HAIVISION

Live Streaming using SRT with QUIC Datagrams

Maxim Sharabayko, Ph.D., Principal Research Engineer

Maria Sharabayko, Ph.D., Principal Data Scientist

John Leipper | Principal Solutions Architect

08.05.2023

What is SRT?

- A sub-second latency live contribution protocol on top of UDP (unicast)
- Stream multiplexing
- Bidirectional transmission
- Packet loss recovery (ARQ and/or FEC) within a fixed end-to-end latency constraint
- Connection bonding (or path redundancy)
- Content agnostic
- An open-source library is available on GitHub

SRT SECURE RELIABLE TRANSPORT

*Enabling **low-latency video** contribution & distribution and **fast file transfer** over unpredictable networks*

SRT ALLIANCE SECURE RELIABLE TRANSPORT

More info: SRT Protocol Overview (SVA 2020)
https://www.youtube.com/watch?v=MFJeyInLKZY

SRT Alliance is More than 600 Members Now!

Webinar: Tuesday, May 9th at 10am ET
Plugfest: The whole week

[Mark Your Calendars for the Next SRT InterOp Plugfest with YouTube]

- DATAGRAM frames (like all QUIC frames) must fit completely inside a QUIC packet. In turn, QUIC packets must fit completely inside a UDP datagram since fragmentation is disabled in QUIC.
- To tunnel SRT over QUIC datagrams, a single SRT packet should be encapsulated into a single DATAGRAM frame (within the Datagram Data field of a QUIC datagram).
- See Tunnelling SRT over QUIC Internet-Draft (draft-sharabayko-srt-over-quic-00) for details.

Listing 1: DATAGRAM frame format

```
DATAGRAM Frame {
    Type (i) = 0x30..0x31,
    [Length (i)],
    Datagram Data (..)
}
```

Proof of Concept Implementation

- **The quicly library by Fastly** was selected for the project as it supports both QUIC STREAM and DATAGRAM frames.
- srt-xtransmit is a testing utility that
  - supports the UDP, TCP, SRT, and QUIC transport protocols,
  - implements generate, receive, and route commands which allow the simulation of live media transmission at a constant or variable bitrate without the need for a media encoder and decoder.
- The transmission was made from a MacBook Pro laptop located in Rendsburg, Germany (client/sender side), to a Raspberry Pi 3 Model A+ computer based in Madrid, Spain (server/receiver side). Both devices were connected to the Internet via Wi-Fi.

Test Setups

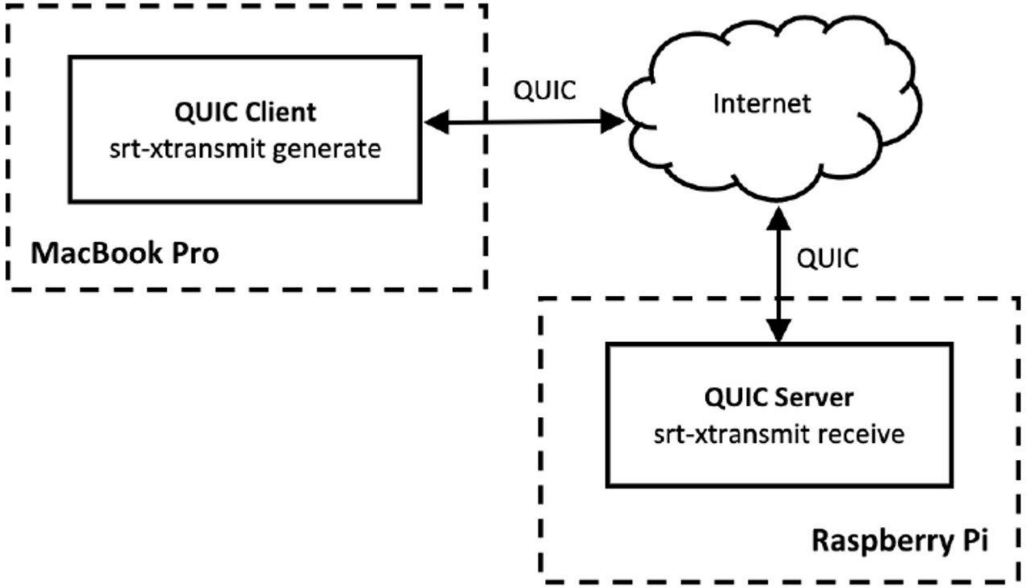


Figure 1: Test setup when sending data via QUIC datagrams.

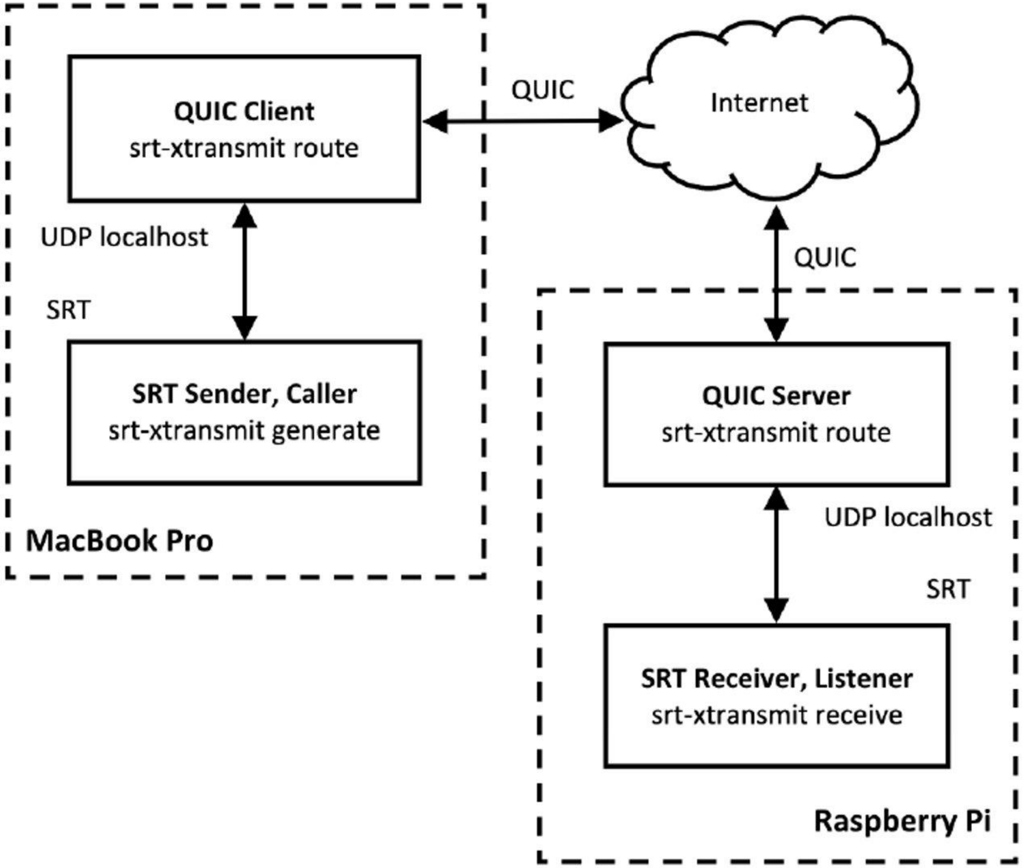


Figure 2: Test setup when tunnelling SRT over QUIC datagrams.

Performance Evaluation

- A payload contains a packet sequence number, both NTP 64-bit system clock and monotonic clock timestamps of the moment when the generation of the payload was completed at the sender side, and other fields.
- This information is used to measure the transmission time of a payload, RFC3550 jitter, a Time-Stamped Delay Factor (TS-DF), and other performance metrics at the receiver side under the assumption that the clocks on both sender and receiver machines are synchronized.

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                   Payload Sequence Number                     |
|                          (64 bit)                             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|P P|                   Group Sequence Number                   |
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                    NTP 64-Bit Timestamp                       |
|                          (64 bit)                             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                  Monotonic Clock Timestamp                    |
|                          (64 bit)                             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                       Payload Length                          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                     MD5 Checksum                              |
|                                                               |
|                                                               |
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                      Remaining Payload                        |
|                                                               |
                             (...)
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 1: Payload Structure

[Internet-Draft "Estimating Transmission Metrics on a QUIC Connection" draft-sharabayko-moq-metrics-00]

HAIVISION

Experimental Results

- We chose to limit the generated constant bitrate (CBR) stream to 3 Mbps for both streaming with QUIC datagrams and tunnelling SRT over QUIC datagrams (giving 6 Mbps in total) to ensure that link capacity would be enough for concurrent transmission of both streams.
- Streaming was done simultaneously for each experiment to equally capture the effect of possible network congestion or packet loss in both datasets.

Table 1: Summary of experiments

Experiment	SRT Latency	Duration
Experiment 1	400 ms	15 minutes
Experiment 2	600 ms	15 minutes
Experiment 3	800 ms	15 minutes
Experiment 4	800 ms	60 minutes

* Note that SRT Latency setting was applied for tunnelling SRT over QUIC transmission only

Streaming via QUIC Datagrams (Experiment 3)

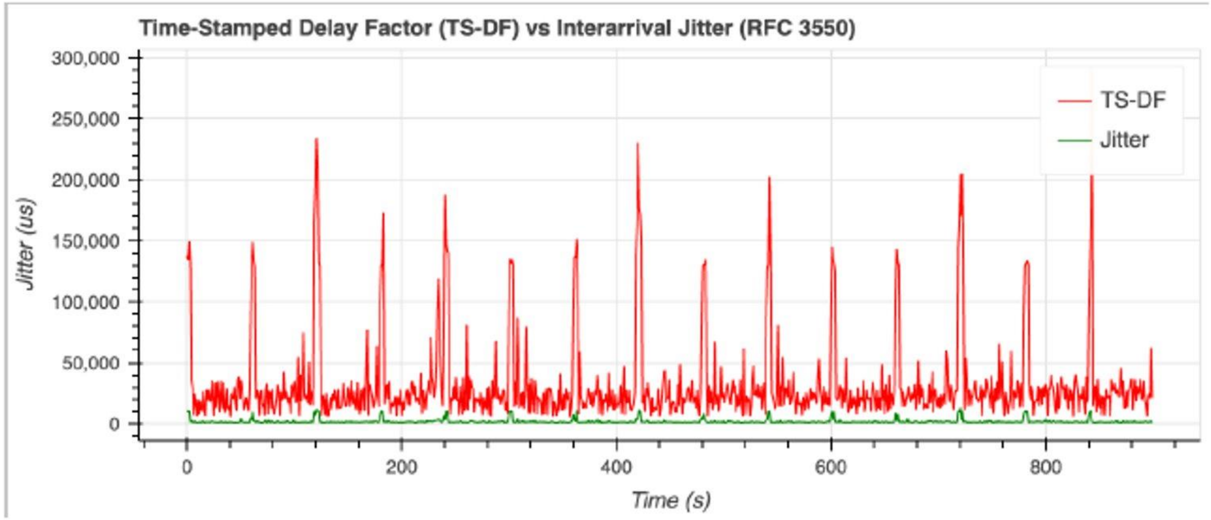


Figure 3: TS-DF vs RFC3550 jitter, in microseconds, observed at the server side when streaming via QUIC datagrams during the 3rd experiment.

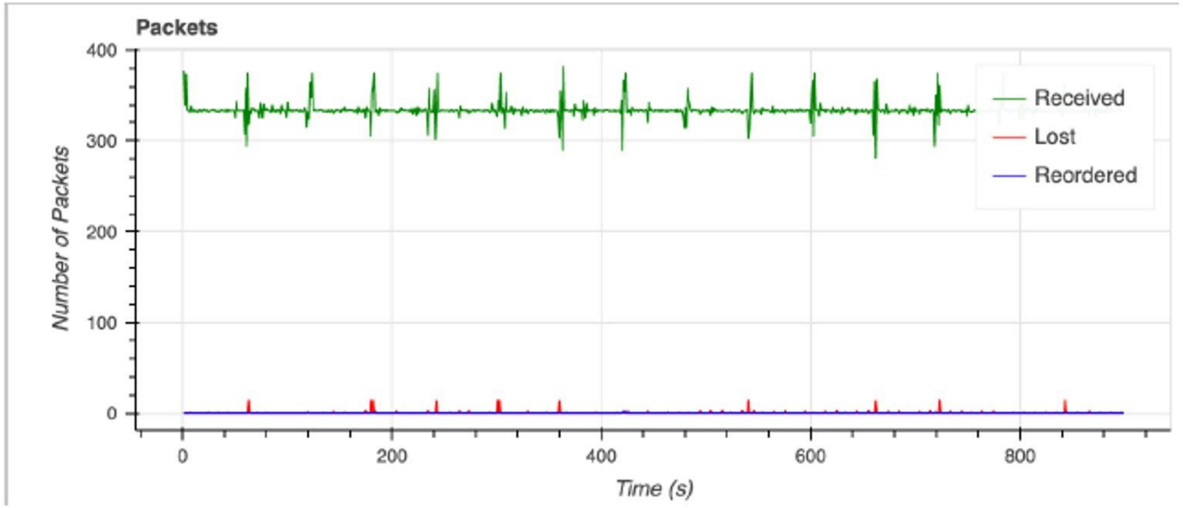


Figure 4: Number of received, lost, and reordered packets observed at the server side when streaming via QUIC datagrams during the 3rd experiment.

Table 2: Percentage of lost and reordered packets per experiment when streaming via QUIC datagrams

Statistic	Experiment 1	Experiment 2	Experiment 3	Experiment 4
Lost Packets (%)	0.001058	0.001219	0.000879	0.001083
Reordered Packets (%)	0.000019	0.000035	0.000016	0.000019

HAIVISION

Tunnelling via SRT over QUIC (Experiment 3)

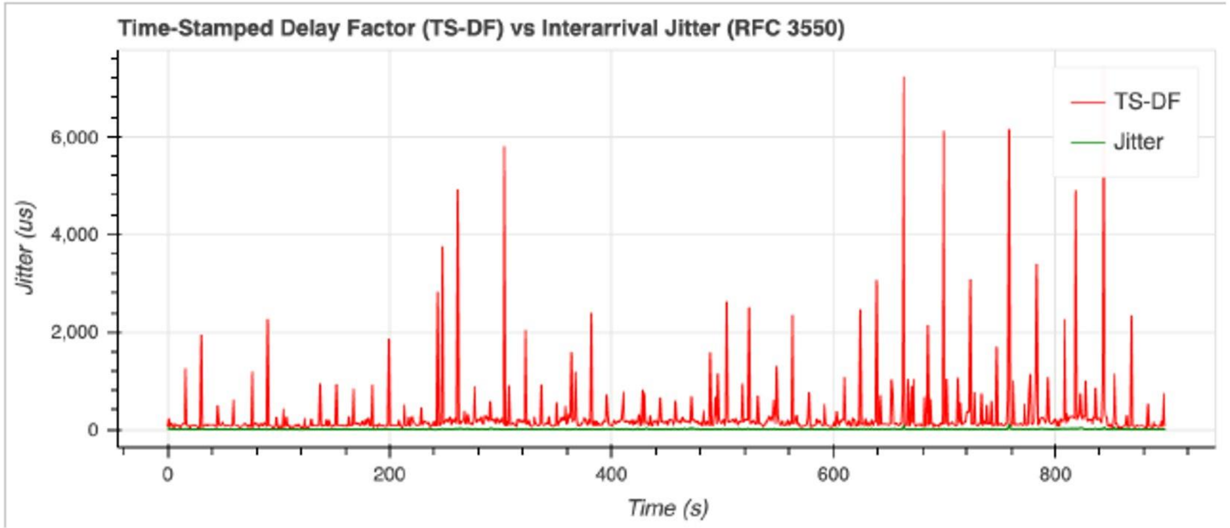


Figure 7: TS-DF vs RFC3550 jitter, in microseconds, observed at the server side when tunnelling via SRT over QUIC during the 3rd experiment.

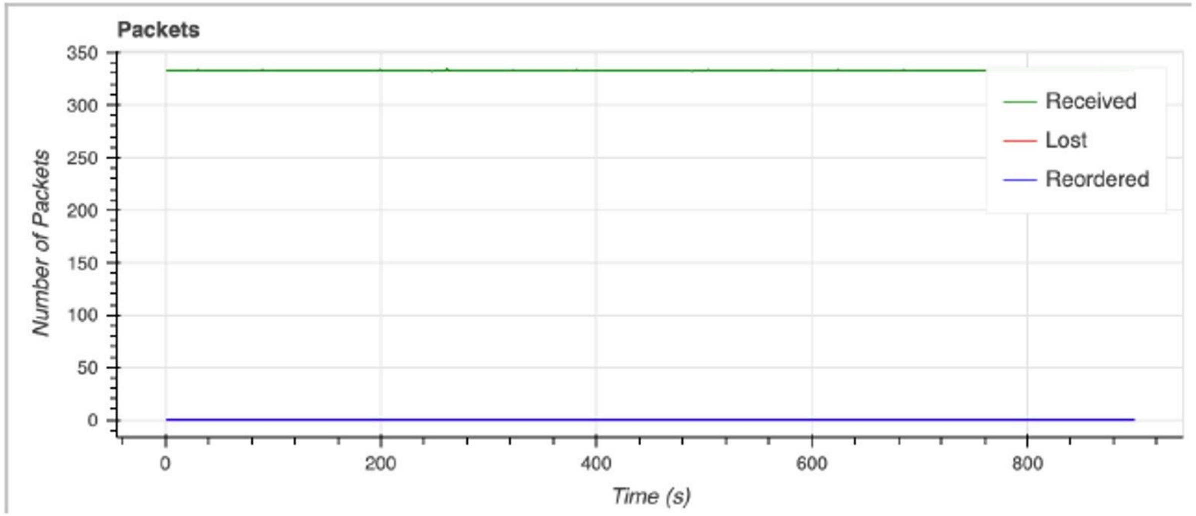


Figure 8: Number of received, unrecovered (labeled "Lost" on the graph), and reordered packets observed at the server side when tunnelling via SRT over QUIC during the 3rd experiment.

Table 3: Percentage of unrecovered and reordered packets per experiment when tunnelling via SRT over QUIC

Statistic	Experiment 1	Experiment 2	Experiment 3	Experiment 4
Unrecovered Packets (%)	0.000161	0.000033	0	0
Reordered Packets (%)	0	0	0	0

HAIVISION

Side by Side Comparison (Time-Stamped Delay-Factor)

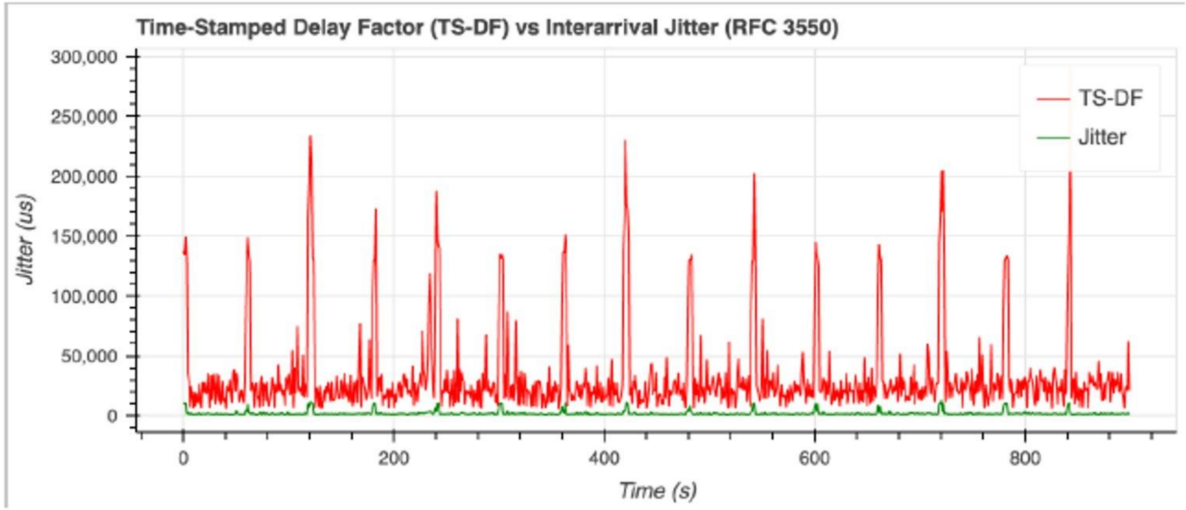


Figure 3: TS-DF vs RFC3550 jitter, in microseconds, observed at the server side when streaming via QUIC datagrams during the 3rd experiment.

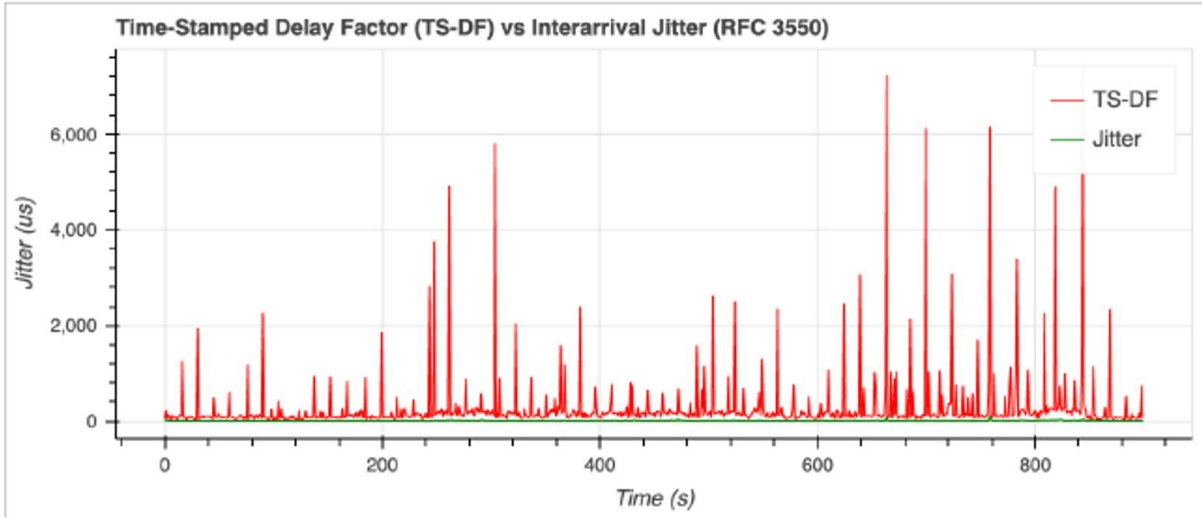


Figure 7: TS-DF vs RFC3550 jitter, in microseconds, observed at the server side when tunnelling via SRT over QUIC during the 3rd experiment.

QUIC Datagrams
Average: 33.09 ms
Spikes up to: 292.19 ms

SRT over QUIC Datagrams
Average: ~0.28 ms
Spikes up to: ~7.41 ms

HAIVISION

Smoothed Round-Trip Time (Experiment 3)

The RTT graph is built from the SRT protocol msRTT statistics observed at the receiver side and includes delay associated with transmission over QUIC datagrams.

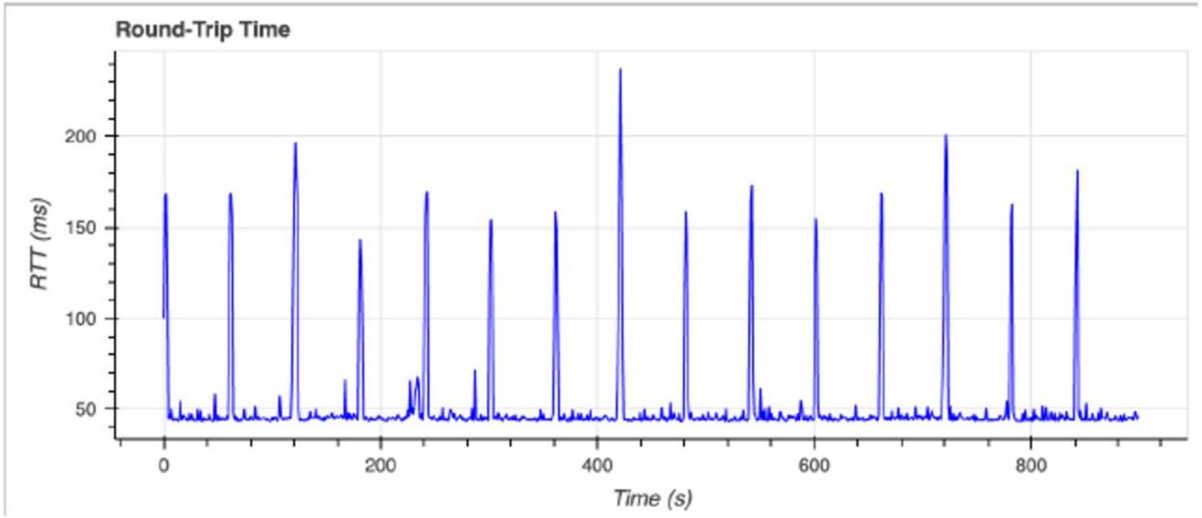


Figure 6: Smoothed round-trip time, in milliseconds, observed at the server side when streaming via SRT over QUIC during the 3rd experiment.

See also [Examining SRT streaming over 4G connection]

Time-Stamped Delay Factor for all the Experiments

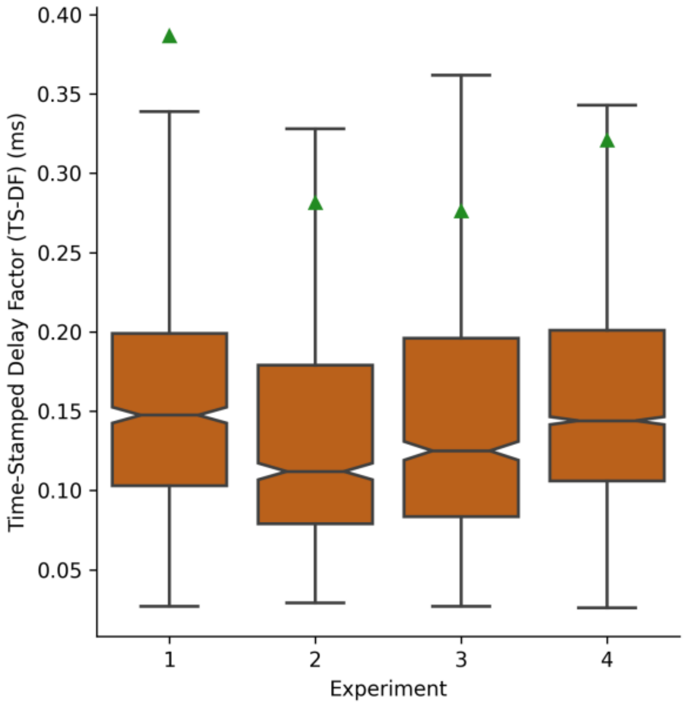
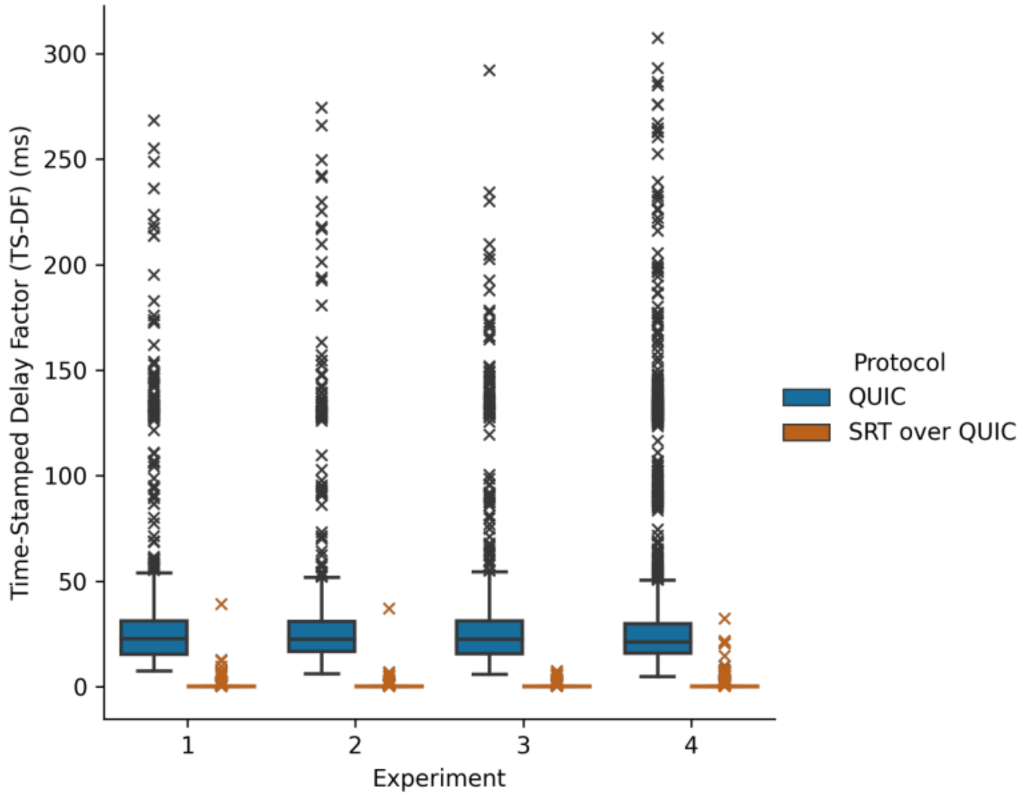


Figure 8: Time-Stamped Delay Factor, in milliseconds, observed at the server side in each experiment. Extreme values are marked with crosses, average values are marked with green triangles.

Table 1: Summary of experiments

Experiment	SRT Latency	Duration
Experiment 1	400 ms	15 minutes
Experiment 2	600 ms	15 minutes
Experiment 3	800 ms	15 minutes
Experiment 4	800 ms	60 minutes



HAIVISION

Conclusions

- The research has shown that live streaming protocols such as SRT can be implemented on top of QUIC datagrams to achieve low latency streaming, while mechanisms such as SRT’s latency-aware ARQ-based packet recovery can reduce packet loss.
- The resulting latencies and jitter can be constrained to sub-second values, depending on the network round-trip time.
- Lost packets can be recovered within the configured latency buffer, or dropped when latency boundaries are exceeded.

Contacts and References

Get more info & share ideas:

- SRT Protocol Internet-Draft
  https://datatracker.ietf.org/doc/html/draft-sharabayko-srt-01
- Tunnelling SRT over QUIC Internet-Draft
  https://datatracker.ietf.org/doc/draft-sharabayko-srt-over-quic/
- Blog on Medium
  https://medium.com/innovation-labs-blog/tagged/secure-reliable-transport
- SRT Open-source Library
  https://github.com/Haivision/srt
- SRT Alliance
  https://www.srtalliance.org/
- SRT Slack Channels
  https://srtalliance.slack.com/

  To join
  https://slackin-srtalliance.azurewebsites.net/

  Channels: #general, #develop, #quic-srt, #rfc

Authors' Contacts

Maxim Sharabayko

E-mail: maxsharabayko@haivision.com

GitHub: @maxsharabayko

LinkedIn: https://www.linkedin.com/in/maxim-sharabayko/

SRT Alliance Slack: @Maxim - Haivision

Maria Sharabayko

E-mail: msharabayko@haivision.com

GitHub: @mbakholdina

LinkedIn: https://www.linkedin.com/in/maria-sharabayko-ph-d-0256b718b/

SRT Alliance Slack: @Maria - Haivision

HAIVISION

haivision.com