A Performance Measurement Study of the Reliable Internet Stream Transport Protocol

Ciro A. Noronha, Ph.D.
Director of Technology, Compression Systems
Cobalt Digital
Agenda

• Motivation
• Overview of the Reliable Internet Stream Transport protocol
• Performance Measurement
  – Packet loss performance
  – Packet re-ordering configurations
• Conclusions: how to fine-tune a RIST link
• Review of multi-company demonstrations
Motivation

• Advances in compression technology and in network infrastructure have made it possible to use the Internet as a low-cost contribution link.
• The Internet drops packets, and a recovery protocol is necessary as every packet loss is a glitch.
• There are many proprietary solutions on the market that do not interoperate.
• The Video Services Forum (VSF) formed the Reliable Internet Stream Transport (RIST) Activity Group in early 2017 to create a common specification for a protocol suite to solve this problem.
• RIST Simple Profile was published October 2018.
Packet Recovery using ARQ

- ARQ stands for:
  - Automatic Repeat reQuest
  - Automatic Repeat Query

- This is the generic name for a number of retransmission strategies in the face of packet loss
  - Standard TCP uses a couple of ARQ variants

- In video transmission, the most useful variant is “Selective Retransmission” (NACK-based)
  - If you don’t hear from me, everything is OK
  - If I miss anything, I let you know and you resend just that

- RIST uses ARQ
Transmitted packets are saved for possible retransmission.

Packets may be retried multiple times at the expense of added latency.
RIST Protocol Basics (Sender)

- Primary stream transmission is through RTP, using the relevant standards
  - SMPTE-2022-1 for Transport Streams
  - UDP flow sent to port P, where P is an even number
- RIST sender is required to transmit RTCP packets
  - Packets sent to port P+1
  - Primary function is to establish state in firewalls for the NACK return packets
  - Suggested content:
    - Sender Report (SR) plus CNAME
    - Empty Receiver Report (RR) plus CNAME
RIST Protocol Basics (Receiver)

• Receiver listens on port P for the content, and on port P+1 for the RTCP packets
• Receiver sends periodic RTCP packets (RR+CNAME)
  — Receiver RTCP packets are sent to the source IP address and source UDP port of the received RTCP packets
  — Firewalls will treat these as “response” to the sender RTCP packets
• If the receiver detects packet loss, it will send a retransmission request for the missing packets
  — Retransmission request is an RTCP packet
RIST Retransmission Requests

- RIST NACKs (Retransmission Requests) are built using standard compound RTCP packets
- A compound RTCP packet from a RIST receiver will contain RR (may be empty), CNAME, and NACK.
- RIST has defined two types of NACK messages:
  - Bitmask Message:
    - Can request any pattern within a group of 17 consecutive packets
    - Useful for “salt and pepper” loss
    - Generic NACK from RFC 4585
  - Range Message
    - Can request a block of consecutive packets
    - Implemented with Application-Defined RTCP message
    - RIST AG may approach IANA for a permanent registration
RIST Retransmissions

• RIST retransmissions are an exact copy of the original missed packet
• Retransmitted packets are sent together with media packets (RTP sent to the same port P)
• Retransmitted packets are differentiated from original packets using the SSRC field
  – Last bit of SSRC is zero for original packets, one for retransmissions
  – Identifying retransmissions helps with system stability
RIST and Firewalls

Sender

RTP
RTCP

Public IP “S”

Transmits to IP “R” ports P and P+1

Creates state in the firewall for return RTCP packets

Media

Internet

No configuration needed

Forward ports P and P+1 to Receiver

NACKs

Flows through the firewall since it is considered “response” to sender RTCP packets

Receiver

Public IP “R”

Listening on ports P and P+1

Sent to IP “S”, directed at the source port of the RTCP flow
Bonding Support

• RIST Simple Profile has support for Bonding
  – Sender splits the stream over multiple physical channels
  – Receiver can send NACKs over each of the paths
  – Can also be used for redundancy (in the same fashion as SMPTE-2022-7)
    • Two or more copies of the same stream can be sent over distinct links

Packet reordering is supported by adding a reorder section to the receiver buffer
Packet Loss
Performance Measurement

- Media bit rate: 8 Mb/s (1920×1080i59.54 source)
- Simulated round-trip delay: 200 milliseconds
- Random i.i.d. packet losses:
  - Single packet losses
  - 5-packet burst losses
- Two-minute runs
- Independent variable: number of retries, tested from 1 to 10
- Receiver retransmission buffer set to $(200R + 100)$ milliseconds, where $R$ is the number of retries
- Sender buffer set high enough to handle the worst-case receiver buffer
- For each retry value, increase the packet loss until at least one unrecovered packet is detected in the two-minute run.
- Record this packet loss rate
- Repeat each test 10 times

![Diagram](https://via.placeholder.com/150)

**Encoder**

**Packet Delay**

**Linux “netem”**

**Packet Dropping**

**Custom App**

**Test Automation**

**Custom SNMP**

**Network Emulator**

**Decompressor**

**Encoder**

**Network Emulator**

**Decoder**

**Test Automation**

**Custom SNMP**

**Packet Delay**

**Linux “netem”**

**Packet Dropping**

**Custom App**

**Test Automation**

**Custom SNMP**
Single-Loss Results

Maximum Packet Loss for 2-minute Error-Free Run (single losses)

Number of Retries

Packet Loss (%)

Safe Operating Region
Burst Loss Results

Maximum Packet Loss for 2-minute Error-Free Run (5 packet burst loss)

Packet Loss (%) vs Number of Retries

Safe Operating Region

Average
Low
High
Packet Re-Ordering

• In the Internet, packet re-ordering only happens when paths change
  – The only way a packet with “overtake and pass” another is if it uses a different (shorter) path
• Question: if not using bonding or multipath intentionally, is it necessary to accommodate packet re-order?
• Trade-offs:
  – Non-zero re-order buffer: increased latency
  – Zero re-order buffer: possibility of unnecessary retransmissions
• Question can only be answered with actual data on Internet traffic
Data from the Internet

<table>
<thead>
<tr>
<th></th>
<th>Total Packets</th>
<th>Reordering</th>
<th>% Reorder</th>
</tr>
</thead>
<tbody>
<tr>
<td>CDN</td>
<td>90,905,926</td>
<td>28,558</td>
<td>0.031%</td>
</tr>
<tr>
<td>Tier-1 ISP</td>
<td>39,403,671</td>
<td>307,615</td>
<td>0.781%</td>
</tr>
<tr>
<td>Tier-2 ISP</td>
<td>245,535,16</td>
<td>1</td>
<td>0.384%</td>
</tr>
<tr>
<td>OC48</td>
<td>153,143,82</td>
<td>2</td>
<td>0.427%</td>
</tr>
<tr>
<td>Total</td>
<td>528,988,58</td>
<td>0</td>
<td>0.365%</td>
</tr>
</tbody>
</table>

- Internet backbone measurements indicate that the incidence of out-of-order packets is, on average, a fraction of a percent of the traffic.

- In the absence of any additional information, it is unnecessary to set a re-order buffer for a single-link RIST connection over the Internet.

Data derived from:
Configuring a RIST Link

- Input parameters/requirements (site data):
  - Network round-trip time (found with “ping”)
  - Maximum acceptable transport latency (if required)
  - Network loss (if known)
- Configurable parameters:
  - Retransmission Buffer
  - Re-order Buffer
  - Number of Retries
- Problem: select the values for the configurable parameters from the site data
Recommendations

• If there is a latency limit:
  – Set the retransmission buffer to the latency limit
  – Divide the latency limit by the round trip time and round up to find the number of retries

• If there is no latency limit:
  – If the network loss is known, read the number of retries from the performance plots and add a margin; set the retransmission buffer to at least the number of retries times the round trip
  – If the network loss is not known, a good starting point for the number of retries is 4

• Set transmitter buffer size (if configurable) as high as it will go

• Re-order can be set to zero unless using bonding
  – If using bonding, set to at least the worst case differential delay
IBC 2018 Demo

- 8 companies each sent a 5 Mb/s stream over the Internet to the Cobalt headquarters in Champaign, Illinois
- The streams were received by Cobalt 9990-DEC decoders, combined in a multiviewer, and published to YouTube in real time
- Streams were sent from UK, Canada, Israel, and the US (Northern CA, Southern CA, Florida, Virginia and Massachusetts)
- Independent implementations from the specification (no source code sharing)
IP SHOWCASE THEATER AT NAB – APRIL 8-11, 2019
VidTrans 2019 Demo

- VidTrans 2019 was held in Los Angeles (Marina Del Rey) in February 2019
- A number of participating companies provided on-site receivers at the conference
- Streams were sent from locations in the world to the receivers at the conference
  - “Mix and match” of senders and receivers
- A camera in the show floor transmitted to a relay in the San Francisco area which bounced it back to the conference
  - Sub 1-second end-to-end latency
IP SHOWCASE THEATER AT NAB – APRIL 8-11, 2019
Ongoing RIST Work

• Planned for future RIST profiles:
  – Content encryption
  – VPN support
  – NULL packet suppression (for transport streams)
  – Encoder rate control based on network availability
  – Support for high bit rate streams
• The objective is to provide all the features required for Internet contribution
Thank You

Ciro A. Noronha, Ph.D.
Cobalt Digital
ciro.noronha@cobaltdigital.com
+1 650 208-0605