	20000	For Rendezvous mode, source and destination ports are the same.
Destination Port	20000	This is the port over which the decoder will be listening



Content

Name: My SRT Stream
Protocol: TS over SRT
Video: HD Video Encoder 0
Audio: Audio Encoder 0
Metadata: (None)

Connection

Mode: Rendezvous
Address: 198.51.100.20
Source Port: 20000
Destination Port: 20000

Step 2 — Configure the firewall(s)

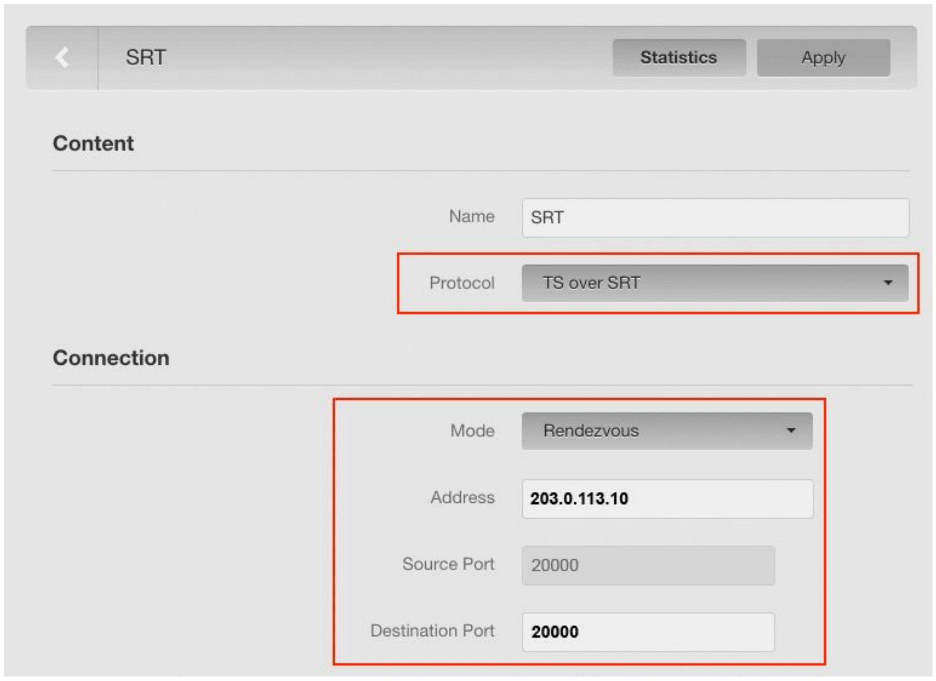
Make sure that the firewalls between the source and destination devices have port rewriting turned off (i.e. static port mapping must be allowed). In this scenario, for example, this is necessary to allow the encoder and decoder to establish an SRT session over port 20000.

Step 3 — Configure the decoder

On the decoder (the SRT destination device), do the following:

1. Using the settings in the table below, create an input stream:

Setting	Example	Description
Protocol	TS over SRT	SRT is based on the UDP protocol.
Mode	Rendezvous	Decoder will attempt to initiate the SRT session, and listen for incoming SRT connection requests.
Address	203.0.113.10	This is the public IP address of the firewall at the source.
Source Port	20000	In Rendezvous mode, the source port must be the same as the destination port.
Destination Port	20000	The port on which the encoder will be listening.



Content

Name: SRT

Protocol: TS over SRT

Connection

Mode: Rendezvous

Address: 203.0.113.10

Source Port: 20000

Destination Port: 20000

Once all settings have been applied, the encoder and decoder will handshake and establish an SRT session. The encoder will send the video stream to the decoder, which will process the stream and return control packets that include network throughput, latency and other statistics. The encoder can use this information to adapt its transmission (resend lost packets, adjust bit rate, etc.).

Rendezvous Mode and Firewalls

The *Rendezvous* connection mode allows SRT traffic between the source and destination to traverse a firewall without the need for an IT administrator to open a port, as long as the firewalls are stateful.

Even if there is no rule to explicitly allow an SRT destination device to communicate with the outside world, its control packets will nonetheless be able to return to the SRT source device because of the connection tracking feature of stateful firewalls. *Rendezvous* mode uses this behavior to create "holes" through both firewalls.

Connection Tracking

Stateful firewalls maintain a connection tracking table, which is dynamically built based on actual traffic passing through the firewall.

In a connection tracking table a typical entry might consist of UDP traffic from a device with a source IP and port number, which is converted by NAT, and then connected to the public IP of a firewall on that destination port:

Protocol	Source → Router → Destination
UDP	`192.168.1.10:20000 --> 203.0.113.10:20000 --> 198.51.100.20:20000`

If you have a call coming in from the other endpoint, then you would see the same entry in reverse in the connection tracking table:

Protocol	Source → Router → Destination
UDP	`198.51.100.20:20000 --> 203.0.113.10:20000 --> 192.168.1.10:20000`

Typically, an encoder would start an SRT session in *Caller* mode, with a decoder in *Listener* mode waiting for control packets. In *Rendezvous*, both are sending control packets to initiate the connection. The outgoing packets create an entry in the table. When the incoming packets arrive, they create a complementary entry. This tricks the firewall into thinking that the inbound packets are the responses to the outbound ones, and so it permits the packets to pass through for the duration of the streaming session.

Rendezvous mode allows the source and destination devices to "punch out" holes from the inside of their respective firewalls. The only conditions are that both devices must be in *Rendezvous* mode, both must be using the same port number, and there must be an outbound entry set up on each firewall so that the source port number is preserved (i.e. so that there is no need to create input entry rules).

Other Considerations

- It is important that the port number used is unique. Using a firewall's option to accept any (ephemeral) port may be problematic in a scenario where there are multiple encoders sending out streams.
- If the firewall at the source is set to allow the source device to use any (ephemeral) port, then outbound port rewriting must be disabled for *Rendezvous* mode. The port assignment must be static so the SRT participants can meet.
- If either the SRT source or destination device is behind a firewall that does not allow outgoing traffic, you will need to configure the firewall to allow outgoing SRT/UDP traffic.
- There are still occasions where it would be necessary (or better) to manually configure *Caller* and *Listener* settings to traverse firewalls at each end. For example, in high security environments that employ "egress filtering", firewalls are often configured to prevent arbitrary outbound ports from being assigned. An internal address cannot send to a random outbound port. In such cases, both the source and destination devices would be registered at the firewall, which would block everything else.

Configuring SRT Streams

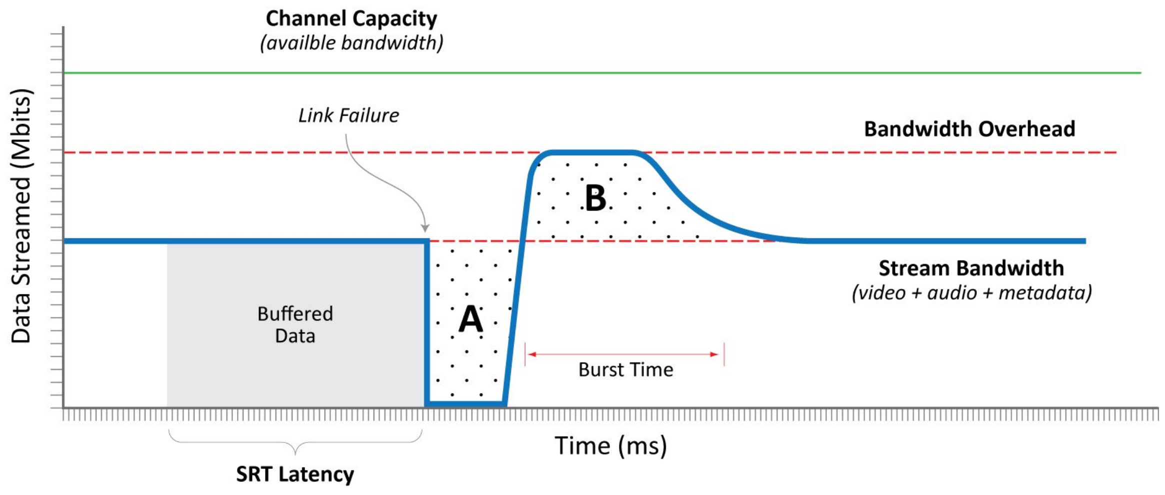
This section describes how to configure and tune an SRT stream. For complete details on how to configure a stream, please refer to the User's Guide for your device.

NOTE: *The SRT open source project does not contain any web UI elements. For illustrative purposes, this section contains screenshots from Haivsion's SRT-enabled products.*

Background

Along with the standard parameters associated with any streaming output, there are other important values you must specify for an SRT stream. To introduce you to these parameters, let's take a look at a hypothetical example of an SRT stream being sent to a destination device, and see what happens over time.

The diagram below depicts a stream being sent over a channel of some kind, such as a LAN or Internet connection, with a certain capacity. Packets being sent from a source are stored at the destination in a buffer. Let's say at some point there is a total link failure, and then, shortly after, the connection is re-established. So, for a short period of time, the destination is receiving no data.



SRT deals with such situations in two ways. The first is that the destination relies on its buffer to maintain the stream output at its end. The size of this buffer is determined by the SRT Latency setting. Once the link is re-established, the source is able to resume sending packets, including re-sending the packets lost during the link failure. To handle this extra "burst" of packets, an SRT stream allows for a certain amount of overhead. This Bandwidth Overhead is calculated such that, in a worst case scenario, the source can deliver the number of packets "lost" during the link failure (area A) over a "burst time" (area B), where area B must be equal to area A. The maximum time period for which a burst of lost packets can be sustained without causing an artifact is:

```
SRT Latency (ms) * Bandwidth Overhead (%) ÷ 100
```

Example:

```
200 ms * 20% ÷ 100 = 40 ms
```

Configuring an SRT Stream

With your source and destination devices set up (including having established call modes and any firewall settings), follow these steps to configure an SRT stream:

1. Measure the Round Trip Time (RTT) using the `ping` command.
    - If `ping` does not work or is not available, set up a test SRT stream and use the RTT value from the Statistics[2] page.
    - If the RTT is <= 20 ms, then use 20 ms for the RTT value. This is because SRT does not respond to events on time scales shorter than 20 ms.
2. Measure the Packet Loss Rate.
    - A channel's Packet Loss Rate drives the SRT Latency and Bandwidth Overhead calculations. This loss rate can be extracted from `iperf` statistics.
    - If using `iperf` is not possible, set up a test SRT stream, and then use the resent bytes / sent bytes (as reported on the SRT stream's Statistics page) over a 60 second period to calculate the Packet Loss Rate as follows:

      ```
      Packet Loss Rate = Resent Bytes ÷ Sent Bytes * 100
      ```
3. Measure the nominal Channel Capacity available to the SRT stream using the `iperf` utility.
    - If `iperf` does not work or is not available, set up a test SRT stream and use the Max Bandwidth or Path Max Bandwidth value from the Statistics page.
4. Using the following table*, find the RTT Multiplier and Bandwidth Overhead[3] values that correspond to your measured Packet Loss Rate:

Worst Case Loss Rate (%)	RTT Multiplier	Bandwidth Overhead (%)	Minimum SRT Latency (for RTT <= 20 ms)
<= 1	3	33	60
<= 3	4	25	80
<= 7	5	20	100
<= 10	6	17	120

* This table takes into account constant loss and burst loss.
See the section below on "Packet Loss Rate" for more information.

[2] While SRT accumulates a variety of statistics, it is the responsibility of each device manufacturer to expose them in some way, such as via a web page.

[3] The data in this table are derived from a mathematical model of SRT behaviour with respect to the two possible types of packet loss: *periodic* and *burst*. The net effect is that while the RTT multiplier used to determine latency will continually rise, the minimum bandwidth overhead percentage actually decreases at first (as longer latency values allow for recovery from single RTT burst losses), but then increases as heavy network congestion begins to overwhelm SRT's ability to handle continuous packet loss.

5. Determine your SRT Latency value using the following formula:

   `SRT Latency = RTT Multiplier * RTT`

   - If RTT < 20, use the Minimum SRT Latency value in the table above.

6. Determine the stream bitrate.

   - The stream bitrate is the sum of the video, audio and metadata essence bit rates, plus an SRT protocol overhead. It has to respect the following constraint:

     `Channel Capacity > SRT Stream Bandwidth * (100 + Bandwidth Overhead) ÷ 100`

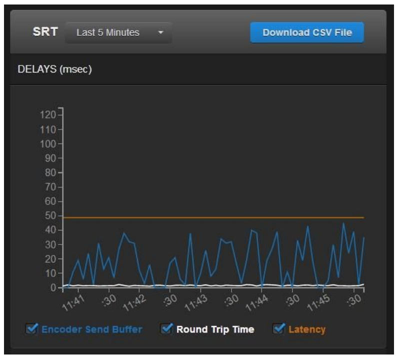
     If this is not respected, then the video/audio/metadata bitrate on the encoder must be reduced until it is respected.

   - It is recommended that a significant amount of headroom be added to cushion against varying channel capacity, so a more conservative constraint would be:

     `0.75 * Channel Capacity > SRT Stream Bandwidth * (100 + Bandwidth Overhead) ÷ 100`

7. Determine if the SRT stream has been set up correctly.

   - The best way to determine this is to set up a test SRT stream and look at the SRT Send Buffer graph on the Statistics page of the source device. The send buffer value should never exceed the SRT Latency boundary. If the two plot lines are close, increase the SRT Latency.



SRT Parameters

This section describes the various parameters that have an effect on an SRT stream's performance.

Round Trip Time

Round Trip Time (RTT) is the time it takes for a packet to travel from a source to a destination and back again. It provides an indication of the distance (indirectly, the number of hops) between endpoints on a network. Between two SRT devices on the same fast switch on a LAN, the RTT should be almost 0. Within the Continental US, RTT over the Internet can vary depending on the link and distance, but can be in the 60 to 100 ms range. Transoceanic RTT can be 60–200 ms or more, depending on the route.

RTT is used as a guide when configuring Bandwidth Overhead and Latency. To find the RTT between two devices, you can use the `ping` command. For example:

```
ping 198.51.100.20
56 data bytes 64 bytes from 198.51.100.20: seq=1 ttl=64
time=6.633 ms
```

In this example, the Response, or RTT, is 6.633 ms. You may also find the RTT for an active SRT streaming session displayed on a Statistics page.

RTT Multiplier

The RTT Multiplier is a value used in the calculation of SRT Latency. It reflects the relationship between the degree of congestion on a network and the Round Trip Time. As network congestion increases, the rate of exchange of SRT control packets (as well as retransmission of lost packets) also increases. Each of these exchanges is limited by the RTT for that channel, and so to compensate, SRT Latency must be increased. The factor that determines this increase is the RTT Multiplier, such that:

```
SRT Latency = RTT Multiplier * RTT
```

The RTT Multiplier is an indication of the maximum number of times SRT will try to resend a packet before giving up.

Packet Loss Rate

Packet Loss Rate is a measure of network congestion, expressed as a percentage of packets lost with respect to packets sent. Constant loss refers to the condition where a channel is losing packets at a constant rate. In such cases, the SRT overhead is lower bound limited, such that:

```
Minimum Bandwidth Overhead = 1.65 * Packet Loss Rate
```

Burst loss refers to the condition where a channel is losing multiple consecutive packets, up to the equivalent of the contents of the SRT latency buffer. In such cases, the SRT overhead is lower bound limited, such that:

```
Minimum Bandwidth Overhead = 100 ÷ RTT Multiplier
```

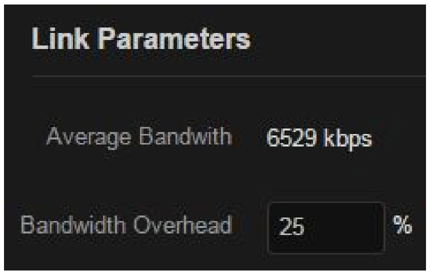
Burst losses that last longer than the SRT Latency will result in stream artifacts. SRT Latency should always be set to a value above your worst case burst loss period.

Bandwidth Overhead

The control packets associated with an SRT stream do, of course, take up some of the available bandwidth, as do any media packet retransmissions. When configuring an SRT stream, you will need to specify a Bandwidth Overhead value to allow for this important factor. The portion of audio and video content in the stream is determined by their respective bit rate settings, which are configured on the audio and video encoders themselves. SRT Bandwidth Overhead is calculated as a percentage of the A/V bit rate, such that the sum of the two represents a threshold bit rate, which is the maximum bandwidth the SRT stream is expected to use.

The SRT Bandwidth Overhead is a percentage you assign, based in part on the quality of the network over which you will be streaming. Noisier networks will require exchanging more control packets, as well as resending media packets, and therefore a higher percentage value.

NOTE: *SRT Bandwidth Overhead should not exceed 50%. The default value is 25%.*



← The **Average Bandwidth** is calculated based on the video and audio encoder settings.

← The **Bandwidth Overhead** is the percentage of the **Average Bandwidth** added to accommodate SRT controls.

Sample Bandwidth Overhead Calculation

Let's say you are streaming video at a bit rate of 1000 kbps, and audio at 128 kbps. This gives a total of 1128 kbps, which we will round up to 1200 kbps to account for any metadata and other ancillary data. This is the Average Bandwidth, which is calculated automatically based on your actual output settings. If you accept the default Bandwidth Overhead setting of 25%, then the total bandwidth reserved for the SRT stream will be:

```
    1200 + (25% * 1200) = 1500 kbps (1.5 Mbps)
```

This is the maximum bandwidth SRT will use. If there is no loss, only a slight overhead for control is used. As long as this total SRT bandwidth is less than or equal to the bandwidth available between the SRT source and destination devices, the stream should flow from one to the other without incident.

Latency

There is a time delay associated with sending packets over a (usually unpredictable) network. Because of this delay, an SRT source device has to queue up the packets it sends in a buffer to make sure they are available for transmission and re-transmission. At the other end, an SRT destination device has to maintain its own buffer to store the incoming packets (which may come in any order) to make sure it has the right packets in the right sequence for decoding and playback. SRT Latency is a fixed value (from 80 to 8000 ms) representing the maximum buffer size available for managing SRT packets.

An SRT source device's buffers contain unacknowledged stream packets (those whose reception has not been confirmed by the destination device).

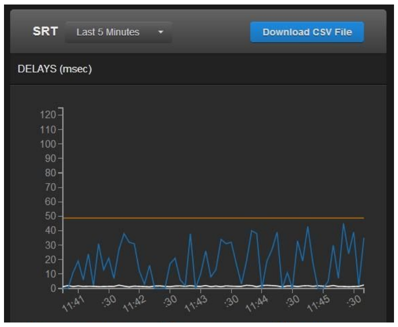
An SRT destination device's buffers contain stream packets that have been received and are waiting to be handed off (e.g. to a decoding module).

The SRT Latency should be set so that the contents of the source device buffer (measured in msecs) remain, on average, below that value, while ensuring that the destination device buffer never gets close to zero.

The value used for SRT Latency is based on the characteristics of the current link. On a fairly good network (0.1 to 0.2% loss), a “rule of thumb” for this value would be four times the RTT. In general, the formula for calculating Latency is:

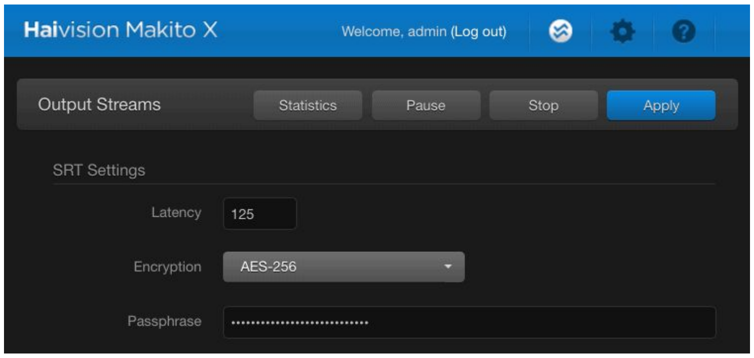
```
SRT Latency = RTT Multiplier * RTT
```

SRT Latency can be set on both the SRT source and destination devices. The higher of the two values is used for the SRT stream.



Encrypting SRT Streams

SRT streams can be encrypted using AES cryptographic algorithms, and decrypted at their destination. To implement encryption on an SRT stream, you must specify the type of encryption on the source device, and then a passphrase on both source and destination.



Latency 125

Encryption AES-256

Passphrase ••••••••••••••••••••••••••••••••

If encryption is enabled, the Passphrase entered on source and destination devices must match.

Parameter	Description
Encryption	This parameter specifies the AES (Advanced Encryption Standard) encryption key length and cipher. Range: AES-128, AES-256
Passphrase	This specifies a string used to generate the AES encryption key wrapper via a one-way function such that the encryption key wrapper used cannot be guessed from knowledge of the password. Range: 10 to 79 UTF-8 printable characters

IMPORTANT: *In SRT the passphrase is used for the exchange of encryption keys, not authentication. Setting a passphrase on the receiver does not prevent it from accepting unencrypted streams.*

Optimizing SRT Performance

SRT source and destination devices exchange a variety of information about network conditions and packet transfer, which allows them to negotiate the best possible delivery of the main audio/video content.

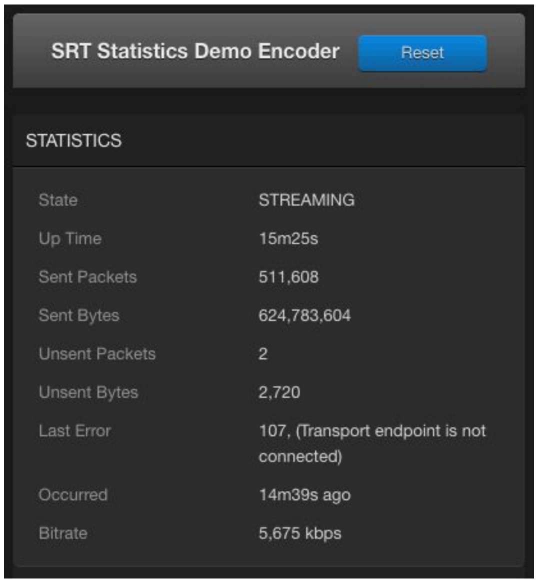
NOTE: *The open source SRT project contains all the underlying functionality necessary to create a graphical representation of this exchange, which could then be made available from the web interface of any SRT device. Monitoring and understanding the statistics data can help you to tune and optimize your SRT streaming performance.*

For illustrative purposes, what follows are descriptions of the various sections of a Statistics page using images from an SRT exchange between a Makito X Encoder (source) and Makito X Decoder (destination). These are intended to show how, by comparing the related sections between SRT source and destination devices, you can determine performance characteristics and identify opportunities to optimize the stream.

Statistics

The Makito X Statistics section provides an overview of an SRT stream's condition: its current state, the packets and bytes sent and received, the current bit rate, and (in the case of the source device), how long the stream has been active.

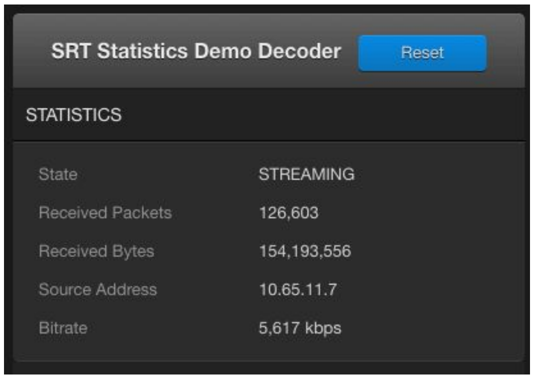
Encoder (SRT source device)



STATISTICS

State	STREAMING
Up Time	15m25s
Sent Packets	511,608
Sent Bytes	624,783,604
Unsent Packets	2
Unsent Bytes	2,720
Last Error	107, (Transport endpoint is not connected)
Occurred	14m39s ago
Bitrate	5,675 kbps

Decoder (SRT destination device)



STATISTICS

State	STREAMING
Received Packets	126,603
Received Bytes	154,193,556
Source Address	10.65.11.7
Bitrate	5,617 kbps

Src	Dest	Parameter	Description
✓		State	The current operating status of the stream (STREAMING, STOPPED, CONNECTING, LISTENING, SECURING, SCRAMBLED, or PAUSED)

✓		Up Time	The length of time the stream is actively streaming (e.g., 1d22h5m41s); only available when State is STREAMING.
✓		Sent Packets	Number of UDP packets sent for that stream.
✓		Sent Bytes	Number of bytes sent for that stream.
✓		Unsent Packets	Number of UDP packets not sent for that stream (displayed only when not 0).
✓		Unsent Bytes	Number of bytes not sent for that stream (displayed only when not 0).
	✓	Received Packets	Number of UDP packets received for that stream.
	✓	Received Bytes	Number of bytes received for that stream.
✓		Last Error	The last error reported by the SRT subsystem. All errors are recorded in a log file (see "SRT Logs").
✓		Occurred	The time elapsed since Last Error was reported
	✓	Source Address	IP address of the SRT source device
✓	✓	Bitrate	The stream bitrate (in kbps).
✓	✓	Reset	Click to reset the Statistics page.

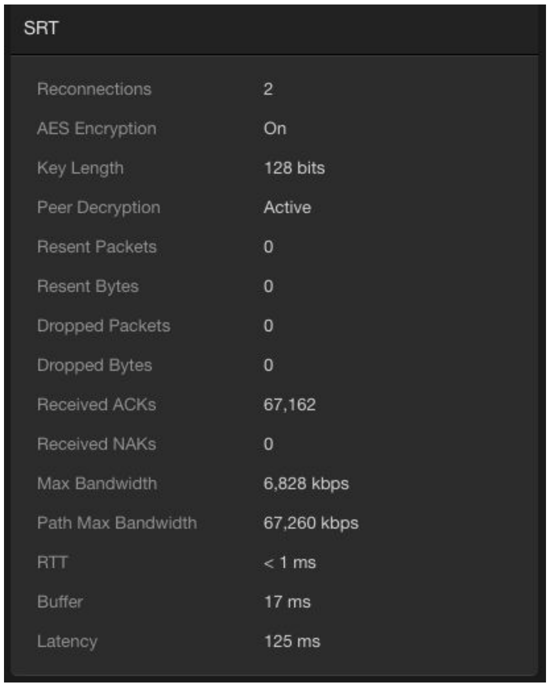
What to look for:

- Make sure the connection state is not SCRAMBLED or STOPPED.

SRT

The SRT section provides a detailed look at the SRT traffic.

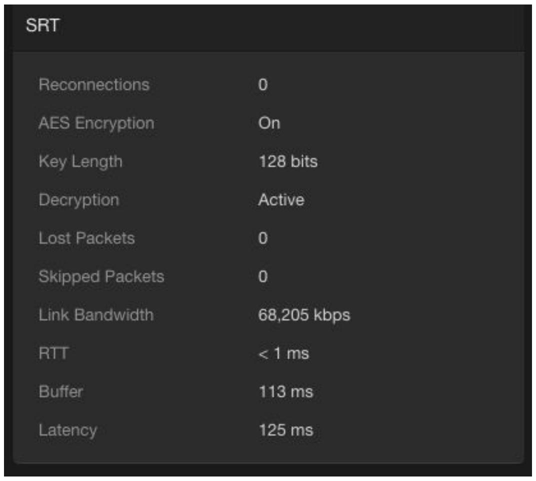
Encoder (SRT source device)



SRT

Reconnections	2
AES Encryption	On
Key Length	128 bits
Peer Decryption	Active
Resent Packets	0
Resent Bytes	0
Dropped Packets	0
Dropped Bytes	0
Received ACKs	67,162
Received NAKs	0
Max Bandwidth	6,828 kbps
Path Max Bandwidth	67,260 kbps
RTT	< 1 ms
Buffer	17 ms
Latency	125 ms

Decoder (SRT destination device)



SRT

Reconnections	0
AES Encryption	On
Key Length	128 bits
Decryption	Active
Lost Packets	0
Skipped Packets	0
Link Bandwidth	68,205 kbps
RTT	< 1 ms
Buffer	113 ms
Latency	125 ms

Src	Dest	Parameter	Description
✓	✓	Reconnections	Number of reconnections since the stream started. Severe network congestion may cause the connection to drop, but it will be automatically reconnected if less than ~1 second.
✓	✓	AES Encryption	Indicates status of AES Encryption (if enabled).
✓	✓	Key Length	Indicates the specified key length for AES encryption.
✓		Peer Decryption	Indicates whether or not the SRT stream is being successfully decrypted.
	✓	Decryption	Indicates whether or not the SRT stream is being successfully decrypted.
✓		Resent Packets	Number of packets retransmitted based on reports from the destination device (or lack thereof).
✓		Resent Bytes	Total bytes retransmitted.
✓		Dropped Packets	Number of packets reported missing by the destination device. This is the raw number of packets dropped by the network. These packets may be recovered by

			retransmission by the source device, and so do not necessarily result in any video artifacts.
✓		Dropped Bytes	Number of dropped bytes.
	✓	Lost Packets	Number of packets reported missing by the decoder.
	✓	Skipped Packets	Packets that have arrived at the destination device too late, or that never arrive at all. If the “time to play” for a packet has passed, and it is either not at the decoder or arrives after the content it is associated with has already played, that packet is reported as “skipped” on the destination device. Usually this results in some type of video artifact (a replayed frame or video blocking).
✓		Received ACKs	Total number of acknowledgment packets received from the destination device (this is a measure of transmission progress).
✓		Received NAKs	Total number of negative acknowledgment packets received from the destination device.
	✓	Link Bandwidth	Estimated maximum bandwidth available as viewed from the destination device.
✓		Max Bandwidth	Maximum bandwidth used by the source device for this SRT stream (i.e. the current total of audio/video bit rate plus ancillary data plus the SRT Bandwidth Overhead).
✓		Path Max Bandwidth	The same value as Link Bandwidth. The destination device sends the value to the source device with an acknowledgment packet.
✓	✓	RTT	Round Trip Time (RTT) is the time it would take for a packet to travel from a specific source to a specific destination and back again. In SRT, this is measured as the time it takes for the destination device to send an acknowledgment (ACK) packet, and then receive a corresponding confirmation (ACKACK) packet.
✓	✓	Buffer	Size of an SRT buffer (in milliseconds). The SRT source device’s buffer contains unacknowledged stream packets (those whose reception has not been confirmed by the destination device). The size of the source device buffer, in the absence of congestion or packet loss, is around the RTT value. In the presence of recoverable packet loss, the value should be between those for RTT and Latency. The SRT destination device’s buffer contains stream packets received and waiting to be forwarded or decoded. This statistic shows the portion of the

			destination device's buffer up to the first missing packet (i.e. the time remaining for the retransmission of the missing packet before it's too late). The value of the destination device buffer in the absence of packet loss is just below the latency value. In the presence of packet loss, it is between 0 and the latency value. Note that if you change the bit rate from 6 to 2 Mb/s, the byte count will change but a 2 second buffer will remain 2 seconds.
✓	✓	Latency	A fixed value (from 80 to 8000 ms) representing the maximum buffer size available for managing SRT packets. Values can be entered on both the source and destination devices. When the handshake occurs at the initiation of an SRT streaming session, the higher of the two values is implemented on both devices. The destination default is set to the minimum (80 ms) so that the value can be controlled from the source. The default latency value on the source is 125 ms.

What to look for:

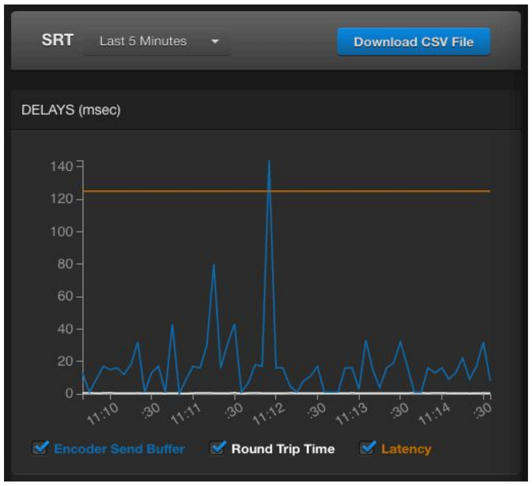
- If you start to see lots of reconnections, this is an indication that there is a problem with the communication channel between devices.
- A steadily increasing number of NAKs indicates a problem.
- It's normal to see a few dropped packets, but there should not be too many. A low number of dropped packets indicates you are using your bandwidth optimally.
- If the number of Skipped Packets increments slowly, the best thing to do is increase the SRT Latency. If it increments in large jumps, the best thing to do is to lower the video bitrate, or to increase the Bandwidth Overhead % if you have available capacity.

TIP: *When adjusting the SRT parameters, ignore any spikes or other variations that appear on the Statistics charts at the time of the change.*

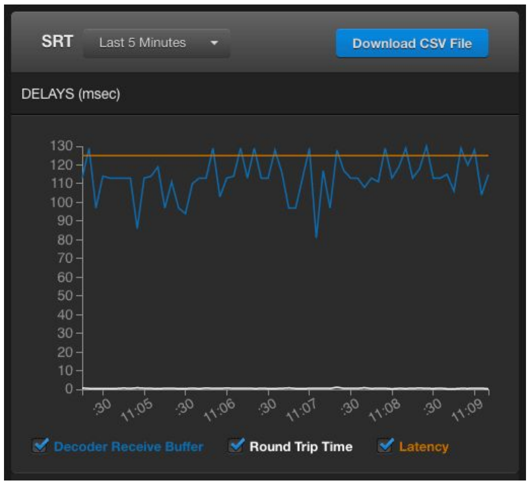
Delays

The Delays section provides a graphical overview of how the SRT stream buffers are handling the traffic.

Encoder (SRT source device)



Decoder (SRT destination device)



Src	Dest	Parameter	Description
✓	✓	Last X Minutes	Choose a time frame (5 minutes, 60 minutes, or 24 hours) for the data to be displayed in the DELAYS and BANDWIDTH USED graphs.
✓	✓	Download CSV file	Click to download a CSV file.
✓		Encoder Send Buffer	Check this box to view a plot of the contents (in ms) of the encoder buffer over time (blue line in the graph).
	✓	Decoder Receive Buffer	Check this box to view a plot of the contents (in ms) of the decoder buffer over time (blue line in the graph).
✓	✓	Round Trip Time	Check this box to view a plot of the RTT value over time (white line in the graph).
✓	✓	Latency	Check this box to view a plot of the Latency value over time (orange line in the graph).

What to look for:

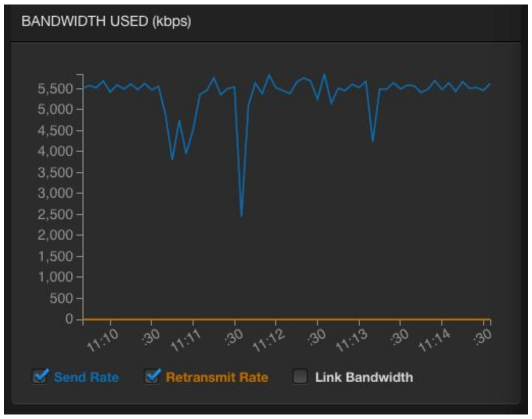
- The value for Buffer (blue line) on the source device should normally not exceed the Latency value (orange line). If the Encoder Send Buffer spikes above the latency line, then increase the SRT Latency. You may see occasional spikes (with corresponding anomalies in the display). But if these are few and far between, and the impact on the display of the stream is tolerable, you may choose to ignore these spikes as you tune the stream parameters.

- Ideally, the destination device buffer level should be just below the latency value. If the Decoder Receive Buffer goes to 0 often, then there is most likely insufficient BW to support the desired bitrate (which should therefore be lowered). If the Decoder Receive Buffer only occasionally goes to 0, then the SRT Latency should be increased.
- On the source device, if the Encoder Send Buffer often reaches or exceeds the Latency value, then there is most likely insufficient bandwidth to support the desired bitrate (which should therefore be lowered). If the Encoder Send Buffer only occasionally goes to or above the Latency value, then the SRT Latency should be increased.

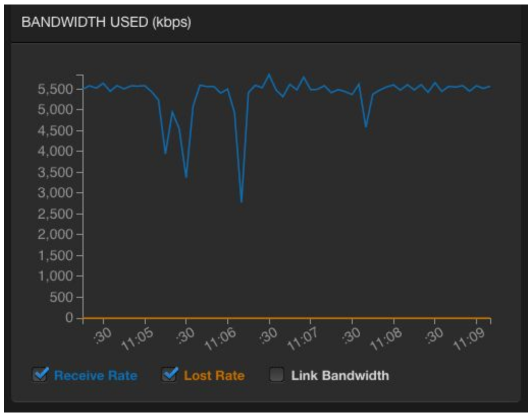
Bandwidth Used

The Bandwidth Used section provides a graphical overview of how the network bandwidth available to the SRT stream is actually being used, in terms of the rate at which packets are sent, received, resent and lost, as well as the data volume (rate × time).

Encoder (SRT source device)



Decoder (SRT destination device)



Src	Dest	Parameter	Description
✓		Send Rate	Check this box to view a plot of the SRT source device's outbound bitrate over time (blue line in the graph).
	✓	Receive Rate	Check this box to view a plot of the SRT destination device's inbound bitrate over time (blue line in the graph).
✓		Retransmit Rate	Check this box to view a plot of the rate at which the SRT source device is resending packets (orange line in the graph).
	✓	Lost Rate	Check this box to view a plot of the rate at which SRT source device is reporting lost packets (orange line in the graph).
✓	✓	Link Bandwidth	Check this box to view a plot of bandwidth available to the SRT stream (white line in the graph). This is a plot of the Link Bandwidth (or equivalent Max Path Bandwidth on the source device) shown in the SRT panel.

What to look for:

- The use of encryption has an impact on the processing resources of the source device, which may have an effect on streaming capacity overall.
- If the SRT destination device shows too many skipped or dropped frames, increase the SRT Latency, lower the video bit rate, and/or increase the Bandwidth Overhead %.

Graph Sample Rates

The information presented in the graphs on the Statistics page is based on continuous samples taken every 5 seconds over a 24 hour period (sampling does not occur during pauses or disconnects).

The data is displayed in either 60 or 120 columns (depending on the time span being displayed), where each column represents an average value based on the time span being displayed.

Graph Time Span	# Columns (Data Points) Displayed
Last 5 minutes	60
Last 60 minutes	60
Last 24 hours	120

For example, if the graph is showing data from the last 5 minutes (300 seconds), then it will contain 300 ÷ 5 sec. = 60 samples over 60 columns (this works out nicely to one sample per column).

If the graph is showing data from the last 24 hours (86400 seconds), then it will contain 86400 ÷ 5 sec. = 17280 samples over 120 columns. Each column contains the average value of 144 samples (17280 ÷ 120).

SRT Logs

Data associated with an SRT stream is continuously logged (one log per stream). The log file will contain time-stamped entries at 5 second intervals for up to a maximum of 24 hours. The sampling is non-contiguous (no log entries are made when the stream is inactive). For example, the log might contain data covering 4 days, but only for 6 hours per day.

From the Statistics page, you can download the log file in CSV format, which may be used with applications such as Microsoft Excel or third party log analysis software. It can be useful to examine logs to help with troubleshooting, or to determine how available bandwidth is being used over time, which may allow you to better control associated costs.

The table below shows a sample log with a few rows of data:

DateTime	SendRate	ReSnRate	DropRate	MaxBw	AvailBw	RTT	SendBufs	PdDelay
2018-11-09T10:54:30-0400	3479	0	0	4226	140504	74.417	72	125
2018-11-09T10:54:35-0400	3334	0	0	4226	155224	74.304	77	125
2018-11-09T10:54:40-0400	3385	0	0	4226	92273	74.514	87	125
2018-11-09T10:54:45-0400	3377	0	0	4226	66601	74.248	91	125
2018-11-09T10:54:50-0400	3493	0	0	4226	56608	74.204	81	125
2018-11-09T10:54:55-0400	3358	0	0	4226	156397	74.214	73	125
2018-11-09T10:55:00-0400	3391	0	0	4226	176659	74.338	73	125
2018-11-09T10:55:05-0400	3397	0	0	4226	302024	74.512	70	125
2018-11-09T10:55:10-0400	3460	0	0	4226	72274	74.341	97	125
2018-11-09T10:55:15-0400	3334	0	0	4226	51725	74.264	43	125

Frequently Asked Questions

If SRT is open source, does that mean it is royalty-free?

SRT was released as open source software in April 2017, under version 2.0 of the Mozilla Public License. It is royalty-free and available to everyone on GitHub.

Can SRT carry modulated signals like DVB-T and T2, or complex streams like BTS and MPTS over the internet?

SRT is completely content agnostic. It can carry any kind of stream.

What are the bandwidth limitations when using SRT?

SRT has no specific bandwidth limitation, although a bandwidth limit can be set, if required. All other limits result from the network parameters and characteristics of the devices that handle it. Keep in mind that the bandwidth overhead setting is there for re-sending traffic in parallel with new content, and that you have to take that into account with a rigidly rate constrained transmission channel. It should actually be easier with DVB-T/T2 than with a random VBR stream, but don't send a 17 Mbps payload over an 18 Mbps link with a 25% SRT overhead.

Would SRT be suitable for VSAT SCPC C-Band, SCPC Ku-Band or iDirect Ku-Band transmission? Would latency would be adversely affected?

The loss of a packet and its recovery can be described as follows:

1. A packet doesn't arrive at the receiver.
2. The next packet arrives, triggering the receiver to send back a lost packet report.
3. The sender receives the report and schedules the packet for retransmission[4].
4. The retransmitted packet reaches the receiver.

The typical time it takes for packet traversal (the single transmission time or "STT") at each step will vary for different types of networks. Normally you need just one STT unit to receive a packet. But when a packet is lost, you need three STT units plus the time interval between two consecutive packets. This aggregate time (multiplied by 2 to be safe) must be still less than the configured latency.

The RTT for VSAT links is over 540[5] ms, and can be much higher. Using a conservative value of 800 ms, we can calculate the single transmission time (STT) as RTT/2 = 400 ms. So for a lost packet you have: STT (originally sent packet) + packet delta (time to realize packet loss) + STT (send loss report) + packet delta (sender decides to retransmit) + STT (sending the lost packet again) = 3*STT plus some packet interval time (100 ms max). Multiplying this by 2 gives a

[4] When the sender receives the loss report, the lost packet is scheduled for retransmission (which has a higher priority than regular transmission) at the next possible opportunity. But this is not instantaneous. There may be a "sleep" delay as required by the bandwidth limit. Earlier regular data might already be underway in the network, or in the system network buffer. So the retransmitted packet is only sent as quickly as the application can react upon receiving the loss report.

[5] https://www.satellitetoday.com/telecom/2009/09/01/minimizing-latency-in-satellite-networks/

minimum safe latency of 2600 ms to allow for reliable packet recovery (providing that no packet gets lost twice, and no subsequent loss happens). In practical terms, it would be better to use an even higher latency value, like 4000 ms.

What about using SRT for MCPC signals?

With multiple channel per carrier (MCPC) signals, as described above for VSAT, you need to allow sufficient latency/buffer for the 4 x RTT requirement. Here is some additional guidance:

1. Use traffic shaping and idle cells to provide a CBR stream.
2. Ensure the stream MTU is below the link MTU. You may need to do a ping test with the DF flag set.
3. Leave yourself some bandwidth overhead above your SRT bandwidth, and remember that you're dealing with video bandwidth + audio bandwidth + TS muxing overhead + Ethernet/IP packet overhead + SRT resent buffering. If you have signal degradation on SAT and experience persistent packet loss, you can end up with SRT retransmissions spiking, and you never catch up.

If there's an SRT connection between two devices and the network goes down, does the SRT protocol automatically reconnect when the network comes up?

Once the stream has started the protocol works by establishing a connection with regular acknowledgement and negative acknowledgement packets between the peers. In addition, keepalive packets are exchanged as necessary to maintain the connection. For network interruptions of less than 1 second, the SRT connection may be re-established automatically. For longer interruptions, it is up to the application (not SRT) to re-establish the connection. For example, all Haivision products will try to reconnect immediately, and retry indefinitely, until a user deletes the connection from their configuration.

Does SRT encrypt a stream at the packet level?

No, SRT encryption occurs at the payload level. Only the data that are being transported are encrypted. All control information is sent in the clear (this is a security "best practice" to avoid repeated patterns that would weaken the encryption). Note that encryption does not have an impact on the SRT stream itself, but does place a burden on the CPUs of the sender and receiver, which is proportional to the number of streams and their bitrates.

Can a connection be established between devices with different versions of SRT?

All versions of SRT are backwards compatible. When establishing a connection, a newer version of SRT will "fall back" to the capabilities of the older version.

Can Pro-MPEG error recovery be used with SRT?

Pro-MPEG FEC is not used by SRT. The only error recovery mechanism in SRT is retransmission requests, in the form of NAK (Negative Acknowledgment) packets. Whenever missing packets are detected, the receiver sends NAKs with information about those packets back to the sender. The sender responds by re-sending the missing packets.

Is SRT FIPS 140-2 certified? Has it been evaluated against the FIPS 140-2 standard?

SRT uses only FIPS-140.2 approved algorithms, but it can be linked with any version of OpenSSL and other crypto libraries. If a FIPS mode does not exist in the linked library, SRT cannot enable it. Note that FIPS 140.2 does not guarantee tamper resistance across site-to-site links. For example, even if using a FIPS 140.2-validated crypto lib, SRT does not provide perfect forward secrecy.

What would be an example of a firewall policy that would cause Rendezvous mode to fail, assuming outbound ports are opened and inbound traffic from established connections is allowed? Any edge-case policies that would explicitly prohibit traffic, or architectures that might confuse the connection process?

Dynamic Port Address Translation (PAT, aka port mangling) will foil *Rendezvous* mode, as will firewalls that do not have UDP “fixup” or session tracking. Remember that UDP itself is a stateless protocol. All *Rendezvous* mode does is to force both sender and receiver to connect to each other simultaneously on the same source and destination port. The devices need to connect to each other’s public IP addresses/ports, and the firewalls have to allow that traffic through to the internal IP address/port of each device. If the firewalls are very strict or are randomizing the addresses, *Rendezvous* will not work.

Rate limiting can also cause issues on enterprise networks.

What can cause an SRT connection to terminate?

SRT connections can only get broken if the connection is timed out (no packet was sent to particular device from its peer for some period of time, about 5 seconds). As of SRT version 1.3.1, a connection timeout may also happen when the receiver buffer is not emptied by the receiver application fast enough, to prevent going into an unrecoverable state.

The continuous corruption of an input source will cause the encoder process to restart, consequently resetting the SRT connection.

Often when a connection is broken, it is simply because the socket has been closed. If the application manages errors correctly, the broken connection is reported with an appropriate error number and a clear message.

What is the underlying SRT protocol structure?

SRT is based on the User Datagram Protocol (UDP), but has its own mechanisms to ensure real-time delivery of media streams over noisy networks such as the Internet.

What ports are required to be open on a firewall?

The source and destination device UDP ports are configurable. Each stream only requires a single port. The port is user-defined and can be between 1025 and 65,535. Specific port requirements for firewalls may depend on the security policies of your organization. Consult with your network system administrator.

What is the bandwidth overhead requirement for a connection traversing the Internet?

Bandwidth (BW) Overhead specifies the maximum stream bandwidth overhead that can be used for recovery of lost packets. The default BW Overhead is 25%. Depending on your settings, SRT will either retransmit lost packets quickly (thereby using more bandwidth) or over a longer period (requiring less bandwidth but resulting in higher latency).

How does encryption affect the bandwidth?

Encryption does not affect the bandwidth. However, applying encryption is a processor-intensive task, and may have an impact on the number and bit rate of the streams an encoder is able to output.

Is SRT compatible with wireless networks?

SRT can be used over wireless networks, WiMANs (Wireless Metro Area Networks), LANs, private WANs, or the public Internet.

Does the number of "Lost Packets" increment even when an SRT retransmit is successful? Would "Lost Packets" result in visual artifacts or quality issues?

"Lost Packets" and the related "Skipped Packets" are statistics reported by an SRT decoder:

- **Lost Packets**: A hole in the packet sequence has been detected. A request to re-transmit the lost packet has been sent to the source device. This lost packet may (or may not) be recovered by the re-transmit request.
- **Skipped Packets**: The time to play the packet has arrived and the lost packet was not recovered, so the decoder will continue playing without it. A video or audio artifact may result from a skipped packet.

How can I determine the SRT bandwidth requirements from a decoder back to an encoder?

SRT back-channel control packets from the decoder to encoder take up a minimum of ~40 Kbps of bandwidth when the channel conditions are perfect. If there are lost packets on the link, then the SRT receiver will generate more signal traffic, proportional to the lost packet rate. A single lost packet will consume about 400 bps of the available bandwidth on the receiver side.

From the decoder back to the encoder, bandwidth usage will increase linearly with the packet loss rate.

How can I determine the bandwidth available between two endpoints before starting an SRT stream?

If you have established an SRT stream, you can view bandwidth information on the Statistics pages of the source and destination devices.

If no SRT stream is currently running, you can use the `iperf` bandwidth measurement utility to get bandwidth and jitter information.

NOTE: `iperf` *should never be run on a system carrying production video.*

Can I stream from one encoder to multiple decoders?

No. SRT is for point-to-point connections. It does not support multicast. If you need to have an SRT stream delivered to multiple decoders, you can use a gateway server.

I'm seeing a "sawtooth" pattern on the Statistics page for my encoder. What does that mean?

It means the decoder is not acknowledging that it has received the packets sent from the encoder. The encoder keeps the packets it has sent in its buffer until it receives a response (this takes at least the equivalent of the round trip time). If the encoder receives acknowledgments promptly from the decoder, the buffer remains relatively empty. Otherwise, the buffer gradually fills until it reaches a point where it must drop the unacknowledged packets, creating the characteristic sawtooth pattern.

This typically occurs when you don't have enough bandwidth to transmit. The buffer value will increase up to the point where it can no longer keep up, and then will drop in a classic sawtooth pattern. In such cases, try increasing the SRT overhead, or lowering your video bit rate.

When should I start to worry about a rising number of dropped packets?

If the rate is going up constantly, it means that you haven't configured enough bandwidth overhead or latency. Check to see if you are streaming too far above the latency value on the encoder.

How about skipped packets on the decoder?

Sometimes a packet, let's call it Packet B, arrives at the decoder ahead of time and sits in the queue, ready to play. But if the packet that should be played immediately before it (Packet A) arrives late (or never), the decoder skips Packet B sitting in the queue. In other words, the time to play for the associated packet has passed, and it is either not at the decoder or arrives after any associated later content has already played. Usually this means some type of video artifact also occurs (a replayed frame or video blocking artifacts).

You could think of a skipped packet as a packet that the decoder drops, except that it doesn't tell the encoder. The decoder sends a positive ACK, even though the packet is "lost" from the point of view of the decoder. It might drop an entire frame because it is corrupted, but the encoder doesn't know it. This is because when things are going badly, the last thing you want to do is to increase the overhead traffic. A packet has a certain time-to-live, and if it doesn't play before that time then it is skipped.

You might see the number of skipped packets increasing on the decoder, without a corresponding effect on the number of dropped packets seen on the encoder. If the number of skipped packets on the decoder increases slowly, you should increase the SRT Latency. If the number of skipped packets on the decoder increments in large jumps, the best thing to do is lower your video bitrate, or increase your Bandwidth Overhead % if you have available capacity.

Do I control the latency value at the source or destination?

You can control the latency, the bit rate, and the overhead percentage on the source device. But you can configure the latency on the destination device as well. The two devices will settle on the maximum value.

How do I decide whether to boost the latency or lower the bit rate?

You have to decide what is most important for you: quality or latency. If low latency is more important to you, then you may see that you are not using your bandwidth as effectively, and image quality may not be optimal. If you want more quality, you need to use the maximum available bandwidth, with minimal overhead, and therefore more latency. An SRT transmission needs either bandwidth or time. So if you are using your maximum bandwidth at a higher bit rate because you want higher quality, you'll have to allow more time to recover from faults.

What happens if I set my latency value too low?

If the delay/latency setting is too low, this will be reflected at the destination as visible artifacts, corresponding to skipped packets on the decoder.

If I see my Send Buffer repeatedly spiking by one or two seconds, by how much should I increase my latency?

In early SRT versions, there was little tolerance. The Send Buffer values had to remain below the Latency to obtain good results. In more recent SRT versions the Send Buffer can occasionally go over one second without being dropped. However, this can cause a problem at the other end, because while the packets are shown as "delivered" on the encoder, they may not arrive at the other end.

Does SRT support Pro-MPEG Forward Error Correction?

No, SRT does not support PRO-MPEG error recovery at this time. Its error recovery mechanism is based on retransmission requests, in the form of NAK (Negative Acknowledgment) packets. The destination device sends NAKs back to the source device whenever missing packets are detected. The source device responds by re-sending the packets identified in the NAKs.

What are "keep alive" packets?

To maintain an SRT connection, control packets must be sent at an interval of 10 ms (max). When a destination device receives a media packet, it acknowledges the reception by returning an "ACK" control packet. If the interval between "ACK" packets is greater than 10 ms, "keep alive" packets are sent to keep the connection open.

Unlike with TCP, if an SRT connection is broken the source device will be unaware that the destination is not receiving packets for up to several seconds before the connection is re-established.

Why do I have to specify an Outbound NAT Source Port on my source device's firewall?

Since UDP is not session based, some firewalls will randomize source UDP. Let's say your source is an encoder, with Firewall A. If you send from a random port on the Makito X to port 20000 on the destination, Firewall A may not use the same port. The assumption is UDP is not bidirectional. Firewall behaviour differs from one to the next, so this is tricky. Return traffic has to come back on the same port it was sent from. So if you sent from X.X.53.36 port 20000, the return traffic has to go back to X.X.53.36 port 20000. This is why the Outbound NAT Source Port setting exists. Some firewalls will randomize the source on the outbound interface.

In some cases, *Rendezvous* mode may be required. *Rendezvous* mode sets the port to be same on both sides; firewalls will almost always allow traffic returned on the same port. Cisco has a "UDP Fixup" setting that tells firewalls to treat UDP packets as if they are session based.

It is important to consider both network (NAT) and port (PAT) address translation. It is possible, for example, for public port 20000 on a firewall to be mapped to 2500 on an internal address. Port Address Translation (PAT), or port rewriting, can actually be made to work in cases where the mapping is fixed. However, many firewalls randomize the source port through "packet mangling."

References

Secure Reliable Transport (SRT) Protocol Technical Overview
https://github.com/Haivision/srt/files/2489142/SRT_Protocol_TechnicalOverview_DRAFT_2018-10-17.pdf

SRT White Paper
https://www.haivision.com/?s=SRT+Technical+Note&webpage%5B%5D=white_paper

SRT Streaming Protocol
http://www.haivision.com/products/secure-reliable-transport

Use SRT to distribute live streams from Wowza Streaming Engine
https://www.wowza.com/docs/how-to-use-srt-to-distribute-live-streams

Videos

SRT Alliance Videos
https://www.srtalliance.org/videos/

Makito X with SRT
https://www.haivision.com/resources/video-portal/makito-x-encoder-srt/

How to setup SRT and Wowza Streaming Engine and Makito @ IBC 2017 [1:59]
https://www.youtube.com/watch?v=ykLu8h0yp6c

Haivision and Wowza Form SRT Alliance
Open Source Low Latency Video Streaming Initiative [1:46]
https://www.youtube.com/watch?v=rJCzkXhoMEE&t=44s

Why you should learn about the SRT protocol and the SRT alliance [5:42]
https://www.youtube.com/watch?v=067v65JiCEg

SRT Protocol DEMO [2:33]
https://www.youtube.com/watch?v=I6G2iaTMuQo

Wowza — Talking Low Latency Streaming Trends and Options [26:05]
https://www.youtube.com/watch?v=SV_Nl43bXjQ&t=504s

Achieving Low-Latency Streaming At Scale [36:48]
https://www.youtube.com/watch?v=WMHdmEB_8PU

SRT explained in 75 seconds [1:14]
https://www.youtube.com/watch?v=68Ug8s4JkVs&feature=youtu.be

SRT Panel Discussion @ NAB Show 2018 [1:03:44]
https://www.youtube.com/watch?v=PuGn2cjIt04

SRT — The Fastest Growing Open Source Streaming Project [10:33]
https://www.youtube.com/watch?v=pNk_plbvy8Q

Terms & Definitions

Term	Abbreviation	Description
Bandwidth	BW	The data exchange rate (measured in bps, Kbps, or Mbps) of a communications channel, such as an SRT stream.
Bandwidth Overhead		The portion of the total bandwidth of an SRT stream that is required for the exchange of control and recovery packets. This value is usually calculated as a percentage of the base (TS) output bitrate of an encoder (video + audio + metadata + ancillary data). The sum of the base output rate plus the bandwidth overhead determines the maximum bandwidth that can be used by the SRT protocol.
Jitter		Jitter (measured in milliseconds) is the undesired deviation from true periodicity of an assumed periodic signal in electronics and telecommunications, often in relation to a reference clock source. In a computer network context, jitter is the variation in latency as measured in the variability over time of the packet latency across a network.
Local Area Network	LAN	A computer network that interconnects computers within a limited area such as an office building.
Latency		The amount of time between the capture and display of a video frame, including the time it takes to encode, transmit and decode. With SRT, latency is a parameter that can be adjusted to increase the buffers at source and destination to allow those devices to adapt to network conditions.
Network Address Translation	NAT	A functionality that remaps one IP address to another. Network Address Translation allows a device, such as a router or firewall, to redirect traffic between the Internet and a private network.
Packet Loss		Packet loss occurs when data packets fail to reach their destination. It can be caused by a number of factors including signal degradation over a network, channel congestion, corruption, faulty networking hardware, faulty network drivers, or normal routing routines, such as Dynamic Source Routing (DSR) in ad hoc networks. Packet loss is measured as a percentage of packets lost with respect to packets sent.
Packet Delay Variation	PDV	Packet Delay Variation is the difference in the end-to-end, one-way delay between selected packets

		in a flow, with any lost packets being ignored. [RFC 3393]
Round-trip Time	RTT	Also called round-trip delay, RTT (measured in milliseconds) is the time required for a packet to travel from a specific source to a specific destination and back again.
Secure Reliable Transport	SRT	A transport technology developed by Haivision that optimizes streaming performance across unpredictable networks such as the Internet.
Virtual Private Network	VPN	A mechanism for connecting to a private network over a public network like the Internet. Most commonly, a VPN is used to provide a secure, encrypted tunnel through which a remote (authorized) user can access a company network.

SRT Alliance Membership

The SRT Alliance was founded by Haivision and Wowza. Its mission is to overcome the challenges of low-latency video streaming, thereby changing the way the world streams. Fundamental to this mission is the support of a freely available open-source video transport protocol and technology stack called SRT (Secure Reliable Transport) that will accelerate innovation through collaborative development. It is called the SRT Open Source Project. The SRT Alliance will promote the industry-wide recognition and adoption of SRT as a common standard for all low-latency internet streaming. The SRT Alliance is open to new members. For companies who want to participate actively in growing the ecosystem of SRT or simply join the alliance to endorse SRT technology, please complete the SRT Alliance membership application here.

Haivision

Haivision provides media management and video streaming solutions that help the world's leading organizations communicate, collaborate and educate.

WOWZA™ media systems

Wowza Media Systems™ is the recognized gold standard of streaming, that enables organizations to expand their reach and more deeply engage their audiences on any device, anywhere in the world.