Cobalt Digital Inc. • 2506 Galen Drive • Champaign, IL 61821 USA • 1-217-344-1243 • www.cobaltdigital.com • support@cobaltdigital.com

Introduction

When setting up a compressed video transport over IP, the system designer has a number of available protocol choices. There is no single "best" answer for the protocol selection – the most appropriate choice is a function of the networking environment, system requirements, and target decoders. Identifying this choice can be a challenge.

This white paper presents an overview of the various IP transport protocols (including UDP, RTP, FEC, HTTP, RTMP and HLS), and offers guidance on where to best apply each of them. Throughout this paper, we assume the H.264 (MPEG-4 Part 10) compression, but the concepts are general and can be extended to any compression scheme.

Video Transport Basics

In general, a compressed video transmission system over IP may have the following requirements and considerations:

- Reliable Delivery – basically, every bit in a compressed video stream is important (otherwise, it would have been "compressed out"). Therefore, all the bits must be delivered to the receiver(s).
- Point-to-Multipoint – some video over IP systems may require that the content be delivered to one or a small number of receivers; others may require that the content be delivered to a large number of receivers.
- Latency – some applications require very low latency, while others are insensitive to latency considerations.
- Receiving Device Considerations – some high-end broadcast systems require the use of professional-grade decoders (professional IRDs); others may use consumer-grade set-top boxes, or software decoders, or require transmission to mobile devices.

The Effect of Packet Loss

Reliable delivery is one key point in an IP network. By definition, IP offers "best-effort delivery". This means that IP does not "guarantee" that all the packets will be delivered; it will "try its best", but packets may be dropped or delivered out-of-order. Before we discuss the protocols, it is important to understand the effect of packet loss on a compressed video stream.

One simple way of looking at this issue is to assume that every packet lost will cause a visible glitch in the decoded video. Therefore, we can relate the packet loss to something that is observable. This is illustrated in Table 1. The "acceptable" level of packet loss is somewhat subjective and depends on the application. In this example, a loss of one packet every 1,000,000 (44 minutes between glitches) would probably be acceptable to the average consumer, while the loss of one packet out of 10,000,000 would probably not be good enough for a studio link. The message here is that you need to have very low packet loss for the video to look good.

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. Effect of Packet Loss in a 4 Mb/s Stream

© 2016 Cobalt Digital Inc. All Rights Reserved. Other brands shown are property of their respective owners.

The Latency/Packet Recovery Tradeoff

If the IP network drops a packet, recovering it is a job for protocol. Later on this paper we will discuss exactly how the various protocols go about doing that. However, regardless of the technique used, there is a fundamental tradeoff between a protocol's ability to recover from packet loss, and the latency it imposes into the communication. If a packet is lost, the more time the protocol has to deal with that, the better job it can do. In other words, in order to improve the ability to recover from packet loss, increased latency has to be accepted and accounted for. This happens because video packets have "expiration dates". At some point in time, they need to be played out; after that, they are useless. (See Figure 1.)

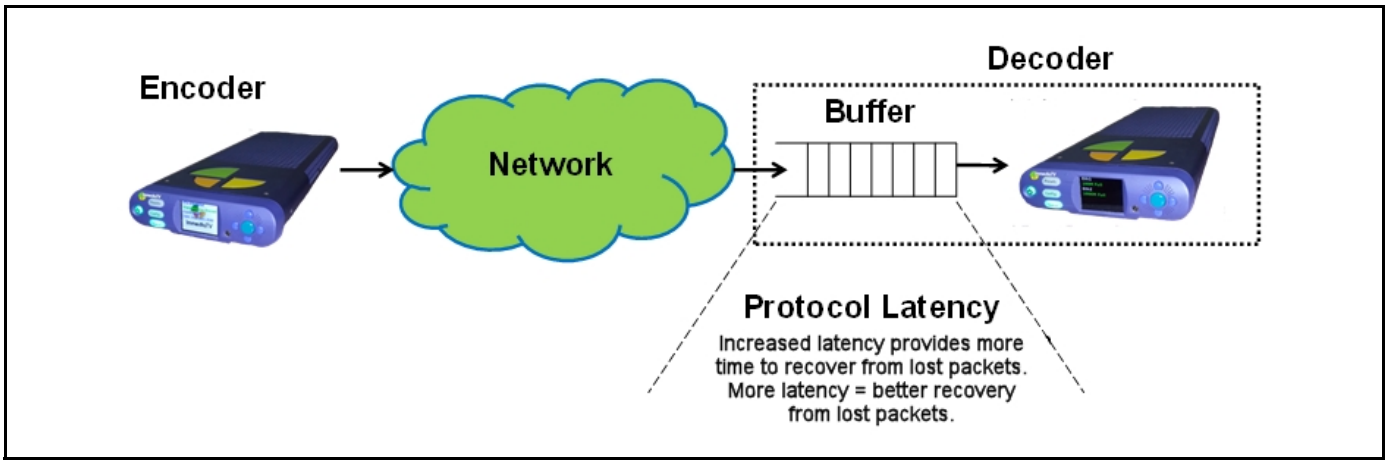

Figure 1. The Latency/Packet Recovery Tradeoff

The Protocol Roadmap

All IP applications today make use of a transport protocol (they do not run directly over IP). The two standard transport protocols are **UDP** (User Datagram Protocol) and **TCP** (Transmission Control Protocol). One common feature of both protocols is that they support application-level multiplexing, i.e., they can deliver multiple flows to the same receiver and keep them separate.

The main characteristics of **UDP** are:

- "Raw" network service – UDP packets can be dropped or delivered out-of-order. There is no indication from the network for either of these events.
- Packets are delivered as soon as possible (no latency).
- Support for IP Multicast – a packet can be given a special group address, and the network infrastructure will replicate it as needed to deliver to all receivers that are subscribed to that group.

The main characteristics of **TCP** are:

- Reliable delivery – TCP will ensure that every bit of data is delivered correctly, and in order.
- Unbounded latency – the protocol may impose arbitrary (and variable) delays on the data to ensure correct delivery.
- Flow-control – the protocol has end-to-end flow control (which can be undesirable when sending compressed video, because generally there is no way to flow-control an encoder from the network).
- One-to-one communication – there is no support for one-to-many communication using TCP. The only way to do that is to create multiple one-to-one TCP flows and send the same data multiple times.

© 2016 Cobalt Digital Inc. All Rights Reserved.

All the video transport protocols run on top of either TCP or UDP, and inherit the features and limitations of the underlying transport protocol. In this paper, we will consider the following video transport protocols:

- UDP-based protocols:
  - Raw UDP
  - RTP
  - RTP plus SMPTE 2022 FEC
- TCP-based protocols:
  - Direct HTTP
  - RTSP
  - RTMP
  - HLS (and other similar protocols, such as MPEG-DASH).

The video transport protocol roadmap is illustrated in Figure 2.

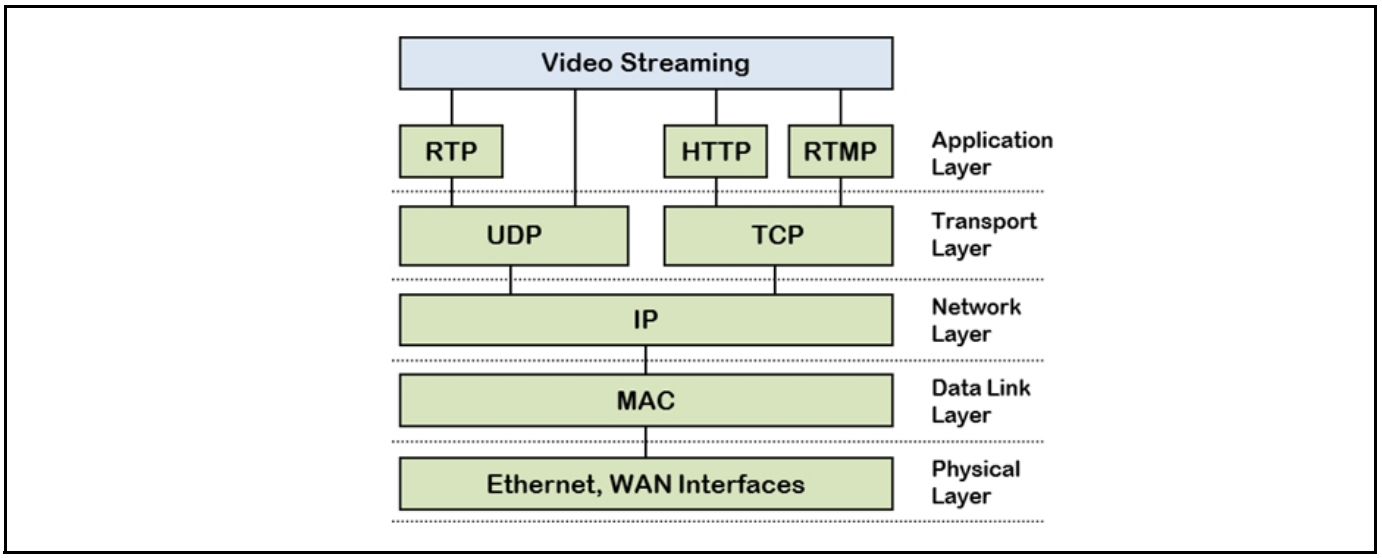

Figure 2. Video Transport Protocol Stacks

© 2016 Cobalt Digital Inc. All Rights Reserved.

UDP-Based Video Transport

Raw UDP

This is the simplest form of compressed video transport over IP: Simply divide the bitstream into packets and send them directly over UDP. Since UDP has no packet recovery or re-ordering capabilities, this can only be used in very clean and well-managed networks, and will definitely not work over the Internet as it exists today. On the other hand, the protocol latency is negligible (because it does not do anything about packets lost).

Raw UDP Summary	
Protocol Latency	Negligible
Packet Loss Recovery	None
Decoder Support	Professional IRDs, Cobalt® 9990-DEC-MPEG, consumer set-top boxes, software decoders
Point-to-Multipoint	IP Multicast

RTP

The Real-Time Protocol (RTP) is a thin layer over UDP, adding primarily time-stamps and sequence numbers. The protocol itself has no packet loss recovery mechanisms, but the sequence numbers allow a receiver to fix out of order packets and detect packet loss. Since it has no recovery mechanisms, its intrinsic latency is the same as UDP.

It should be stressed that out of order packets can only happen in networks where there are multiple paths between the encoder and the decoder(s), and these paths are used simultaneously. A special case of this is when multiple links between two locations are "bonded" to appear as a higher-bandwidth link. Note, however, that modern networking equipment providing this function typically re-orders packets, thereby preserving packet order.

RTP Summary	
Protocol Latency	Negligible
Packet Loss Recovery	No recovery; can handle out-of-order packets
Decoder Support	Professional IRDs, Cobalt® 9990-DEC-MPEG, software decoders
Point-to-Multipoint	IP Multicast

RTP plus SMPTE-2022 FEC

SMPTE-2022 Forward Error Correction (FEC) is a mechanism that can be added to RTP to handle a certain level of packet loss. The idea is to send a certain amount of redundant information together with the main audio/video stream; if some packets are lost, they can potentially be reconstructed at the decoder by using the received packets and the redundant information. This is somewhat different from the traditional FEC used in RF channels for correcting bit errors.

In a modern IP network, the primary cause of packet loss is congestion – too many packets attempting to go out through a link of limited bandwidth. When congestion happens, the network typically drops multiple back-to-back packets. The SMPTE-2022 FEC algorithm is designed to handle this situation. It uses a packet matrix arrangement that allows the recovery of a block of dropped back-to-back packets; the number of columns in the matrix determines the maximum number of dropped back-to-back packets, and the total size of the matrix determines how often this recovery can take place. By tuning the dimensions of the matrix, the user has a certain amount of control of the tradeoff between latency (which is proportional to the size of the matrix), overhead (inversely proportional to the number of rows), and packet recovery capability. This is illustrated in Table 2.

© 2016 Cobalt Digital Inc. All Rights Reserved.

Columns	Rows	Recovery Capability	Overhead	Latency @ 2 Mb/s	Latency @ 10 Mb/s
5	5	5 pkts every 25	20%	263 ms	53 ms
10	5	10 pkts every 50	20%	526 ms	105 ms
20	5	20 pkts every 100	20%	1052 ms	211 ms
10	10	10 pkts every 100	10%	1052 ms	211 ms

Table 2. SMPTE 2022 Parameter Examples

In some situations, it may be possible to run an RTP+SMPTE 2022 FEC video transport over the Internet. Whether or not this works depends on the level of packet loss – if it is low enough, it can work (but there is always a risk of uncorrected packets). One situation where it has a chance of working is when the same ISP is used on both ends of the connection, and that ISP has a backbone with decent capacity (or is willing to provide QoS guarantees).

RTP plus SMPTE 2022 Summary	
Protocol Latency	Low to medium – depends on settings and bit rate
Packet Loss Recovery	Can recover a block of dropped packets
Decoder Support	Professional IRDs, Cobalt® 9990-DEC-MPEG
Point-to-Multipoint	IP Multicast

TCP-Based Video Transport

Direct HTTP

Direct HTTP is the simplest form of video transmission over TCP. The encoder is a server, and the decoder actively connects to it. Once the connection is established, the decoder issues an HTTP GET command, and the encoder responds with the bitstream. The latency and packet loss reliability of this method are those intrinsic within the underlying TCP.

It should be noted that TCP translates "lack of bandwidth" or "excessive packet loss" into "flow control". Since a CBR encoder cannot be flow-controlled, the equipment has a buffer at the sending side, and if the data cannot be sent fast enough, it will be dropped at the encoder side.

Direct HTTP Summary	
Protocol Latency	Medium to high – depends on network conditions
Packet Loss Recovery	Very good, as long as there is end-to-end bandwidth
Decoder Support	Cobalt® 9990-DEC-MPEG, software decoders
Point-to-Multipoint	Multiple connections, poor scalability (scales at the encoder)

© 2016 Cobalt Digital Inc. All Rights Reserved.

RTMP

The Real-Time Messaging Protocol (RTMP) was originally designed by Macromedia as part of their Flash product. Macromedia was later acquired by Adobe, who placed the protocol specification in the public domain. In its most common form, a client opens an RTMP connection to a server (typically a large computer) to retrieve a live or stored stream. Additionally, the protocol has an option to allow the client to publish a stream to the server; this is what encoders use and is the de-facto Internet standard for pushing a live stream to a Content Distribution Network (CDN). This is illustrated in Figure 3. A server tasked with receiving live streams from encoders is typically called the "Origin Server".

RTMP is similar to Direct HTTP in the sense that the bitstream is being encoded to a TCP connection. However, in this case the encoder is a client, not a server, and the scalability problem moves to the server - the decoders connect to the server, not to the encoder. The encoder-to-decoder latency can be high, depending on how much processing is done in the server.

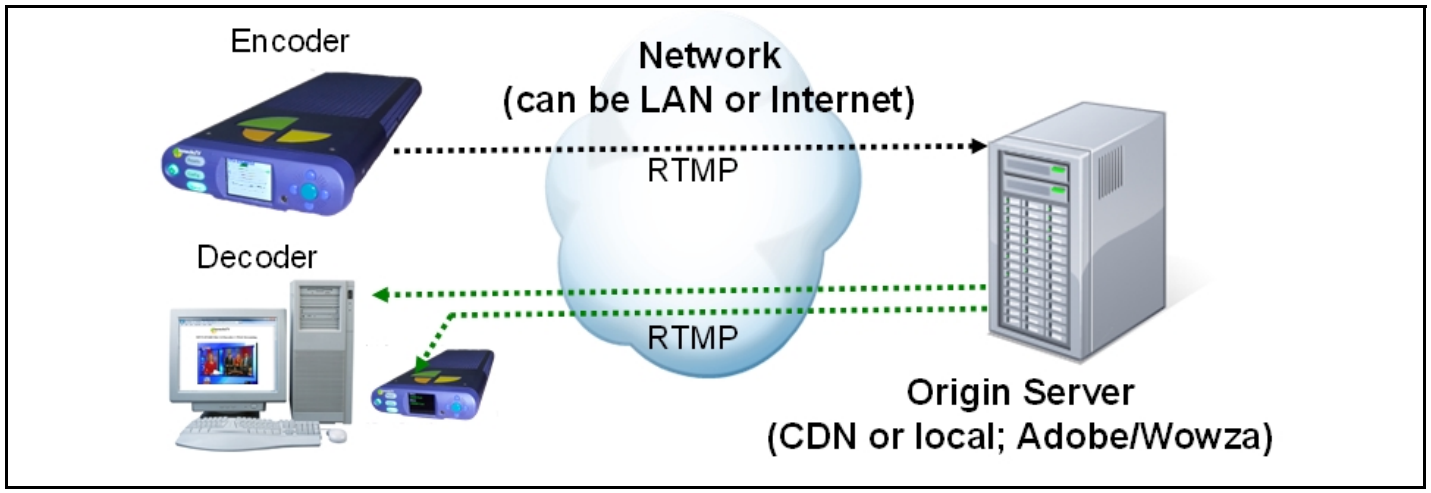

Figure 3 RTMP Operation

RTSP

The Real-Time Streaming Protocol (RTSP) is an Internet standard, and was designed to allow a multimedia client to control a streaming media server. RTSP is defined in RFC-2326. RTSP is primarily a control protocol, allowing the client to start/pause/stop streams. The actual streaming is done using RTP, discussed earlier in this paper.

RTSP runs on top of TCP, and the RTP flow is negotiated as part of the RTSP interchange. The RTSP client can request that the RTP flow be sent over UDP (in which case there is no error recovery - lost packets stay lost) or be send embedded in the TCP control connection used to convey the RTSP messages. In the latter case, the performance of RTSP is the same as the underlying TCP performance.

RTSP can be used in live encoders, and is the most common protocol for IP surveillance cameras with built-in encoders. The latency is the intrinsic latency of TCP.

RTSP Summary	
Protocol Latency	Medium to high - depends on network conditions
Packet Loss Recovery	Very good, as long as there is end-to-end bandwidth
Decoder Support	Cobalt® 9990-DEC-MPEG, software decoders
Point-to-Multipoint	Multiple connections, poor scalability (scales at the encoder)

© 2016 Cobalt Digital Inc. All Rights Reserved.

HTTP Live Streaming (HLS)

HTTP Live Streaming (HLS) is one example of a class of protocols used for Adaptive Streaming – making available multiple bit rate/resolution profiles to a decoder and letting it dynamically chose which one to display. The HLS protocol was designed by Apple with the objective of delivering live streaming using an unmodified web server. Other protocols, such as Microsoft's Silverlight and Adobe's HDS work in a similar fashion. There is an ongoing effort, called MPEG-DASH, to produce a standard protocol for this application. In this paper, we will only discuss HLS, with the understanding that all the other adaptive streaming protocols work in a similar manner.

The protocol works by breaking the bitstream into "chunks", which are placed in a standard web server as files. Each chunk is independently decodable; the chunk size can vary between 1 and 30 seconds, with 10 seconds being a typical number. There is also a "manifest" file that gives the decoder the file names of the current chunks; in HLS, this file is an m3u8 playlist. The decoder first downloads the manifest file, then retrieves the next chunk, plays it, and repeats the process. This is illustrated in Figure 4.

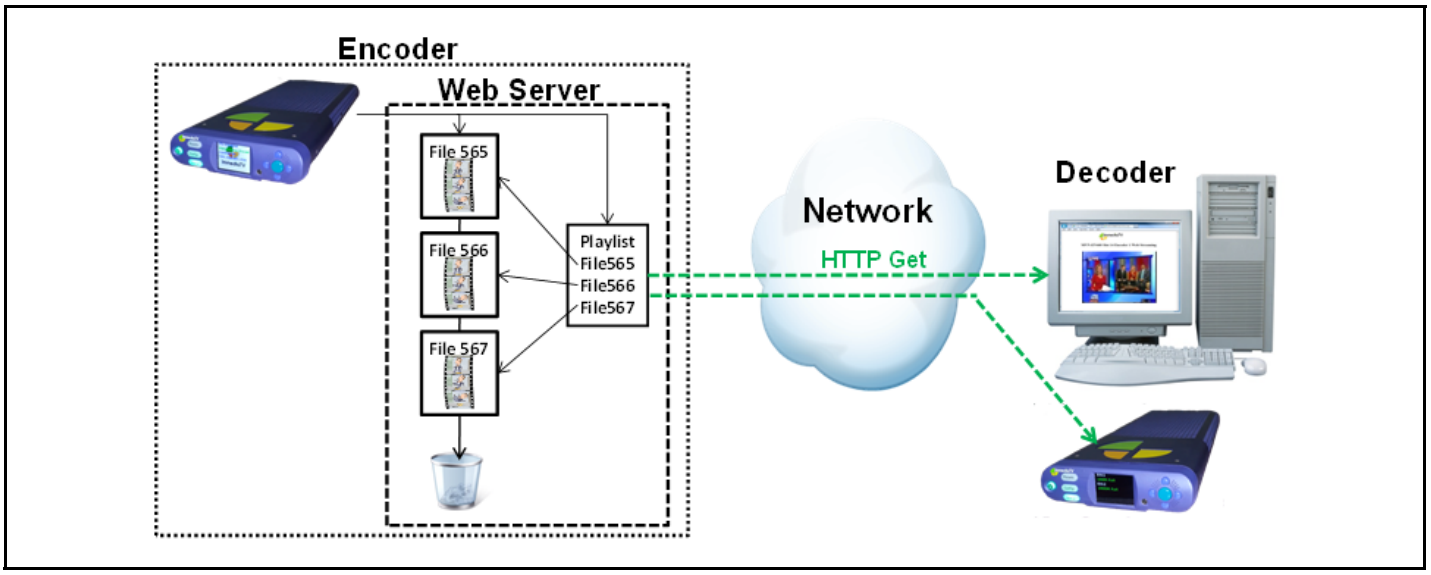

Figure 4 HLS Operation

The web server can be either inside the encoder itself, or external. Scaling to a large number of decoders is achieved simply by scaling the web server. Although HLS uses TCP to deliver the data, it does solve the flow control problem – the encoder is no longer streaming directly on a TCP connection. If there is not enough bandwidth, the decoder will simply fall behind.

It should be noted that HLS is the preferred protocol for mobile devices; it is natively supported in all Apple devices (iPhones, iPads, iPods) and in all Android devices (Android 4.0 or later).

HLS Summary	
Protocol Latency	Very high – 3 to 4 times the chunk duration
Packet Loss Recovery	Very good, as long as there is end-to-end bandwidth
Decoder Support	Software decoders, mobile devices, IP set-top boxes, Cobalt® 9990-DEC-MPEG
Point-to-Multipoint	Multiple connections, scales at the server (good scalability)

© 2016 Cobalt Digital Inc. All Rights Reserved.

Summary

As we pointed out at the beginning of this paper, there is a latency/reliability tradeoff when it comes to delivering video over IP. In order to achieve reliability, a protocol needs time to recover from losses. The various protocols discussed here represent different latency/reliability tradeoffs; these are shown graphically in Figure 5. The way to read this figure is:

- If you have a network that reliably delivers packets (either because it is simple, or because you engineered it this way), you can use a simple, low-latency protocol such as UDP or RTP.
- If, on the other hand, your network is prone to packet loss, or is a network that you do not control (such as the Internet), you will need to use a more sophisticated protocol, and likely have to accept additional latency.

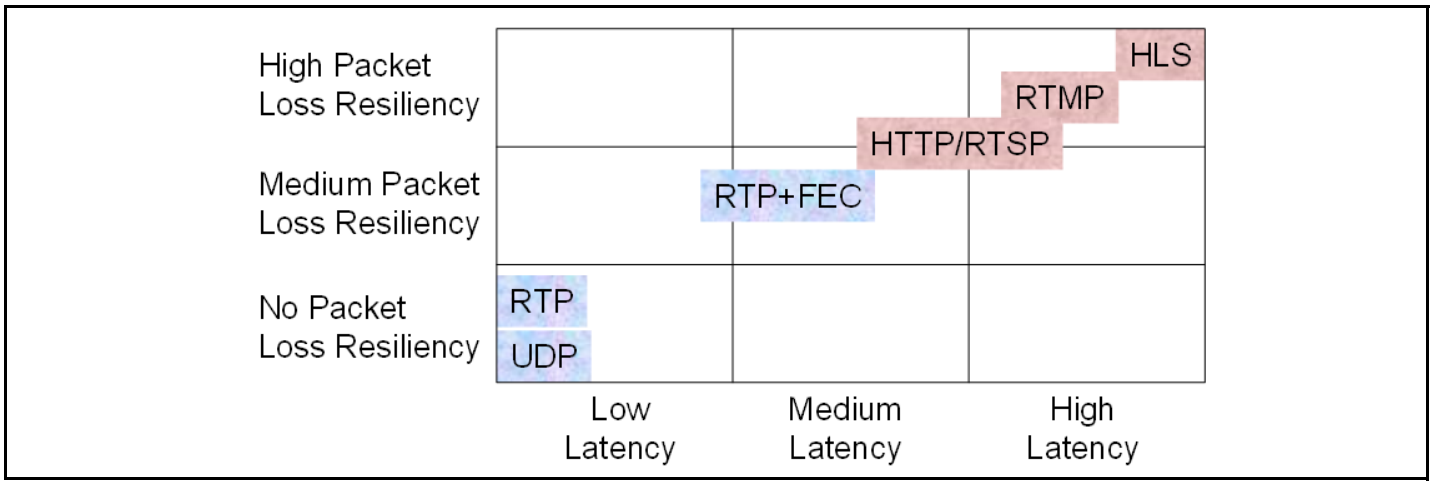

Figure 5 The Latency/Packet Loss Resiliency Tradeoff

Unfortunately, not all decoders support all the protocols. As a matter of fact, is it quite common to start the design from the decoder – a TV studio will require a professional IRD, an IPTV system will require an IP set-top box. This often limits the set of available protocols. The only exception is the Cobalt® 9990-DEC-MPEG series professional decoders, which support almost all of the protocols described in this paper. Figure 6 illustrates the protocols supported by the various decoder classes.

	UDP	RTP	RTP+FEC	HTTP	RTMP	HLS	RTSP
IP Set-Top Boxes	■					■	■
Professional IRDs	■	■	■	Cobalt 9990-DEC-MPEG / BBG-1099-DEC-MPEG			
Software Decoder (VLC)	■	■	■	■		■	■
Flash Player					■		
Mobile Devices					**	■	

** Apple and Android devices do not natively support RTMP (Flash) but there are third party players and browsers that do

Figure 6 Decoder Protocol Support

Using the appropriate Cobalt encoders and decoders (Cobalt® 9223 MPEG-4 Encoder at the encoder side and the Cobalt® 9990-DEC-MPEG Decoder at the decoder side), **all** of the protocols discussed here are supported using the **same** encoders and decoders.

© 2016 Cobalt Digital Inc. All Rights Reserved.