A Study of Protocols for Low-Latency Video Transport over the Internet

Ciro A. Noronha, Ph.D.
Cobalt Digital
Santa Clara, CA
ciro.noronha@cobaltdigital.com
Juliana W. Noronha
University of California, Davis
Davis, CA
jwnoronha@ucdavis.edu

Abstract - *The speed and quality of the Internet links available today have made it possible to employ them as broadcast contribution links. However, Internet links experience packet loss. The protocols that deal with this loss represent a tradeoff between latency and reliability. This paper investigates two common packet loss recovery protocols: Selective Retransmission (also known as ARQ) and SMPTE-2022 FEC. We start with a statistical modeling analysis of these two protocols. Next, using an actual encoder/decoder pair, we present a few lab measurement results using a network simulator specifically designed for this test, as well as results from an Internet link between locations in California and Illinois. We conclude with guidelines for broadcasters who are considering the Internet as a low-cost contribution link, but are concerned with latency and reliability. At the time of this writing, the Video Services Forum (VSF) has started a working group called RIST (Reliable Internet Stream Transport) specifically to standardize such a protocol.*

INTRODUCTION

The speed and reliability of the Internet has steadily increased over the last few years. Over-The-Top (OTT) services are routinely used by millions to watch high-quality content. It is natural for broadcasters to consider using the Internet as a cost-effective means to transmit contribution-grade video between locations, rather than purchasing expensive dedicated links. Studios and TV stations all have Internet access already, so why not use that for contribution as well?

The issue with this approach is that Internet delivery is best-effort (since IP is best-effort), and one cannot set end-to-end priorities. Thus, there will be occasional packet losses, primarily due to instantaneous congestion. Since what is being transmitted is compressed audio/video, effects of uncorrected packet loss are quite noticeable.

Transport protocols such as TCP can deal with packet loss and provide reliable delivery. However, there is a fundamental tradeoff between reliable delivery and latency. Given enough time, a well-designed protocol can get every packet delivered. OTT applications, for instance, use HLS [1] or DASH [2] for delivery; these protocols may impose a latency in the order of tens of seconds. For stored content, this is acceptable; it is better to have reliable delivery than low latency. However, for many broadcast applications, latency needs to be low, and the protocol may need to "give up" on packets that cannot be recovered in a given window of time.

To date, the only standard protocol that satisfies this latency requirement while providing some packet loss recovery is SMPTE-2022 FEC [3] [4]. This protocol was not designed to be used on the Internet; its primary purpose was to recover from occasional packet loss on well-managed links. However, due to the lack of options and the quality improvements on Internet links, a number of broadcasters have used SMPTE-2022 FEC with some degree of success. However, in general, something better is required.

There are a number of proprietary solutions in the market today that use variants of the well-known Selective Retransmission technique. These solutions are usually denoted by ARQ (**A**utomatic **R**epeat re**Q**uest), described in many networking textbooks [5]. The ARQ technique is known to work significantly better than SMPTE-2022 FEC.

In this paper, we provide a comparison between SMPTE-2022 FEC and one standards-based implementation of ARQ, with the objective of providing low-latency transport over the Internet. The comparison includes a statistical analysis, a lab test with a custom network simulator, and one actual test over the Internet.

ACCEPTABLE PACKET LOSS

Even with the advances in the capacity and reliability of the Internet links, they are not yet fast enough to support raw video transmission in an economical way. Therefore, compression is needed, and the better job the compression does, the more important the resulting bits become, since the amount of information they contain is higher. In other words, all the bits are important, and they must be delivered with high reliability. Given that packet loss is unavoidable if there is a latency constraint, it is important to determine what an acceptable level of packet loss is.

A simple way to determine the acceptable packet loss is to assume that every packet loss causes a glitch [6]. This is

a reasonable assumption with compressed streams because fundamentally, compression works by removing the redundancy, and thus every bit is important. In addition, the effects of a loss may persist for a while, since decoding a video frame may depend on data from previous video frames. In reality, a small fraction of the losses may not be noticeable or may be concealed by the decoder, but the more conservative approach is to assume that every loss is a noticeable glitch.

Based on this approach, it is straightforward to derive the glitch interval from the data rate and the packet size.
Defining:
R: stream rate in bits/second
B: packet payload size in bytes
p: packet loss probability
G: glitch interval in seconds
The glitch interval is given by (1):

$$G = \frac{8B}{Rp} \quad (1)$$

As an example, let us assume a 4 Mb/s video stream, with the usual 1316-byte payload. Table 1 shows the glitch interval given by (1) at different packet loss rates.

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

TABLE 1: GLITCH INTERVAL AT 4 MB/S

Determining the acceptable packet loss is application-specific, and, to a certain extent, subjective. A broadcaster may be willing to tolerate more frequent glitches on a live feed from a reporter in the field, but require a stricter standard for a studio contribution link. In any case, the acceptable packet loss can always be derived from the desired glitch interval using the method described above.

PROTOCOL REVIEW

I. SMPTE-2022 FEC

The SMPTE-2022 FEC protocol was designed to correct occasional packet loss. Its basic features are:

- The actual audio/video transfer uses RTP.
- The protocol adds parity (XOR) FEC packets to the stream to handle loss.
- A FEC packet protects a group of audio/video packets. If there is a single packet loss in this group, the lost packet can be rebuilt from the received packets and the FEC packet.
- For the purposes of FEC computation, the transmitted audio/video packets are arranged in a matrix of *C* columns and *R* rows. FEC packets are computed per column. An optional second FEC flow may be computed per row.
- The matrix arrangement enables the protocol to correct burst losses of up to *C* consecutive packets in every group of *R*×*C* packets, with an overhead of 1/*R*.
- The use of optional row FEC packets enables the protocol to correct single packet losses on each row with an overhead of 1/*C*.
- The FEC packets for a given matrix are transmitted during the next matrix. This makes the latency of this protocol equal to the transmission time of 2*RC* packets.
- The protocol is strictly unidirectional. There is no communication from the receiver(s) back to the sender, so it is well suited for one-to-many applications.
- SMPTE-2022 [3] limits *R* and *C* to 20 and *R*×*C* to 100.

The latency and overhead of this protocol depend on the matrix size and the bit rate. Table 2 shows a few common operating points.

C	*R*	Recovery pkts/block	Overhead	Latency 2Mb/s	Latency 10Mb/s
5	5	5/25	20%	263 ms	53 ms
10	5	10/50	20%	526 ms	105 ms
20	5	20/100	20%	1052 ms	211 ms
10	10	10/100	10%	1052 ms	211 ms

TABLE 2: SAMPLE SMPTE-2022 OPERATING POINTS

II. ARQ (Selective Retransmission)

ARQ stands for **A**utomatic **R**epeat re**Q**uest (or **A**utomatic **R**epeat **Q**uery) and is the generic name for a class of retransmission strategies in face of packet loss. The most useful technique for video transmission is Selective Retransmission, where the receiver only sends Negative Acknowledgements (NACK) if packet loss is detected. The sender will then retransmit only the requested packets. This is illustrated in Figure 1.

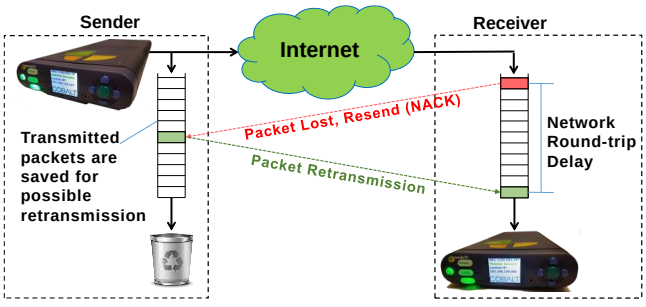

FIGURE 1: ARQ ILLUSTRATION

As indicated in Figure 1, the latency requirement for this protocol is at least one round-trip delay. If multiple round-trip delays are allowed, a dropped packet can be retried multiple times (e.g., when both the packet and its retransmission are dropped).

There are a number of proprietary and incompatible ARQ implementations available in the market today. However, for the purposes of this paper, we will consider the variant defined in RFC 4585 [7]:

- The actual audio/video transfer uses RTP, in the same fashion as SMPTE-2022 [4].
- Retransmissions are requested using the Generic NACK packet from RFC 4585 [7].
- A given packet may be requested multiple times.

The overhead of the ARQ implementation consists of the retransmitted packets, and thus is a function of the packet loss. In our overhead computations later in this paper, we disregard the NACK packets since they are typically very small, and flow in the opposite direction as the audio/video traffic.

STATISTICAL ANALYSIS OF THE PROTOCOLS

In this section, we provide a statistical analysis of the protocols. We assume that the packet loss process is an independent and identically distributed process where each packet is lost with probability p and received with probability $(1 - p)$. In most real networks, the packet loss process is more complex than this (since packets are usually dropped in bursts due to congestion), but this simplified model allows us to derive useful results.

In this analysis, we derive results for the following variables, as a function of the packet loss probability p:

p_c: uncorrected packet loss (after application of the protocol)

H: protocol overhead - amount of extra data transmitted to achieve correction, as a fraction of the total transmitted data

I. ARQ Analysis

In ARQ, a packet will remain lost if all of its retransmissions from the sender are also lost [6]. If we denote the number of allowed retransmissions by R, the uncorrected packet loss probability is given by (2), since we assume an i.i.d. process.

$$p_c = p^{R+1} \tag{2}$$

The overhead is composed simply by summing the retransmissions and is given by (3).

$$H = \sum_{i=1}^{R} p^i \tag{3}$$

II. SMPTE-2022 FEC Analysis

In SMPTE-2022 FEC, a group of packets is protected by one FEC packet. If there are two or more losses in that group of packets, then correction is not possible. Finding the uncorrected packet loss in such a case is simply a matter of calculating the probability of such an event.

In the following discussion, we will use:

N: number of rows in the FEC matrix

M: number of columns in the FEC matrix

Column-Only FEC:

In Column-Only mode, the FEC packet protects N data packets, provided there is only one loss in the group. Therefore, the uncorrected packet loss is given by (4), which is the probability of two or more losses in the group.

$$p_c = 1-(1-p)^{N+1} - Np(1-p)^N \tag{4}$$

Since there is one packet added for each group of N packets, the overhead is simply[1]:

$$H = \frac{1}{N} \tag{5}$$

Row and Column FEC:

In order to compute the uncorrected loss for the row and column mode, we divide the correction process into two parts: first, we apply row FEC, and recover whatever packets are possible. Then, we apply column FEC to the resulting stream. This is how most existing systems operate, because the row FEC packet comes immediately after the block being protected, while the column FEC packets come after the matrix is transmitted. In theory, depending on the loss pattern, it may be possible to recover additional packets by iteratively applying row and column FEC multiple times. However, such a method is not required by SMPTE-2022 and we believe that the improvement is negligible, so it is not included in our model.

In order to find the uncorrected packet loss, we simply apply (4) twice, first for the row FEC, and then for the column FEC. For a matrix with M columns, the intermediate uncorrected loss probability after row FEC is applied is given by (6).

$$p_M = 1-(1-p)^{M+1} - Mp(1-p)^M \tag{6}$$

The resulting stream is then fed to the column FEC stage, with an incoming packet loss probability of p_M. The final uncorrected loss probability is then given by (7), with p_M given by (6).

[1] In reality, SMPTE-2022 FEC packers are slightly larger than the data packets, but the difference is negligible and (5) provides a good approximation of the actual overhead.

$$p_c = 1-(1-p_M)^{N+1} - Np_M(1-p_M)^N \qquad (7)$$

In this case, M column FEC packets and N row FEC packets are added to each $M \times N$ matrix, so the overhead is:

$$H = \frac{M+N}{MN} \qquad (8)$$

III. Comparison Plots

Plotting the uncorrected packet loss p_c as a function of the input loss probability p is a good way to compare the protocols. For a given value of p, the protocol exhibiting the lower p_c has better performance.

Figure 2 shows a comparison plot between ARQ with 1, 2 and 4 retries (blue lines) versus column-only FEC with 5, 10 and 20 rows (yellow lines). It shows that ARQ outperforms column-only FEC for all settings, and that there is a significant improvement in ARQ performance as more retries are allowed.

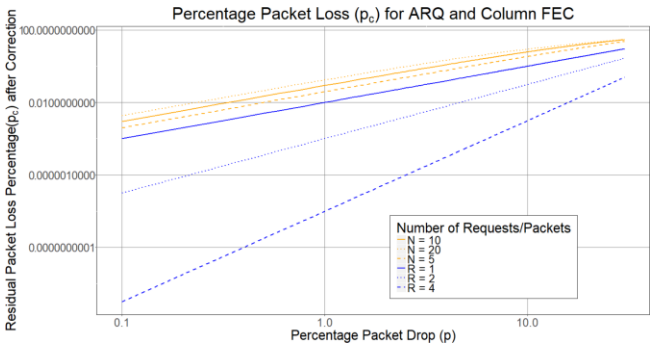

FIGURE 2: ARQ VERSUS SMPTE-2022 COLUMN ONLY

Figure 3 shows a comparison plot between ARQ with 1, 2 and 4 retries versus row and column FEC with 5×5, 10×10, and 20×5 matrices. In this case, for some operating points at low loss rates, FEC can outperform ARQ with 1 or 2 retries. However, at 4 retries ARQ is still superior to these FEC configurations.

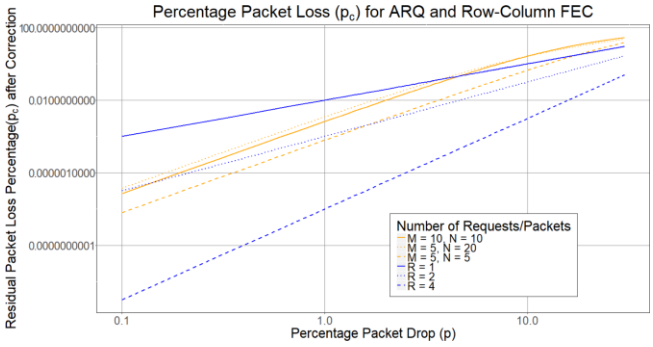

FIGURE 3: ARQ VERSUS SMPTE-2022 ROW AND COLUMN

Finally, the protocol overhead is presented in Figure 4. The SMPTE-2022 FEC overhead is constant and independent of the packet loss (since FEC packets are always transmitted regardless of loss). ARQ overhead will increase with packet loss, but it is lower than FEC overhead until the loss is about 10%. As we will show in the next section, this is past the useful operating point of a FEC system.

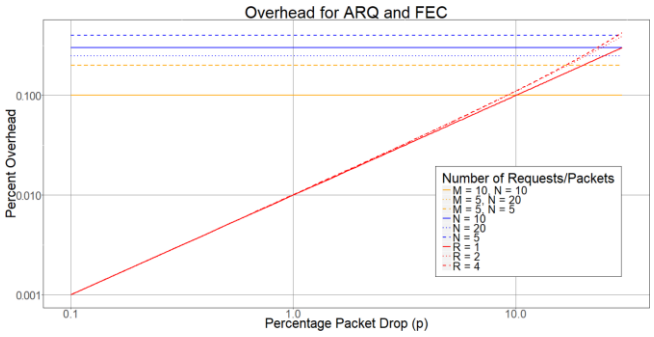

FIGURE 4: PROTOCOL OVERHEAD

SIMULATOR TESTS

In order to evaluate the protocols in a lab environment, we developed a simple network simulator to create a controlled amount of packet loss. This section describes the simulator and the tests performed with it.

I. Description of the Network Simulator

As indicated in Figure 5, the network simulator is simply a software application running on a standard computer. The simulator receives packets from the encoder and resends them to the decoder, after imposing a certain amount of packet loss. At the output of the decoder, there is an actual video monitor connected so the resulting video quality can be subjectively assessed.

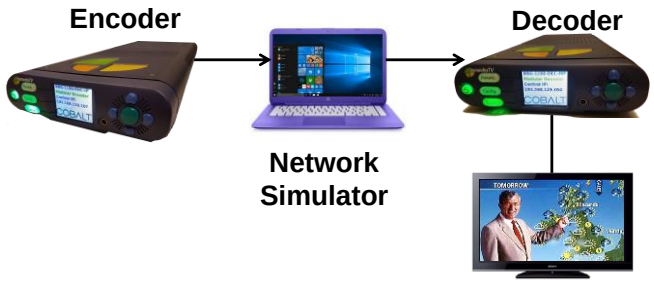

FIGURE 5: NETWORK SIMULATOR TEST ENVIRONMENT

The features of the network simulator developed for this testing are:

- The simulator will receive packets from the encoder and retransmit them to the decoder. It is aware of SMPTE-2022 packets, and will automatically forward them to the decoder if present.
- The simulator will also forward RTCP packets from the decoder to the encoder. This is necessary

for relaying the RFC 4585 NACK packets in ARQ tests.

- When the simulator decides to create packet loss, it will drop one or more consecutive packets from the encoder to the decoder. If the encoder is operating in SMPTE-2022 mode, FEC packets may also be dropped; if the encoder is operating in ARQ mode, retransmissions may also be dropped.
- The number of consecutive packets to be dropped is determined as follows:
  - The simulator is configured with a minimum burst loss, denoted by L_{min}, and a maximum burst loss, denoted by L_{max}, with $1 \le L_{min} \le L_{max} \le 30$.
  - An individual loss drops a random number of packets, uniformly distributed between L_{min} and L_{max}. Therefore, the average packet loss burst is $(L_{min}+L_{max})/2$.
- The actual loss events are generated at random. The simulator is configured with an overall packet loss target, expressed as a percentage of the packets being forwarded. If we denote the desired packet loss probability by p, the loss event probability used by the simulator is given by (9).

$$P_{Loss} = \frac{2p}{L_{\min} + L_{\max}} \tag{9}$$

The simulator was implemented as Windows program on a standard computer.

II. Test Methodology and Results

Our test methodology was:

- Establish an end-to-end real-time flow from encoder to decoder through the simulator using the protocol under test.
- Configure the simulator for a specific burst loss setting.
- Increase the packet loss rate until, in our subjective assessment, the video was no longer "watchable". Our definition of no longer "watchable" was the occurrence of noticeable glitches every few seconds.
- Record this packet loss rate.

Clearly, that the numerical results for this test will be highly dependent on the subjective definition of "watchable", as different viewers will have different standards. However, if all the tests are performed by the same viewer, the relative strengths of the two protocols can be assessed. In other words, the absolute numerical results will be viewer-dependent, but useful information can be derived from their relative values.

The tests were performed under the following conditions:

- Compression type: H.264 High Profile
- Input video resolution: 1920x1080i 59.94
- Elementary video bit rate: 6 Mb/s
- Audio: MPEG-1 Layer II, stereo, 128 kb/s
- Container: transport stream at 6.877 Mb/s

The protocol parameters were:

- ARQ: up to 4 retries per packet allowed
- SMPTE-2022: 20×5 matrix, row and column FEC

For this test, we set $L_{min} = 1$ for all trials, and set L_{max} to different values between 1 and 30.

The test results are displayed in Figure 6, where we plot the packet loss at which the video became "not watchable" as a function of L_{max}. As expected, ARQ produces significantly better results than SMPTE-2022 FEC. The latter breaks down as soon as the packet loss rate goes over about 3%. ARQ, on the other hand, can reliably operate in the 20% packet loss range, and does better with larger (but less-frequent) losses.

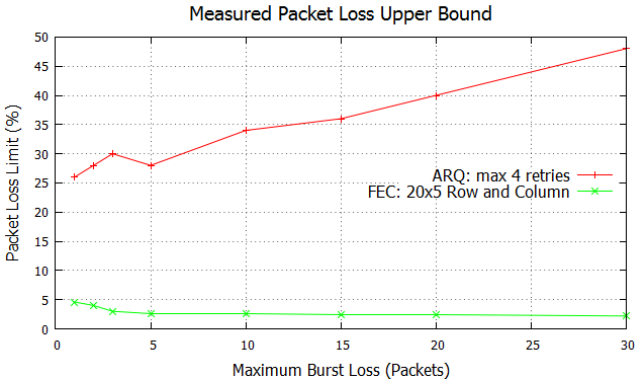

FIGURE 6: SIMULATOR TEST RESULTS

INTERNET TESTS

We conducted one actual field test over the Internet, between two Cobalt Digital facilities. For the tests, we used the normal Internet connections for each of the facilities. These links were also carrying the normal business Internet traffic for each facility. The test characteristics were:

- Endpoints: Santa Clara, CA, and Champaign, IL.
- ISP (both sides): Comcast
- Network round trip time: 75 milliseconds
- Number of hops: 12
- Target bit rate: 3 Mb/s

The tests were performed with actual audio/video data using the same types of encoder/decoder as the simulator test.

I. Initial Link Characterization

The first step was to characterize the link without any packet loss recovery protocol. The results are summarized in Table 3.

Test Duration	25 hours 42 minutes
Total Packets	26,381,219
Dropped Packets	8,187
Packet Loss	0.031%
Packet Loss Instances	2,464
Average Packet Drop	3.3 packets
Max Packet Drop	169 packets
Average Glitch Interval	37.5 seconds
Max Error Free Interval	16 min 17 sec

TABLE 3: INITIAL LINK CHARACTERIZATION

The Table 3 results indicate what appears to be an excellent Internet link, with only a 0.031% packet loss. However, the longest, glitch-free interval for this link was 16 minutes in a 25-hour period. On average, there was a glitch every 37 seconds. The raw performance of this link is, by far, unsuitable for a broadcast application, even though by Internet standards it is a very good link.

It should be also noted that the characteristics of a specific Internet link will change based on time of day and day of the week. What is presented here is simply an average of one weekday.

II. SMPTE-2022 Test Results

The SMPTE-2022 FEC test was performed with a 20×5 matrix, with both column and row FEC. Under these conditions, the protocol can recover a burst loss of up to 20 packets every 100-packet block, with an overhead of 25%. The test results are summarized in Table 4.

Test Duration	65 hours
Test Start Date	05/19/17, 3:50PM
Received Packets	67,185,790
Lost Packets	10.463
Recovered Packets	8,670
Unrecovered Packets	1,793
Network Packet Loss	0.0158%
Corrected Packet Loss	0.0027%
Correction Ratio	83%
Bandwidth Overhead	25%
Network Glitch Interval	1 minute 13 seconds
Corrected Glitch Interval	7 minutes 12 seconds
Protocol Latency	702 milliseconds

TABLE 4: SMPTE-2022 FEC RESULTS – INTERNET LINK

III. ARQ Test Results

The ARQ test was performed with a latency setting of 4 times the round-trip delay, thus allowing for each dropped packet to be retried up to 4 times. The test results are summarized in Table 5.

Test Duration	169 hours
Test Start Date	05/24/17, 12:30PM
Received Packets	173,490,315
Lost Packets	44,606
Recovered Packets	44,471
Unrecovered Packets	135
Network Packet Loss	0.0257%
Corrected Packet Loss	0.000078%
Correction Ratio	99.7%
Bandwidth Overhead	0.027%
Network Glitch Interval	46 seconds
Corrected Glitch Interval	4 hours 7 minutes
Protocol Latency	400 milliseconds

TABLE 5: ARQ RESULTS – INTERNET LINK

IV. Discussion

As predicted by both the analysis and the simulator test, ARQ is far superior to SMPTE-2022 FEC concerning packet recovery capabilities, and this is achieved with a much lower overhead. This is not a surprise, since SMPTE-2022 was not specifically designed to provide error-free operation over the Internet, but rather as an additional reliability layer for dedicated links (to recover from occasional packet losses). However, until the RIST group in the Video Services Forum completes its protocol work, it is the only low-latency IP transport that is a standard.

In the tests shown here, ARQ latency is also lower than that of SMPTE-2022, but this is more an artifact of the relatively low bit rates used, coupled with the fact that the round-trip delay in the test network was not very high. SMPTE-2022 latency is independent of the network and is inversely proportional to the stream rate, while ARQ latency will be a function of the network latency. Which method has lower latency will be highly dependent on the environment.

CONCLUSIONS

Broadcasters can use the Internet today as a low-cost contribution link for some applications. The questions to be asked are:

1. Can the application accept occasional glitches, at the rate of one every few hours?
2. Can the application accept a transport latency on the order of several hundred milliseconds?

If the answer to both questions is yes, then indeed the application is a good candidate for transport over the Internet, and the advantage is a lower operational cost when compared to dedicated IP links. As shown in this paper, the protocols exist to support this application. SMPTE-2022 can potentially be used if the links are of excellent quality and overhead is not an issue. Alternatively, one of the ARQ variants can be used for a wider variety of links, still with acceptable latency. At the time of this writing, the ARQ

implementations in the market are not interoperable, but a standard is being developed [8].

For applications where latency is not a concern, a transport such as HLS or DASH can be considered as well. These will have several seconds of latency, but the protocol is a standard nonetheless and the packet loss performance will be very good for most networks.

Two aspects not covered on this paper that are important on an actual deployment are:

- Content protection and security: depending on the content, it may be necessary to use end-to-end encryption.
- Firewall integration: encoders and decoders will be behind firewalls, which need to be opened enough for the protocol to operate, or, alternatively, a VPN must be provided end-to-end.

REFERENCES

[1] Pantos, R., and May, W., "HTTP Live Streaming", *RFC 8216*, August 2017.

[2] ISO/IEC 23009-1, "Information Technology: Dynamic Adaptive Streaming over HTTP (DASH)", second edition, 2014-05-15

[3] SMPTE 2022-1-2007, "Forward Error Correction for Real-Time Video/Audio Transport Over IP Networks"

[4] SMPTE 2022-2-2007, "Unidirectional Transport of Constant Bit Rate MPEG-2 Transport Streams on IP Networks"

[5] Halsall, F., *Data Communications, Computer Networks and Open Systems*, Fourth Edition, 1996, pp 169-170

[6] Noronha, C. and Jia, F., "Live Video Communication over Computer Networks Using MPEG," *Combined Industry, Space and Earth Science Data Compression Workshop*, Snowbird, Utah, April 4, 1996

[7] Ott, J., Wenger, S., Sato, N., Burmeister, C., and Rey, J., "Extended RTP Profile for Real-time Transport Control Protocol (RTCP)-Based Feedback", *RFC 4585*, July 2006

[8] Video Services Forum, "Reliable Internet Stream Transport "RIST" Activity Group", retrieved from http://www.videoservicesforum.org/RIST.shtml