Streaming Live MPEG-4
The VBasics

Mpeg-4 was standardized back in 1999, so it is not really all that new. If you are one of the thousands of organizations using a VBrick to stream MPEG-1 and MPEG-2 over the Internet Protocol, you are already familiar with many of the basic streaming concepts, and no doubt appreciate the simplicity VBrick brings to the video communications proposition. This paper describes MPEG-4 live video streaming for those already familiar with the VBrick products and will assist those who are familiar with popular proprietary streaming products.

MPEG-4 Is A True Standard

It's been said that the nice thing about standards is there are so many to choose from! The true test of a standard is whether it is really multi-vendor. In other words, is it possible to send video using a VBrick and decode it using someone else's player? Are there well-understood rules ("protocols") that all vendors follow?

The answer is yes. Thanks to the Internet Streaming Media Alliance and the robust standard itself, Apple's QuickTime, Phillips' PlatformPlayer, Real's RealOne, and other players decode the MPEG-4 stream produced by a VBrick. VBrick's StreamPlayer offers some unique advantages over other players, but a solution does not

depend on a single vendor solution. Further, Apple Darwin server, Real's Helix server, and others can reflect a VBrick MPEG-4 stream (more on this later).

The important point is that MPEG-4 is a true standard, just like Ethernet or the Internet Protocol itself. And just like these other standards, there is constant refinement and improvements but with the assurance of backward compatibility.

Audio And Video Compression

MPEG-4, like MPEG-2, is organized into various profiles. Each profile attempts to characterize a minimum set of features and functions intended for a particular application.

For example, MPEG-4 defines a "Simple Studio Profile" that is intended for editing studio video with bit rates that reach almost 2 Gbps. MPEG-4 also defines the "Simple Face and Body Animation Profile" intended for computer-generated animation.

Importantly, MPEG-4 describes the "Simple Profile", which currently provides the widest interoperability for streaming video, and operates at bit rates from dial-up speeds to several megabits per second. This profile is the basis for VBXcast MPEG-4 video. While "Advanced Simple" and other profiles exist, VBrick's implementation provides the best solution for live video streaming in a multi-vendor environment.

Equally important is MPEG-4 audio. The popular “MP3” audio is actually “MPEG-1 Layer 3” audio and MPEG-4 audio provides a dramatic improvement to this popular standard and is poised to become the de facto standard for streaming high quality stereo audio with or without MPEG-4 video.

MPEG-1, 2, & 4 Streaming Differences

Both MPEG-1 and MPEG-2 define a multiplexing mechanism such that you send an audio/video stream as a single entity. That is, a MPEG-1 System Stream or a MPEG-2 Transport Stream is sent on a single IP address. A receiver of such a stream may determine how the audio/video is encoded by looking at the stream itself and configuring the appropriate decoder. Everything needed to decode the stream is present in the stream itself. In fact, if you carefully save such a stream to disk, you have a standard MPEG-1 or MPEG-2 file and no further processing is needed.

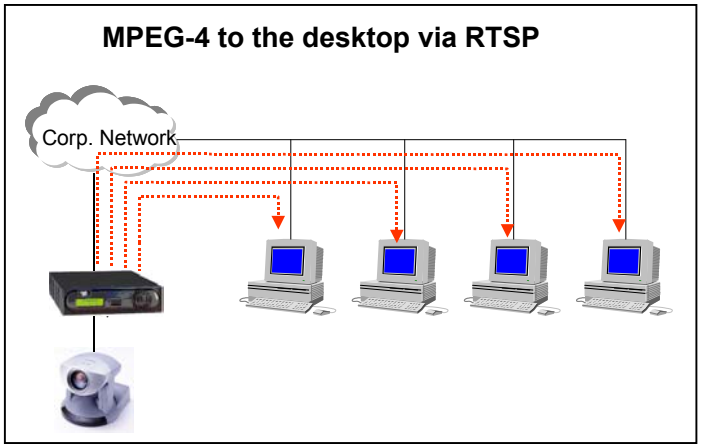
MPEG-4 takes a different path. With MPEG-4, the audio and video are each sent on different IP address ports. There is no notion of a “system stream” at all. This means that a decoder must obtain information about the stream some other way since it cannot learn about the stream from the audio/video itself. The decoder learns about how to decode the stream from the Session Description Protocol (“SDP”). It does this in one of several ways, which we will examine later in this paper.

At first blush, this may seem needlessly complex. However, it provides streaming functionality that is not available with MPEG-1 and MPEG-2. For example, a potential viewer who does not have enough bandwidth to view both the audio and video can receive the audio only.

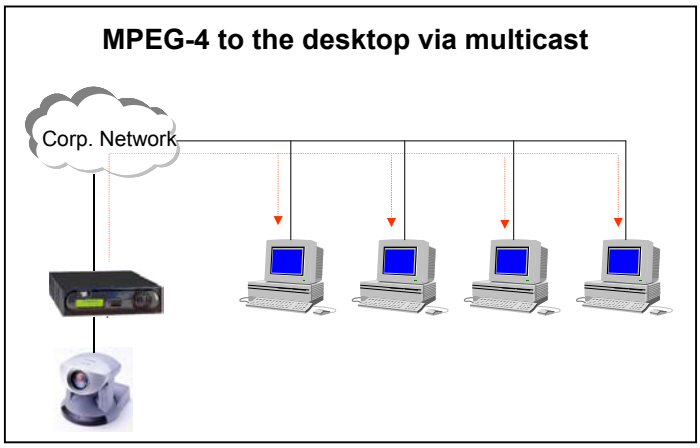
The Streaming Trio

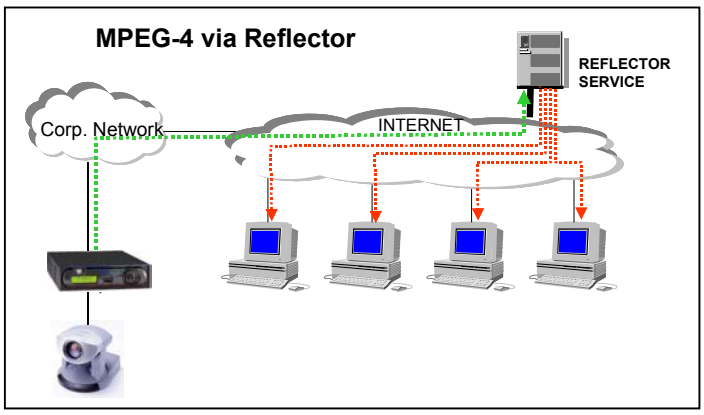
There are three ways to receive a MPEG-4 stream:

1. Direct RTSP – In this case, a VBXcast receives a request from a desktop player. Upon receipt of the request, the VBXcast sends a stream directly to that player. The SDP is automatically delivered to the player as part of the RTSP setup procedure.



2. Multicast – In this case, a VBXcast is continuously sending audio/video via IP multicast on a network. Players simply join the multicast. The SDP is delivered to VBrick





StreamPlayer automatically, but other players will need alternate means to obtain the SDP.

3. RTSP from Reflector – In this case, a VBXcast sends one continuous audio and video stream to a server, and the server replicates the IP packets and delivers them upon request to players. The SDP is delivered to the player from the server as part of the RTSP setup procedure, but the SDP file must be placed on the reflector.

Because a player must have SDP information to play a stream, it is provided in each of the above cases. VBrick makes the generation and delivery of SDP invisible and automatic whenever possible.

The SDP

When a player makes a RTSP request, the VBrick responds to the player with the necessary SDP information automatically. This procedure is defined in the standards, and it occurs automatically and transparently. For example, when you open QuickTime and enter "rtsp://ipaddress/streamname", the VBrick sends the SDP information to the player as part of the startup sequence.

When you send a live MPEG-4 stream via multicast, there is no SDP information present in the stream. So how does the player obtain this necessary information?

If the MPEG-4 stream is viewed via VBrick StreamPlayer, the SDP information is automatically received as part of VBrick's exclusive live Program Guide. From the viewer's perspective it operates the same as when viewing live MPEG-1 or MPEG-2 video, but "behind the scenes" the MPEG-4 decoder is using SDP information.

The SDP may be a file (see the Example SDP File illustration). To create a SDP file, VBXcast provides a simple button in the management interface that saves a SDP file right on your computer. This SDP file contains all of the information necessary to decode the video, including bit rate, resolution, video IP address, audio IP address, and much more. Simply enable and configure your MPEG-4 encoder, then press a button to create a SDP file. Here are some examples where a SDP file is used:

- Configure your VBXcast to multicast and then create a SDP file. Place that SDP file on a web server. Embed a player in a web page and point the player to the SDP file. Viewers may now view the multicast video in their browser. The SDP file and/or the viewing page may be password-protected on the server.

- Configure your VBXcast to unicast directly to a reflector server. Create a SDP and place it on the server. Anyone may view the live stream from the reflector server.

- Configure your VBXcast to multicast, disable the Program Guide, and then create a SDP file. Email the SDP file only to the people you wish to view the stream. This keeps the stream private.

EXAMPLE SDP FILE

```
v=0
o=- 07223011271 1 IN IP4 123.123.123.123
s=VBrick
i=Live MPEG-4 from VBrick Systems
b=AS:248
a=keywds:
a=author:VBrick Systems, Inc.
a=copyright:2003, VBrick
a=streamID:slot1Destination2
a=ISMA-compliance:1,1.0,1
a=mpeg4-iod: "data:application/mpeg4-
iod;base64,AoIvAE/+AQ8BAQOBOgABQJxkYXRhOmFwcGxpY2F0aW9uL21wZWc0LW9kLWF
1O2Jhc2U2NCxBVllCS2dVZkF5WUEzaUFBYndRTklCRUJRQUFBQmR3QUFBTU5RQVlRQUV
RQUFWK1FBQUFBQUNBQUFBQUFBd0VvQXA4REpBQnZBQVFOUUJVQUNBQUFCT0lBQ
UFDN2dBWVFBRVFBQUQ2QUFBQStnQ0FnQUFBQUF3PT0EDQEFAADIAAAAAAAAAAAG
CQEAAAAAAAAAAANpAAJARmRhdGE6YXBwbGljYXRpb24vbXBlZzQtYmlmcy1hdTtiYXNl
NjQsd0JBU2dUQXFCVzBtRUVIOEFBQUIvQUFBQkVLQ0tDbjQEEgINAAAUAAAAAAAAAA
AFAwAAYAYJAQAAAAAAAAA"
c=IN IP4 233.28.209.10/63
m=video 4454/1 RTP/AVP 96
b=AS:200
t=0 0
a=rtpmap:96 MP4V-ES/90000
a=control:trackID=11
a=cliprect:0,0,192,256
a=fmtp:96 profile-level-
id=1;config=000001B003000001B509000001000000012000844007A84020C0A31F
a=mpeg4-esid:222
c=IN IP4 233.28.209.10/63
m=audio 4654/1 RTP/AVP 97
b=AS:48
a=rtpmap:97 mpeg4-generic/16000/1
a=fmtp:97 streamtype=5; profile-level-id=15; mode=AAC-hbr; config=1408; SizeLength=13;
IndexLength=3; IndexDeltaLength=3; Profile=1
a=mpeg4-esid:111
a=control:trackID=12
```

- Configure your VBXcast for audio only, and then create a SDP file. Reconfigure your VBXcast for audio and video and create another SDP file. Use the audio-only SDP file for audio-only, and the other SDP file for audio and video.

MPEG-4 – IP Addresses and Ports

An RTSP controlled video and/or audio session involves both TCP and UDP ports. The RTSP protocol uses a TCP connection that is initiated from the client (player) to the server (VBrick encoder).

The default TCP port used for RTSP is 554, so if the VBrick is behind a firewall, this TCP port must be open to allow clients on the Internet to initiate an RTSP request for a stream.

As part of the RTSP negotiation, the client tells the server which UDP ports to use for the video and/or audio streams. These streams are delivered as RTP over UDP, so if the player is behind a firewall these UDP ports must be open to allow the video and audio to reach the player.

Quick Reference Port Numbers Used

TCP	554
UDP	6970 + 4 for each player

The convention used by most players (QuickTime, Real, VBrick StreamPlayer) is to start with port 6970. Each audio/video session uses a group of 4 ports, so the first player on a particular PC will use 6970, 6971, 6972, and 6973 for an audio/video session. If a second player is started on the same PC it will use 6974, 6975, 6976, 6977. The number of players allowed to run on each PC behind the firewall would determine the range of UDP ports that must be open. For example, if 10 simultaneous players were to be allowed, a firewall would need to allow 40 UDP ports (6970 through 7010).

Not all players follow this convention. For example, the Platform4 player from Philips uses ports 12000, 12001, 12002, and 12003 for the first session.

The even port numbers (i.e. 6970 and 6972) are used for the video and audio RTP packets. The odd port numbers are used for RTCP reports that are sent from the server to the client periodically. The client also sends RTCP reports to the server. The VBXcast uses these as a "keep-alive" and the encoder will stop sending the video and audio streams to the client if it doesn't receive periodic RTCP reports from the client. The VBXcast uses the same ports to receive RTCP as the client (i.e. 6971 and 6973) so if the VBrick is behind a firewall these ports may also need to be opened.

Audio/Video Synchronization and RTCP

The astute observer may note that when audio and video are sent via separate streams, there could be a loss of synchronization. To ensure the sync is maintained, VBXcast sends a “RTCP Report” to players on a periodic basis. Basically, the report tells the players that “this” video sequence belongs with “this” audio sequence. In a perfect world, once the audio and video are aligned, they would stay aligned. But in the real world, with transmission errors, packet jitter, and desktop variability, they may drift apart. The VBXcast resends RTCP reports to the player at user-defined periodic intervals to ensure synchronization is always maintained.

Firewalls

Corporate firewalls often block RTSP and/or UDP traffic. This may be done on purpose to prevent streaming video from using too much Internet access bandwidth, or it may be an accidental consequence of firewall default settings.

To enable efficient viewing of MPEG-4, the TCP port 554 and appropriate UDP ports must be opened. Using a reflector server, MPEG-4 may also be tunneled via HTTP, but this approach uses about 35% more bandwidth than UDP and should be avoided where possible (more on this later).

When UDP is blocked by a firewall, the RTSP communication works fine (you can contact the VBrick and request the video) but the actual video and audio streams do not get through. The players will timeout or try HTTP tunneling which the VBXcast does not directly support.

Some home routers have firewall features which prevent viewing UDP MPEG-4 streams over the Internet. One technique for the popular LinkSys home router is to use its DMZ feature, found in the "advanced setup" area. Set the DMZ host address to match the address of the PC on which you wish to run the player. That PC will have all UDP and TCP ports open to the Internet and video can get through.

Another technique on the LinkSys is to use Port Forwarding, found in the "advanced setup" area. For Real and QuickTime you should set a

port range of 6970-6983. Check the UDP box, enter the IP address of the PC with the player, check the enable box, and press the button to apply the settings. (The Application Name is just for reference - no need to set.)

Yet another technique for the LinkSys is to use Port Triggering. There is a Port Triggering button on the Port Forwarding page in “advanced setup” area. When you click that button you get another table. Again the Application Name is just for reference (you might call it “Streaming Video”). Set the Trigger port range to 554-554, set the Incoming Port range to 6970-6983, and press the button to apply the settings. This tells the router/firewall that when it sees the player go out to the Internet using port 554 (the RTSP port) it should then open the associated incoming port range to allow the video to get to the player. Other routers have similar features that allow you to enable UDP traffic to be received.

MPEG-4 and HTTP

Unfortunately, some IT managers block UDP ports without sufficient forethought. Blocking UDP simply forces viewers to automatically use HTTP, and that uses about 30% more bandwidth per viewer. HTTP is open on virtually all firewalls.

The VBXcast appliance does not directly support HTTP streaming. Rather, VBXcast may be configured to send a single unicast stream to a reflector server, and the reflector server (Darwin, Helix) delivers the stream via HTTP to the viewer.

Embedding MPEG-4 in a Web Page

While you can use VBrick StreamPlayer, Apple QuickTime, RealOne, and other players to view live MPEG-4 streams, there are many advantages to embedding the video in a web page. Among these advantages are:

- The ability to include customized graphics, text, and hyperlinks
- The ability control the size and position of video window
- The ability to obtain viewing demographics from conventional web logs
- The ability to script advanced customized controls

For QuickTime, the first step is to create a reference movie. Apple's "MakeRefMovie.exe" program (freely available from Apple web site) allows you to enter the source address, such as "rtsp://ipaddress/vbrickvideo1") and save the file with a .mov extension. Most computers recognize the .mov extension as a QuickTime movie, and you can treat this .mov file as if it were a video file, when in fact it merely points to a live stream. To embed the video in a web page, simply place the following in the body of your HTML page:

```
<EMBED SRC="myvideo.mov" HEIGHT=192 WIDTH=256 CONTROLLER="false">
```

You can also point directly to a SDP file, which is necessary if you are playing a multicast stream. Here is an example of an embedded

QuickTime player using the <OBJECT> tag that is pointing to a SDP file that is located on the VBrick web site:

```
<OBJECT WIDTH="256"  HEIGHT="192"
CLASSID="clsid:02BF25D5-8C17-4B23-BC80-D3488ABDDC6B">
<param name="src" value="http://www.vbrick.com/mpeg4/uctv/uctv_multicast_Feb23.sdp">
<param name="controller" value="false">
<param name="_ExtentX" value="5768">
<param name="_ExtentY" value="4498">
<EMBED SRC="http://www.vbrick.com/mpeg4/uctv/uctv_multicast_Feb23.sdp" HEIGHT=192
WIDTH=256 CONTROLLER="false">
</OBJECT>
```

Real's RealOne player uses ".ram" files. To view MPEG-4 with RealOne, simply create a text file that points to the stream and save the file with a .ram extension. For example, your file may contain nothing more than "rtsp://ipaddress/vbrickvideo1", and your file name may be "myvideo.ram". You may embed RealOne player and point it to this file. You may also simply create a text hyperlink to myvideo.ram that will cause external RealOne player to launch and play your live video.

The advanced web author may also use SMIL and similar technologies to embed live MPEG-4 video in web pages.