



WHITE PAPER

SUBJECT: The Many Ways to Stream Video Using RTP and RTSP

DATE: February 1, 2011

AUTHOR: Howdy Pierce

When streaming video across an IP network, the Real-time Transport Protocol (RTP) represents an alternative system layer to the better-documented MPEG-2 transport stream. The goal of this white paper—inspired by a post on Cardinal Peak’s company blog—is to explain the four primary ways that video and audio can be encapsulated in RTP in a standards-compliant manner.

1. Background

RTP [5] is used primarily to stream either H.264 or MPEG-4 video, plus associated audio, across an IP network. RTP is a system protocol that provides mechanisms to synchronize the presentation different streams—for instance audio and video. As such, it performs many of the same functions as an MPEG-2 transport or program stream.

RTP is codec-agnostic. This means it is possible to carry a large number of codec types inside RTP; for each protocol, the IETF defines an RTP profile that specifies any codec-specific details of mapping data from the codec into RTP packets. Profiles are defined for H.264 [7], MPEG-4 video and audio [6], and many more. Even VC-1, the standardized form of Windows Media Video, has an RTP profile [8].

Originally, discovering how to actually use RTP in a way that would make your product interoperable with others was quite difficult. Eventually a group of large companies formed the Internet Streaming Media Alliance, or ISMA. ISMA’s role was basically to wade into all the different options presented in the standards and create meta-standards that tied a number of other existing standards documents together and clarified how to build a working system that will interoperate with other systems.

Today there are four common ways to send MPEG-4 or H.264 video using RTP, each of which follows some relevant standard. The remainder of this document provides an overview of each method.

2. Multicast delivery: RTP over UDP

In an environment where there is one source of a video stream and many viewers, ideally each frame of video and audio would only transit the network once. This is how multicast delivery works. In a multicast network, each viewer must retrieve an SDP file [9] through some unspecified mechanism, which in practice is usually HTTP. Once retrieved, the SDP file gives enough information for the viewer to find the multicast streams on the network and begin playback.

In the Multicast delivery scenario, each individual stream is sent on a pair of different UDP ports—one for data and the second for the related RTP Control Protocol or RTCP. That means for a video program consisting of a video stream and two audio streams, you’ll actually see packets being delivered to six UDP ports:

1. Video data delivered over RTP

2. The related RTCP port for the video stream
3. Primary audio data delivered over RTP
4. The related RTCP port for the primary audio stream
5. Secondary audio data delivered over RTP
6. The related RTCP port for the secondary audio stream

Timestamps in the RTP headers can be used to synchronize the presentation of the various streams.

As a side note, RTCP is almost vestigial for most applications. It's specified in RFC 3550 [5] along with RTP. If you're implementing a decoder you'll need to listen on the RTCP ports, but you can *almost* ignore any data sent to you. The exceptions are the sender report, which you'll need in order to match up the timestamps between the streams, and the BYE, which some sources will send as they tear down a stream.

Multicast video delivery works best for live content. Because each viewer is viewing the same stream, it's not possible for individual viewers to be able to pause, seek, rewind or fast-forward the stream.

3. Unicast delivery: RTP over UDP

It is also possible to send unicast video over UDP, with one copy of the video transiting the network for each client. Unicast delivery can be used for both live and stored content. In the stored content case, additional control commands can be used to pause, seek, and enter fast forward and rewind modes.

Normally in this case, the player first establishes a control connection to a server using the Real Time Streaming Protocol, or RTSP [4]. In theory RTSP can be used over either UDP or TCP, but in practice it is almost always used over TCP.

The player is normally started with an `rtsp://` URL, and this causes it to connect over TCP to the RTSP server. After some back-and-forth between the player and the RTSP server, during which the server sends the client an SDP file describing the stream, the server begins sending video to the client over UDP. As with the multicast delivery case, a pair of UDP ports is used for each of the elementary streams.

For seekable streams, once the video is playing, the player has additional control using RTSP: It can cause playback to pause, or seek to a different position, or enter fast forward or rewind mode.

4. RTSP Interleaved mode: RTP and RTSP over TCP

It's not ideal to stream video over TCP. In the event a packet is lost in the network, it's usually worse to wait for a retransmission (which is what happens with TCP's guaranteed delivery) than it is just to allow the resulting video glitch to pass through to the user (which is what happens with UDP).

However, there are a handful of different networking configurations that would block UDP video; in particular, firewalls historically have interacted badly with the two modes of UDP delivery summarized so far.

So the RTSP RFC [4], in section 10.12, briefly outlines a mode of interleaving the RTP and RTCP packets onto the existing TCP connection being used for RTSP. Each RTP and RTCP packet is given a four-byte prefix and dropped onto the TCP stream. The result is that the player connects to the RTSP server, and all communication flows over a single TCP connection between the two.

5. HTTP Tunneled mode: RTP and RTSP over HTTP over TCP

You would think RTSP Interleaved mode, being designed to transmit video across firewalls, would be the end, but it turns out that many firewalls aren't configured to allow connections to TCP port 554, the well-known port for an RTSP server.

So Apple invented a method [1] of mapping the entire RTSP Interleaved communication on top of HTTP, meaning the video ultimately flows across TCP port 80. To my knowledge, this HTTP Tunneled mode is not standardized in any official RFC, but it's so widely implemented that it has become a de-facto standard.

6. References

The following standards documents offer more information about the topics discussed in this memo:

- [1] Apple, Inc., "Tunneling RTSP and RTP over HTTP", available at <http://developer.apple.com/quicktime/icefloe/dispatch028.html>
- [2] Internet Streaming Media Alliance Implementation Specification, version 1.0, August 28, 2001
- [3] ISO/IEC 14496-10, "Information technology — Coding of audio-visual objects — Part 10: Advanced video coding", Fourth Edition, September 15, 2008. (More commonly known as the H.264 specification.)
- [4] RFC 2326, "Real Time Streaming Protocol", H. Schulzrinne, A. Rao, R. Lanphier, April 1998, available at <http://www.rfc-editor.org/rfc/rfc2326.txt>.
- [5] RFC 3550, "RTP: A Transport Protocol for Real-Time Applications", H. Schulzrinne, S. Casner, R. Frederick, V. Jacobson, July 2003, available at <http://www.rfc-editor.org/rfc/rfc3550.pdf>.
- [6] RFC 3551, "RTP Profile for Audio and Video Conferences with Minimal Control", H. Schulzrinne, S. Casner, July 2003, available at <http://www.rfc-editor.org/rfc/rfc3551.pdf>.
- [7] RFC 3984, "RTP Payload Format for H.264 Video", S. Wenger, M.M. Hannuksela, T. Stockhammer, M. Westerlund, D. Singer, February 2005, available at <http://www.rfc-editor.org/rfc/rfc3984.txt>.
- [8] RFC 4425, "RTP Payload Format for Video Codec 1 (VC-1)", A. Klemets, February 2006, available at <http://www.rfc-editor.org/rfc/rfc4425.txt>.
- [9] RFC 4566, "SDP: Session Description Protocol", M. Handley, V. Jacobson, C. Perkins, July 2006, available at <http://www.rfc-editor.org/rfc/rfc4566.txt>.

About Cardinal Peak

Cardinal Peak assists companies in bringing embedded products to market on a short timetable and with very high certainty. We have deep expertise, specifically with building products having to do with video, audio, or signal processing. We offer a full list of services, including technology architecture consulting, high-level requirements definition, hardware design, embedded software design, system software design, QA and documentation. We're 40+ people, based outside Boulder, CO, and we have been in business for over 9 years.

Clients typically engage Cardinal Peak for one of three reasons: To assist in their business planning process by offering extremely senior engineering insight; to deliver complete products or subsystems and offload their internal management bandwidth; and to augment their internal teams with qualified staff members.

You can read more about Cardinal Peak at our [web site](#).

About Howdy Pierce

Howdy Pierce's technical background is in multimedia systems, software engineering and operating systems. Mr. Pierce co-founded Cardinal Peak in 2002. From 1999 to 2002, Mr. Pierce co-founded and served as CEO of Vantum Corporation, a venture-backed startup addressing the market for networked video for enterprise applications: training, meeting recording, video documentation and video surveillance. The company's Video Appliance product was awarded Best of Show at its debut at NetworkWorld+Interop in May 2001, and received strongly positive reviews in several industry trade magazines. Prior to Vantum, Mr. Pierce was employed at DiviCom, where he held senior management roles in engineering and system integration. Earlier, Mr. Pierce served in engineering management and engineering roles at Taligent, the Santa Cruz Operation, and Microsoft. Mr. Pierce holds a B.S. in mathematics and writing from Carnegie Mellon University. He is a member of the Board of Advisors for the Computer Science Department at the University of Colorado.