Synchronized Multi-Stream Transport Using RIST

Ciro A. Noronha, Ph.D.
Cobalt Digital Inc.
Champaign, Illinois, USA
ciro.noronha@cobaltdigital.com

Abstract – Many broadcast contribution applications require synchronized content decoding. In such applications, there are several video sources (typically cameras), whose content needs to be transported using encoders to an equivalent number of remote decoders. At the decoders, the playback needs to be synchronized – frames that arrive together in the encoders are required to come out together in the decoders. Typical applications are sports and worship, where multiple camera angles are generated and need to be played in lockstep. There are a few products available today in the market that provide such a functionality over IP, using proprietary protocols. The Reliable Internet Stream Transport (RIST) Activity Group is completing the work on TR-06-4 Part 1, which provides an open industry specification for providing synchronized multi-stream transport. This paper provides a technical description of the methods in the Specification, as well as some actual field performance data.

Problem Definition

Synchronized content decoding refers to an application whereby multiple audio/video streams are transported from one location to another and must be displayed at the destination with the same relative synchronization as they had at the source. More specifically, the problem consists of the following elements:

- *N* video/audio encoders (not necessarily co-located)
- *M* video/audio decoders (not necessarily co-located), where $M \geq N$
- An IP network (possibly the Internet) connecting the encoders to the decoders.

The requirement is that frames of video that are presented to the encoders at the same time be played back at the decoders at the same time. “At the same time” means within one frame time. It is assumed that audio, if present, is synchronized with the video through the usual means.

Typical Applications

One typical application for synchronized decoding is remote sports production, illustrated in Figure 1. In this application, there are multiple cameras at a sports venue, to capture different angles of the event. If all these camera angles are transported to a remote location for production, the content needs to be synchronized at the playback to allow a director to cut between scenes. In this application, the cameras and decoders are usually all genlocked to a reference.

This paper is excerpted from the Proceedings of the 2022 NAB Broadcast Engineering and Information Technology (BEIT) Conference, © 2022, National Association of Broadcasters, 1 M Street S.E., Washington, DC 20003 USA.

References to the papers contained in the 2022 Proceedings may be made without specific permission but attributed to the *Proceedings of the 2022 NAB Broadcast Engineering and Information Technology Conference.*

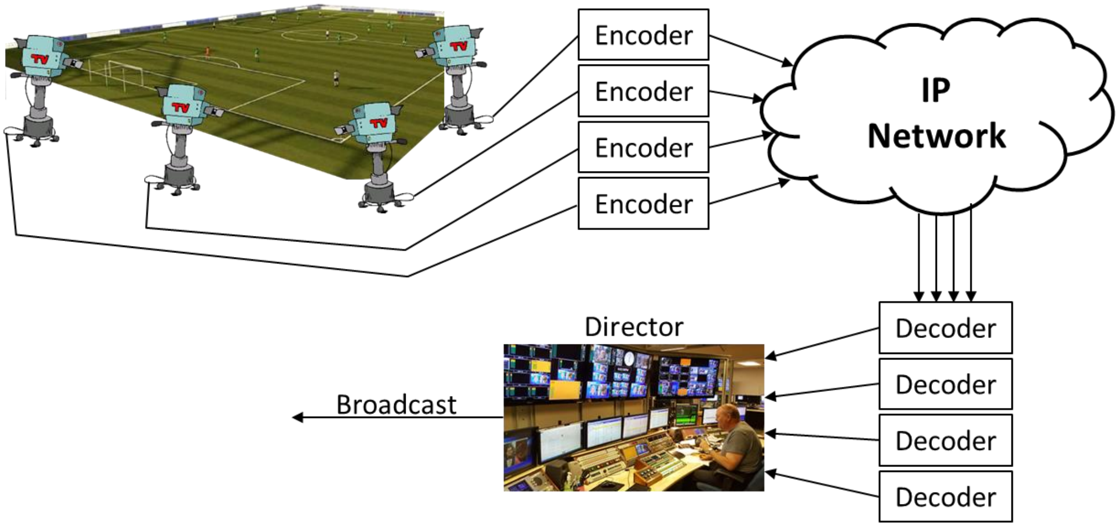


FIGURE 1: REMOTE SPORTS PRODUCTION

Another typical application is a house of worship, illustrated in Figure 2. In this case, there is a main church and a remote church, connected by an IP network. The religious service happens at the main church, and is transmitted to the remote church, which is equipped with large screens. The video between these screens needs to be synchronized as the pastor may move between cameras, and there should be no noticeable lag between them.

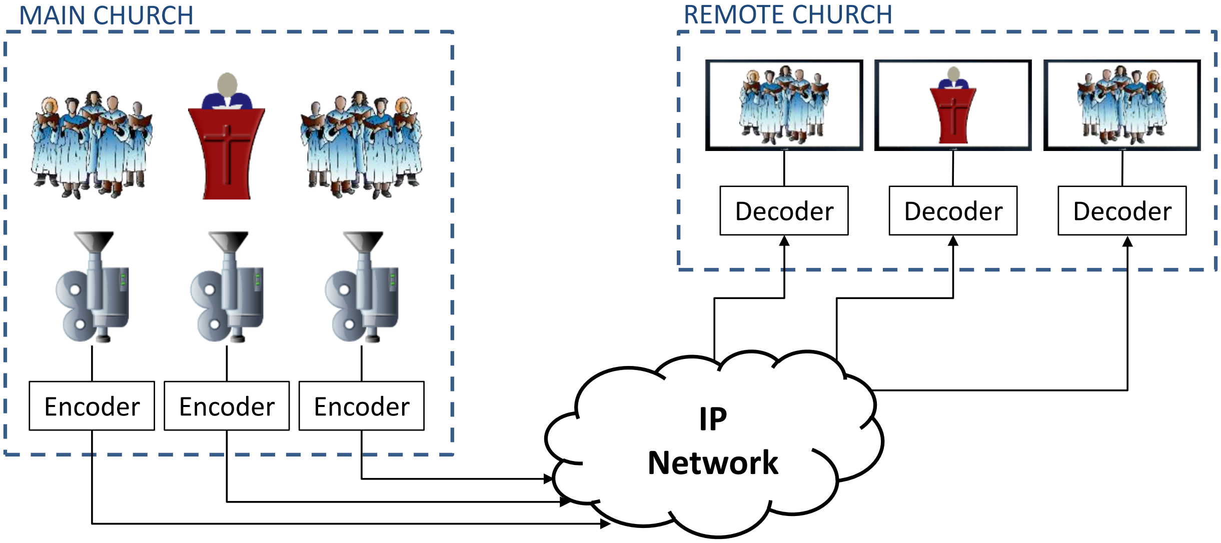


FIGURE 2: HOUSE OF WORSHIP

Using RIST For Decoder Synchronization

The Reliable Internet Stream Transport (RIST) protocol [1] was developed by the Video Services Forum as an open specification for an interoperable low-latency video transport protocol over the Internet. As RIST already provides the reliable transport over an IP network, it is a natural extension to add a decoder synchronization feature to it. The RIST Activity Group (AG) has completed the technical work on this feature. Once approved, the RIST decoder synchronization feature is expected to be published as TR-06-4 Part 1.

Synchronization Algorithm

Figure 3 illustrates the delays from video ingest to playback. The Total End-to-End Delay is composed of the following components:

- **Encoding Delay:** This is the time between the moment a frame of video enters the encoder, and the same frame is transmitted onto the network. This delay is typically fixed and is an intrinsic feature of the encoder.
- **Network Delay:** This is the network propagation delay. This delay can be variable depending on the network traffic.
- **Protocol Delay:** RIST includes some additional delay for ARQ and optional packet reordering.
- **Sync Delay:** Adjustable additional delay to achieve decoder synchronization.
- **Decode Delay:** This is the time it takes the decoder to present the frame once it receives the bitstream.

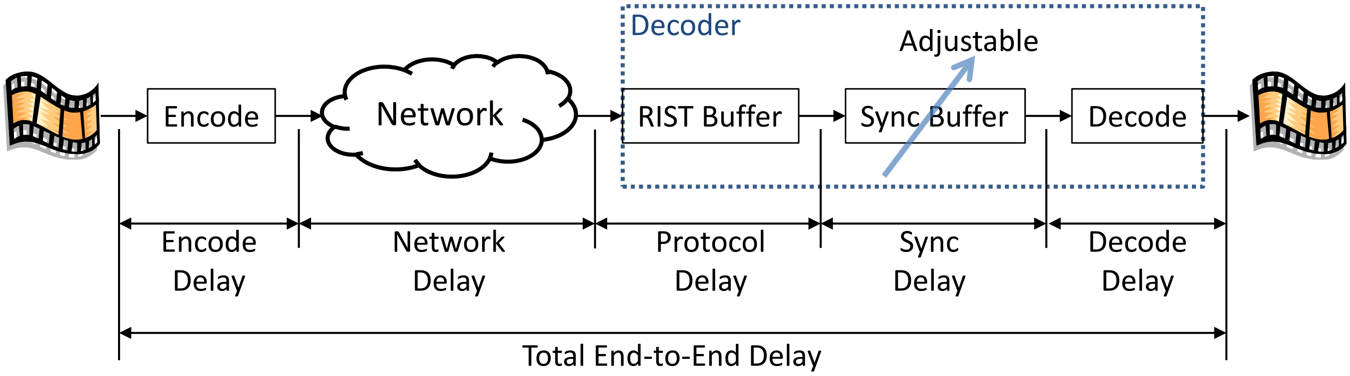


FIGURE 3: END-TO-END DELAY COMPONENTS

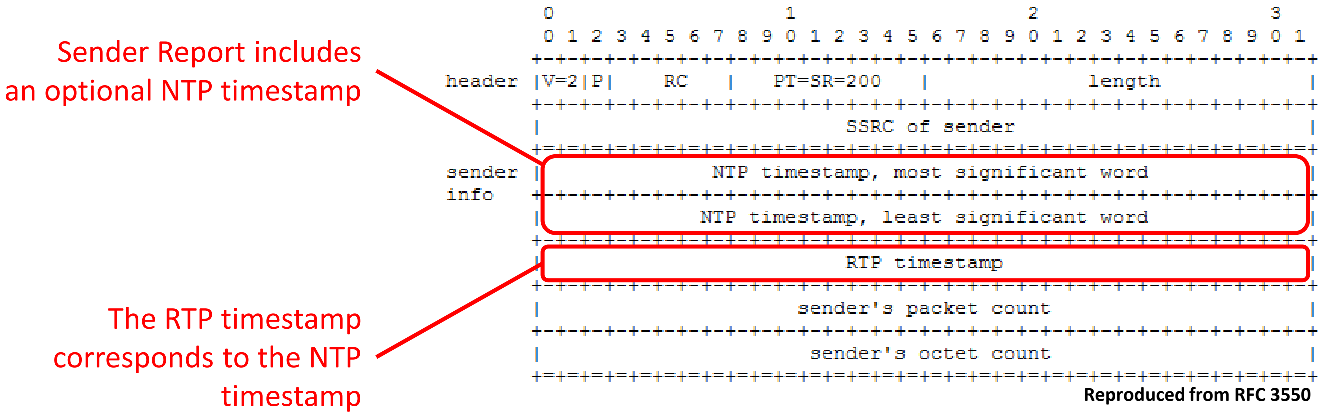
The final purpose of the synchronization algorithm is to provide the decoders enough information to dynamically adjust the Sync Buffer in Figure 3 so that the end-to-end delay is the same for every decoder. To achieve this, a common time base is required between encoders and decoders. The RIST AG has selected the Network Time Protocol (NTP) [2] as this common time base. Once there is a common time base, the following high-level algorithm has been selected:

- Encoders will need to provide decoders with the NTP time corresponding to the instant each frame of video has been captured. For efficiency purposes, this does not need to be done for every frame; it can be done periodically, and the decoders will interpolate. Note that the video frame rate clock is not synchronized with NTP, so the information needs to be sent frequently enough to compensate for the clock drift.
- Decoders add a fixed delay to the received NTP time stamp to determine the frame playback time. This is the Total End-to-End Delay in Figure 3. This value must be larger than the sum of Encode Delay, Network Delay, Protocol Delay and Decode Delay. The decoder implements this delay by adjusting the Sync Delay.

Protocol Support

To implement the synchronization algorithm, the timestamps need to be carried from the encoders to the decoders. Fortunately, RIST Simple Profile [1] already includes a suitable mechanism for this communication – the RTCP Sender Report (SR) message, defined in RFC 3550 [3]. RIST Simple Profile uses RTP for data transmission and requires the sender to transmit periodic SR messages every 100 milliseconds to establish state in firewalls so that NACK messages can come back to the sender (see [4] for a detailed explanation). RIST Simple Profile allows for empty SR messages and does not require receivers to process such messages in any way.

The SR messages already include the necessary fields for synchronization, as illustrated in Figure 4. It includes an optional NTP timestamp, and the corresponding RTP timestamp. These two fields are used to inform the recipient that a packet with the indicated RTP timestamp was transmitted at the indicated NTP time. This mechanism is in widespread use to synchronize audio and video for RTSP devices.



Sender Report includes an optional NTP timestamp

The RTP timestamp corresponds to the NTP timestamp

```
        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
       +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
header |V=2|P|    RC   |   PT=SR=200   |             length            |
       +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
       |                         SSRC of sender                        |
       +=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
sender |              NTP timestamp, most significant word             |
info   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
       |             NTP timestamp, least significant word             |
       +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
       |                         RTP timestamp                         |
       +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
       |                     sender's packet count                     |
       +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
       |                      sender's octet count                     |
       +=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
```

Reproduced from RFC 3550

FIGURE 4: SYNCHRONIZATION FIELDS IN THE SR RTCP MESSAGE

To provide decoder synchronization, the RIST AG has made small changes to the interpretation of the two fields above. The changes are summarized in Table 1.

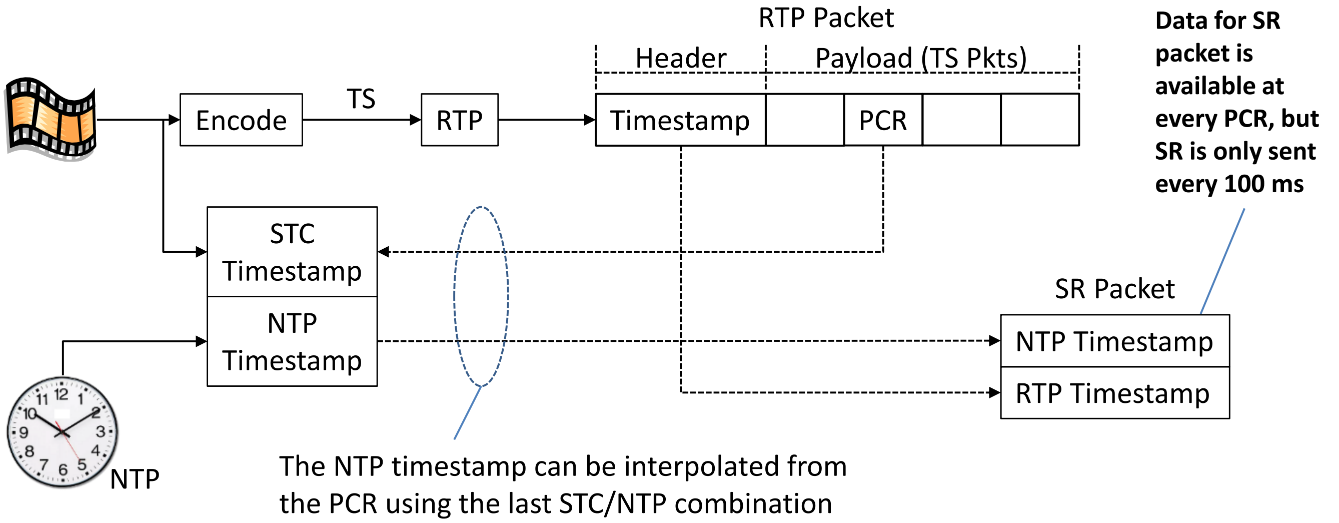
TABLE 1: DIFFERENCES BETWEEN TR-06-4 PART 1 AND RFC 3550

TR-06-4 Part 1	RFC 3550
NTP timestamp is required	NTP timestamp is optional, can be set to zero
NTP timestamp must come from an actual NTP server	NTP timestamp can be device's wall clock
NTP timestamp corresponds to frame capture time	NTP timestamp corresponds to SR message transmission time
RTP timestamp corresponds to timestamp of the packet carrying the frame	RTP timestamp corresponds to the same point in time as the NTP timestamp

Implementation

To fill in the SR message, an encoder will need to tie NTP timestamps obtained when frames are captured to RTP timestamps. The process is illustrated in Figure 5 for an MPEG encoder producing a Transport Stream, and works as follows:

- Every time a frame is captured, the encoder records the value of the System Time Clock[1] (STC) and the corresponding NTP timestamp.
- Every time an RTP packet is prepared for transmission, it is inspected to see if it carries a Program Clock Reference (PCR) timestamp. The PCR is a sample of the STC.
- If the RTP packet carries a PCR, then the value of the PCR is used to interpolate the NTP timestamp from the last STC/NTP capture. This interpolated NTP is now associated with the packet's RTP Timestamp.
- If it is time to send an SR packet, these are the two values used in the packet.



Data for SR packet is available at every PCR, but SR is only sent every 100 ms

The NTP timestamp can be interpolated from the PCR using the last STC/NTP combination

FIGURE 5: SR MESSAGE GENERATION

Actual Field Test Data

We have implemented the proposed algorithm in a set of encoders/decoders and tested the system over the Internet. The test setup is depicted in Figure 6 and includes:

- Two encoders located in Santa Clara, California.
- One video source with burned-in VITC timecode, connected to both encoders, running at 1920x1080i at 29.97 frames/second.
- Two decoders located in Champaign, Illinois, connected to a multi-viewer with output snapshot capability.
- A standard Internet connection between these two locations.

Each encoder transmits a unicast stream over the Internet with RIST to a decoder. RIST Simple Profile is used to recover any lost packets. The multi-viewer can be used to take snapshots of the combined video from both decoders, and the timecode in both screens can be compared to see if the decoders are synchronized.

[1] The System Time Clock for a Transport Stream is a 42-bit counter driven by a 27 MHz clock locked to the video frame rate.

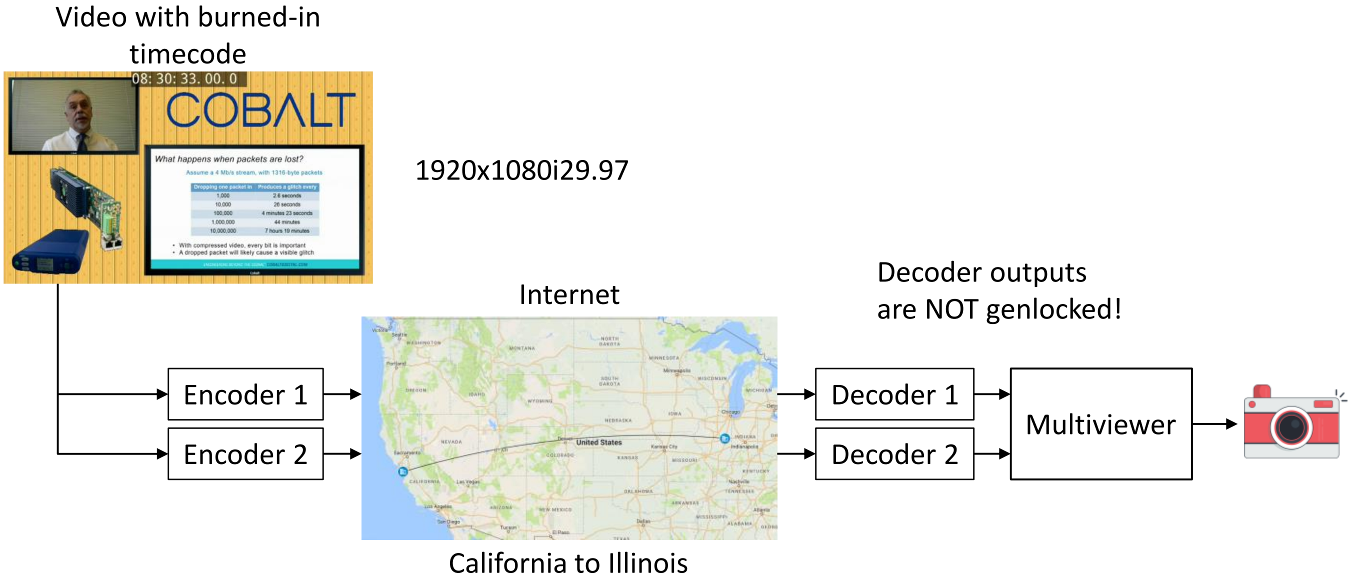
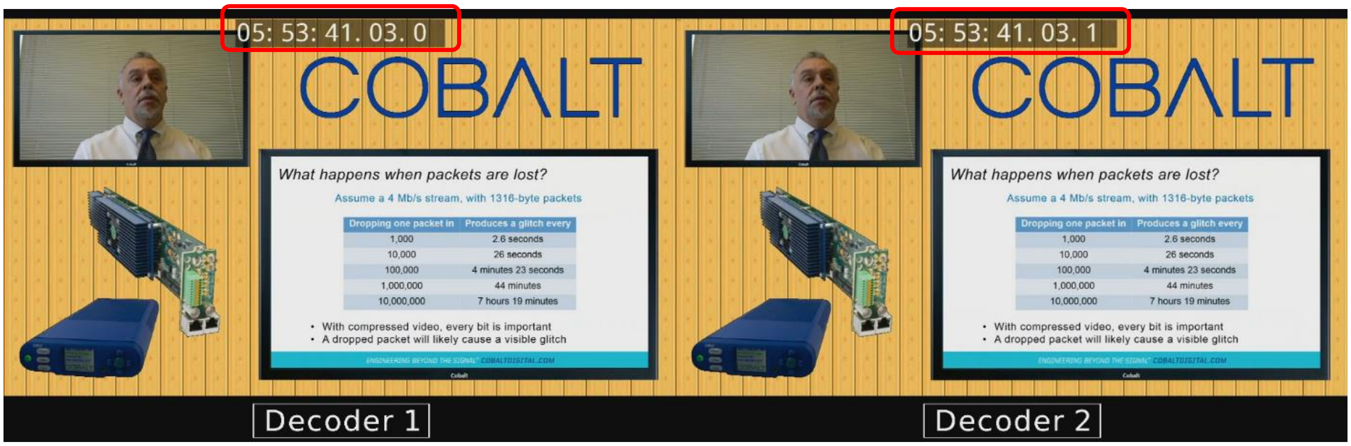


FIGURE 6: TEST SETUP

Test Results

Our first observation is that, when the system starts, the decoders are not quite in alignment. This is because the decoders start with their own version of the STC, which is yet not locked to the incoming stream. Over the course of the first few minutes of operation, the decoders achieve frequency and phase lock with the encoder STC, and the video outputs of the two decoders become aligned. Figure 7 shows the initial lock, a few minutes after the decoders have been started. The video is not completely aligned, Decoder 2 is one field ahead of Decoder 1.



What happens when packets are lost?

Assume a 4 Mb/s stream, with 1316-byte packets

Dropping one packet in	Produces a glitch every
1,000	2.6 seconds
10,000	26 seconds
100,000	4 minutes 23 seconds
1,000,000	44 minutes
10,000,000	7 hours 19 minutes

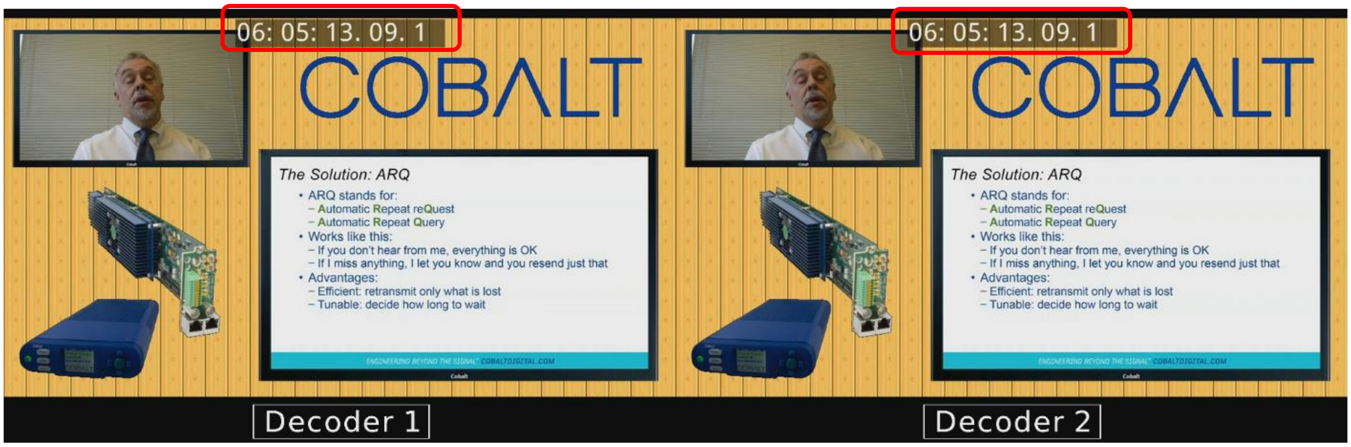
- With compressed video, every bit is important
- A dropped packet will likely cause a visible glitch

ENGINEERING BEYOND THE SIGNAL™ COBALTDIGITAL.COM

FIGURE 7: INITIAL LOCK

Figure 8 shows the final alignment of the two decoders. This snapshot was taken approximately 12 minutes after the one in Figure 7. As demonstrated by the timecodes, both decoders are fully aligned.

(approximately 12 minutes later)

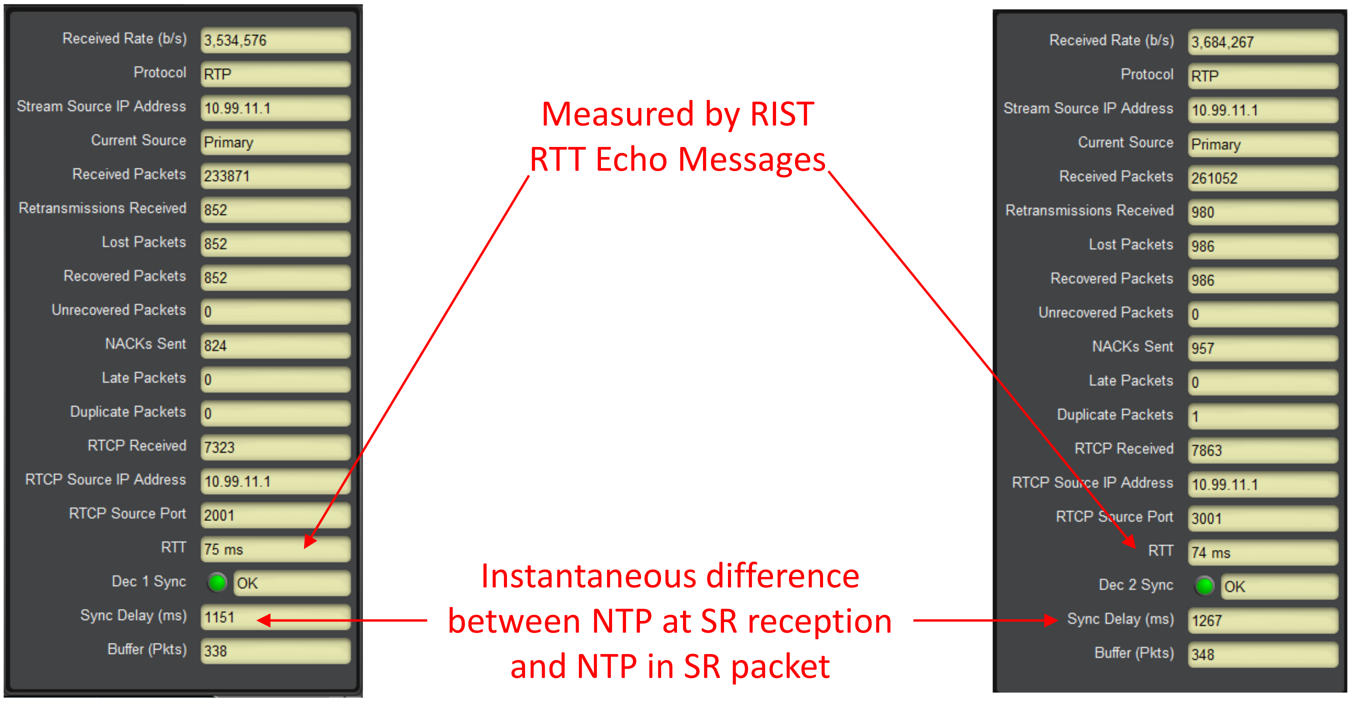


The Solution: ARQ
- ARQ stands for:
  - Automatic Repeat reQuest
  - Automatic Repeat Query
- Works like this:
  - If you don't hear from me, everything is OK
  - If I miss anything, I let you know and you resend just that
- Advantages:
  - Efficient: retransmit only what is lost
  - Tunable: decide how long to wait

The Solution: ARQ
- ARQ stands for:
  - Automatic Repeat reQuest
  - Automatic Repeat Query
- Works like this:
  - If you don't hear from me, everything is OK
  - If I miss anything, I let you know and you resend just that
- Advantages:
  - Efficient: retransmit only what is lost
  - Tunable: decide how long to wait

FIGURE 8: FINAL ALIGNMENT

Figure 9 shows some statistics for this run. It indicates that the round-trip time for the setup is on the order of 75 milliseconds, and even though the Internet dropped packets, RIST recovered them all. Another important measurement is the sync delay, namely the instantaneous difference between the current NTP time and the NTP timestamp in the SR packet, at the time of the SR packet reception. In this case, most of the delay is inside the encoder.



Received Rate (b/s)	3,534,576
Protocol	RTP
Stream Source IP Address	10.99.11.1
Current Source	Primary
Received Packets	233871
Retransmissions Received	852
Lost Packets	852
Recovered Packets	852
Unrecovered Packets	0
NACKs Sent	824
Late Packets	0
Duplicate Packets	0
RTCP Received	7323
RTCP Source IP Address	10.99.11.1
RTCP Source Port	2001
RTT	75 ms
Dec 1 Sync	● OK
Sync Delay (ms)	1151
Buffer (Pkts)	338

Measured by RIST RTT Echo Messages

Instantaneous difference between NTP at SR reception and NTP in SR packet

Received Rate (b/s)	3,684,267
Protocol	RTP
Stream Source IP Address	10.99.11.1
Current Source	Primary
Received Packets	261052
Retransmissions Received	980
Lost Packets	986
Recovered Packets	986
Unrecovered Packets	0
NACKs Sent	957
Late Packets	0
Duplicate Packets	1
RTCP Received	7863
RTCP Source IP Address	10.99.11.1
RTCP Source Port	3001
RTT	74 ms
Dec 2 Sync	● OK
Sync Delay (ms)	1267
Buffer (Pkts)	348

FIGURE 9: DECODER STATISTICS

The results presented above are for non-genlocked decoders. They will play in synchronization, but the video time base will be that of the encoders, recovered by the standard MPEG clock recovery. In many cases, however, it is necessary or desirable to genlock the decoders to a studio reference. The genlock process will occasionally drop or repeat frames as needed to match the video to the reference. Such drop/repeat events are not necessarily synchronized, so multiple genlocked outputs may be one frame off. This is shown in Figure 10, where we have genlocked decoders. The genlock process happens to be in a state where Decoder 2 is one frame ahead of Decoder 1.

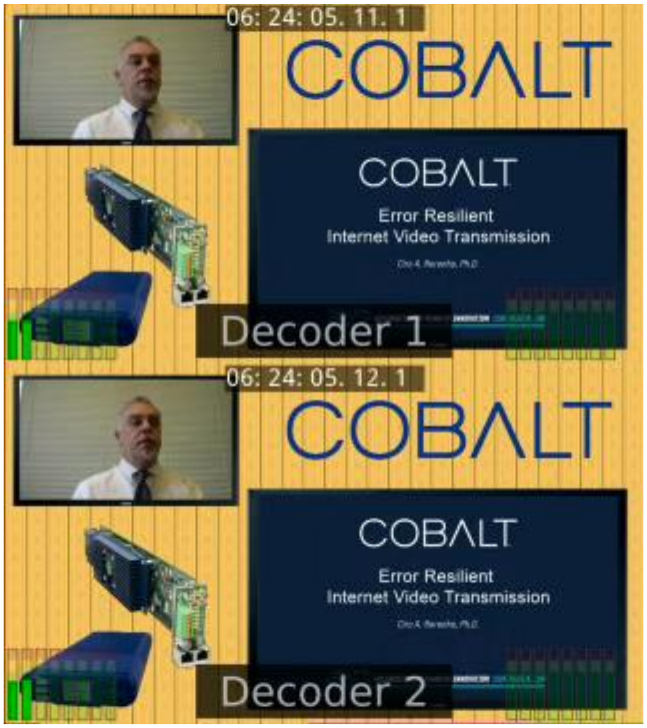


FIGURE 10: GENLOCKED DECODERS SHOWING A FRAME DIFFERENCE

Conclusions

We have demonstrated that the method proposed in the upcoming TR-06-4 Part 1 Specification can provide synchronized decoding over the Internet. Decoder clock recovery is a factor, and it may take a while to achieve synchronization – this is dependent on the specific decoder clock recovery. Ideally, the clock recovery should be made a part of the synchronization process, but this is an implementation detail and not related to the RIST Specification.

Another observation has to do with the Total End-to-End Delay setting from Figure 3. This is a value that needs to be manually configured in all the decoders and needs to be large enough to compensate for the worst-case encoder-decoder delay. The RIST Specification has not addressed this issue; it is currently done by manual configuration and could be error prone. More work is needed to automate this setting.

References

[1] Video Services Forum TR-06-1, "Reliable Internet Stream Transport (RIST) Protocol Specification – Simple Profile", 2020-06-25, < https://vsf.tv/download/technical_recommendations/VSF_TR-06-1_2020_06_25.pdf>.

[2] Mills, D., Martin, J., Ed., Burbank, J., and W. Kasch, "Network Time Protocol Version 4: Protocol and Algorithms Specification", *RFC 5905, DOI 10.17487/RFC5905*, June 2010, <https://www.rfc-editor.org/info/rfc5905>.

[3] Schulzrinne, H., Casner, S., Frederick, R., and V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications", *STD 64, RFC 3550, DOI 10.17487/RFC3550*, July 2003, <https://www.rfc-editor.org/info/rfc3550>.

[4] Noronha, C., "A Performance Measurement Study of the Reliable Internet Stream Transport Protocol", *Proceedings of the 2019 NAB Broadcast Engineering and Information Technology Conference.*