



tutorial

streaming media

magazine

Corporate Streaming with RealSystem iQ

If you're planning to deploy an enterprise streaming system, and hope to keep your CEO smiling and staff headaches to a minimum, RealSystem iQ might be the solution you've been looking for. We'll take you through it step by step. ▼

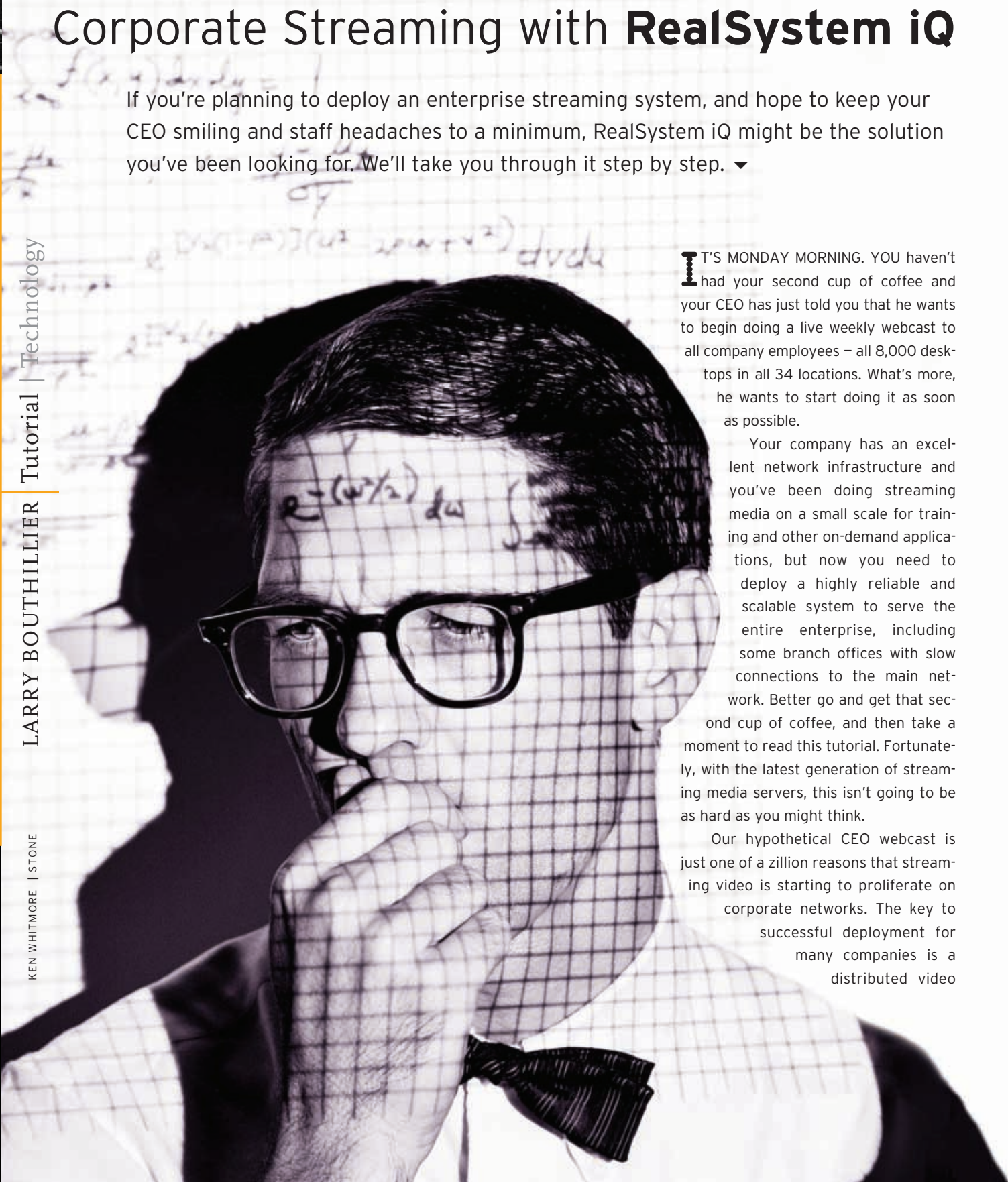
LARRY BOUTHILLIER | Technology Tutorial

KEN WHITMORE | STONE

IT'S MONDAY MORNING. YOU haven't had your second cup of coffee and your CEO has just told you that he wants to begin doing a live weekly webcast to all company employees – all 8,000 desktops in all 34 locations. What's more, he wants to start doing it as soon as possible.

Your company has an excellent network infrastructure and you've been doing streaming media on a small scale for training and other on-demand applications, but now you need to deploy a highly reliable and scalable system to serve the entire enterprise, including some branch offices with slow connections to the main network. Better go and get that second cup of coffee, and then take a moment to read this tutorial. Fortunately, with the latest generation of streaming media servers, this isn't going to be as hard as you might think.

Our hypothetical CEO webcast is just one of a zillion reasons that streaming video is starting to proliferate on corporate networks. The key to successful deployment for many companies is a distributed video



server infrastructure. A distributed architecture can increase delivery quality, provide redundancy and scalability, protect the network and use bandwidth effectively.

There are unique issues involved in setting up a distributed architecture, many of which are easily solved using off-the-shelf software that's part of the new RealSystem iQ platform. First, I'll describe the issues that pertain to corporate streaming media deployments, and then we'll take a look at how RealProxy 8 and RealServer 8's new "Neuralcast" features address these issues.

Issues for Corporate Deployments

What's the one major rule for a successful streaming media architecture? (Humor me and pretend for a moment that there's only one.) It's to get the source server as close to the clients (in a network sense) as you can. You should minimize the number of network hops between your servers and your clients. You'll be maximizing the quality and reliability of delivery while reducing the amount of video running over your network backbone.

Protecting that network is a critical function of your streaming video deployment strategy. Of course, you and I may feel that streaming media, naturally, is the most important kind of network traffic. But some other well-intentioned folks in your company may feel that network bandwidth must be preserved for other functions, such as e-mail, e-commerce systems, and other business applications.

Inevitably, some parts of broad corporate networks are especially vulnerable. Many companies have multiple office locations that are tied to the corporate network via T1, satellite or other limited bandwidth connections. Let's say you have a branch office that has 50 desktops and a full T1 for connectivity to the main headquarters. You'll only support about six viewers of that hypothetical 220Kbps CEO webcast before your T1 is brought to its knees. So it's easy to see that control over bandwidth across that connection is especially critical.

It's also important to manage bandwidth on your company's connection to the Internet. You'll want to control the bandwidth devoted to video traffic that can enter at the company firewall. And it sure would be nice if limiting that bandwidth didn't limit the number of internal users that can view a given video.

If you're streaming live webcasts, you'll have all of the concerns above, plus a few others. You'll want redundancy at every level of your system, from the encoders to the edge servers. And you'll want to be able to easily scale the system to handle any number of users. It just won't do to have the CEO's weekly webcast to the troops fail because a video encoder crashes, a network segment fails, or more users connect than you expected.

It turns out that the newest features of RealSystem iQ – the RealProxy product and the Neuralcast technology built into RealServer 8 – address these issues. It's actually very easy to set up basic handling of these issues using the off-the-shelf capability of Real's products. An overview of these technologies will illustrate how they can be used to create a distributed streaming media infrastructure.

Controlling Bandwidth with RealProxy

RealProxy's key methodology can be summed up as "get once, stream many." It intercepts requests for media streams from RealPlayer clients, retrieves a copy of the requested media, and then streams it to the client. RealProxy keeps the copy around for a while (according to a configuration that you, as system administrator, can control). Subsequent requests for the same content get served from this local cache rather than from across the entire network.

If the content is a live stream, RealProxy acts as a "splitter." The single stream coming in from the remote network becomes the many streams required to support all the users on the local network. Installed on your corporate network, it can solve many of the issues we mentioned above. Sounds

easy, huh? Well, it is easy to set up, but be careful: There are complex operations going on under the hood to make it all work. Let's dig deeper.

First, RealProxy is just a piece of software, similar to RealServer, which installs on ordinary server hardware. You can run it on Linux, Solaris and Windows NT/2000. Installation is simple, and configuration can be done through a Web interface, or by editing the XML configuration file with any text editor. Like RealServer, RealProxy can accept server plug-ins for streaming datatypes other than the standard set, such as MPEG-1, OPTX Screenwatch or Macromedia Flash. If you're going to cache these datatypes, you have to include those plug-ins in your RealProxy installation.

There are two ways to get RealProxy to intercept streaming media requests from clients on your network. The first way is to configure each client to point at the correct proxy server. Within the RealPlayer Preferences there's a tab called "Proxy." You use the options here to tell the RealPlayer which proxy server to point at for all connections. Users in different geographic locations would have different settings here, so you may want to pre-configure your clients to point at the correct server for their location. RealNetworks provides a tool, the RealPlayer Intranet Administrator (RIA), that lets desktop administrators remotely configure the RealPlayers on your intranet.

The other way involves your network switches. If your switch is a Layer 4 switch, you can configure it to automatically redirect all traffic requests on a given port to a given server. In other words, you can tell the router to take all requests on port 554 (the default RTSP port) and automatically send them to the proxy server. This can be handy if your switches support it, since it eliminates the need for any special client configuration at all.

When a client requests a media stream, the RealProxy connects to the RealServer URL that the client was ask-

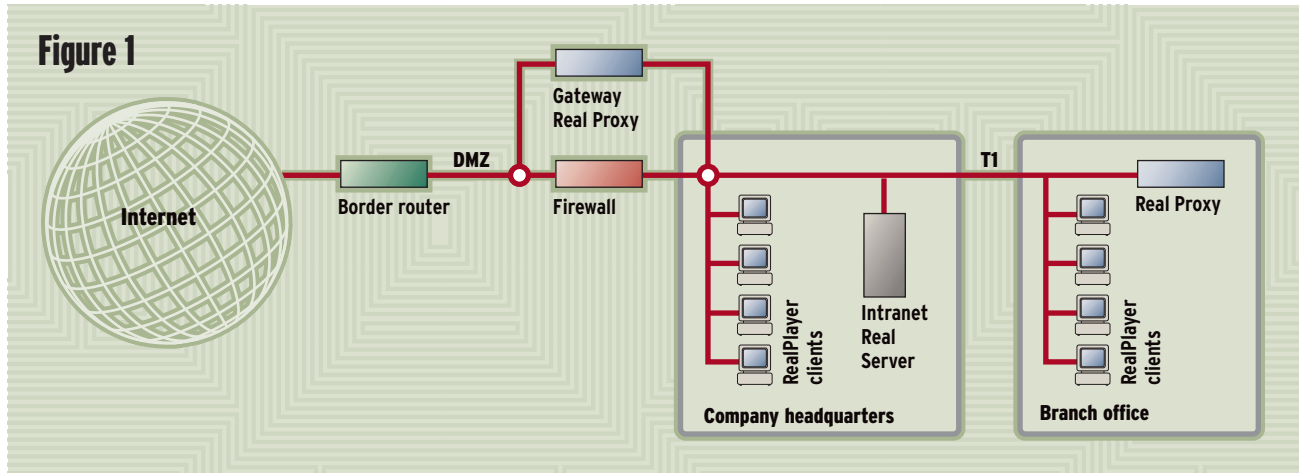


Figure 1:
RealProxy on a corporate intranet.

ing for, and opens a brief conversation with that server. Remember, the owner of that content is going to want to know how many users are viewing it and for how long. The content may be pay-per-view or password-protected, requiring a user to log in to view it. The source server will have a license that allows only a certain number of connections. RealProxy has to honor all of these constraints, so it keeps an accounting connection to the source server open for the duration of the stream, even if the content is actually being streamed from its cache. RealProxy negotiates with the RealServer to pass through any authentication, honor license restrictions, and maintain accurate logging at the source server.

Then, RealProxy looks to see if it already has a copy of the requested media file. If it does, it checks to see if it has changed (or if it has been removed from the source server) since it was cached. If it needs to, it requests a fresh copy and begins streaming that to the client immediately. Now, if you're a content owner, you may be having fits right about now over the thought of all your content being locally cached on proxy servers all over the world (or at least all over your corporate network). To alleviate these concerns, RealProxy encrypts all cached files. Both the file names and the files themselves are encrypted, so all a server administrator can see on disk is a directory full of unreadable files.

Set up at the company firewall, RealProxy lets you control the amount of your company's external network connection available to incoming video content. Configuration settings allow network administrators to restrict the bandwidth allotted to video in various ways. You can set limits on the total bandwidth available for inbound streams from the Internet, for outbound streams from the proxy server into your intranet, or the number of clients on the intranet that may connect. You can also chain several RealProxy servers together, allowing a single stream from the gateway proxy into your intranet to be split into many streams, closer to the edges of our internal network, by other RealProxy installations. Figure 1 illustrates an example deployment of RealProxy on a corporate intranet.

Redundancy and Scalability for Live Webcasts

RealServer 8's Neuralcast technology solves a different problem than RealProxy, although both can be used for live webcast distribution. In addition to facilitating live broadcast distribution, RealServer 8 also includes features that allow redundancy at every level of your streaming video deployment.

The basic idea of Neuralcast is that RealServers can connect to each other in a peer-to-peer manner, with each server receiving and rebroadcasting the streams coming from a "transmitter" server. Now,

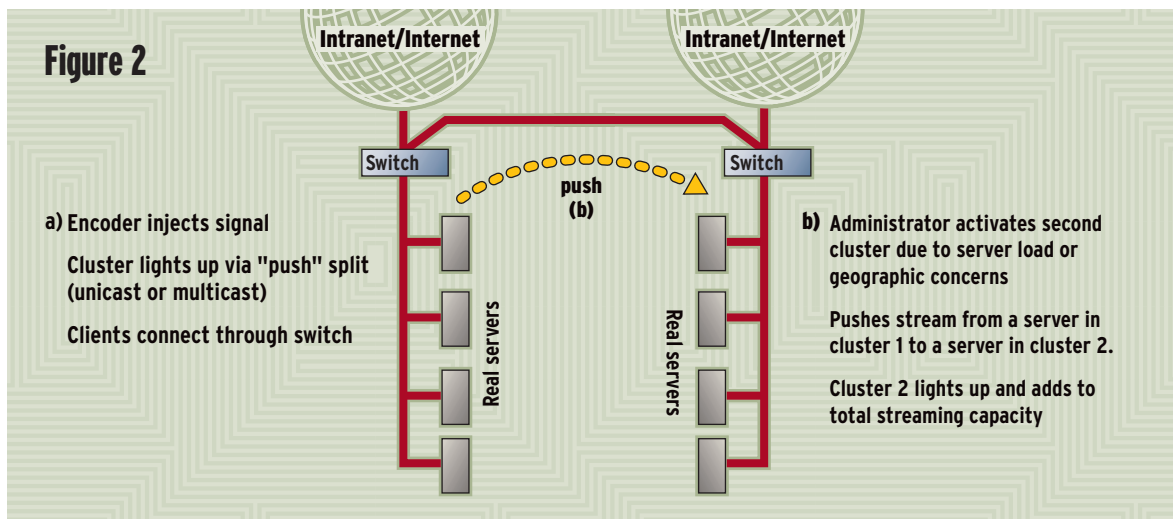


Figure 2:
Scalable network for live video streaming.

receivers and transmitters are just ordinary installations of RealServer. The same system can be both a receiver and a transmitter at the same time, since these are simply roles the software takes on for any given webcast.

This simple concept underlies some radical new ways to distribute video across your network. These transmitters and receivers act as relay stations for your video streams. Instead of a single video server sending bits down the pipe to your clients (and proxies), you can create a web of video servers. Each of these can, in turn, feed other clients and servers. Figure 2 shows how you might deploy a highly scalable network for live video streaming. As the live stream feeds into the server at (a), it uses *push* to immediately send the live stream to all of the servers in the cluster. As load increases, we can use *pull* or *push* to feed a server in the second cluster (b), which in turn lights up the entire second cluster.

The difference between push and pull is one of the differences between a proxy server and the peer-to-peer features of RealServer. A proxy server works on a pull model – the client requests a stream from the proxy server, which responds by pulling the appropriate media from the source server and sending it to the client. With the live broadcast capability in RealServer, you can use the pull model or a push model.

It's easy to set up push in the RealServer configuration. You define a

list of servers to which you want live streams to be pushed, along with the transport protocol to use, the IP ports to use and a security password, among other options. Of course, these receivers need to be configured to receive the push, so you needn't fear that someone's going to hijack your server capacity without your permission. When your live stream starts up on the transmitter server, it's immediately made available to all of the receivers even before a user requests it.

One effect of using push is that it can reduce latency to clients at the edges of the network. But that's small potatoes compared to the big benefits. First, push is a one-way transmission, with no return packets from the receiver expected. It uses forward error correction to send error-correction packets at administrator-defined intervals. If you're pushing your data over a one-way satellite link, you've eliminated the need for the land-based backchannel communications. Secondly, push lets you construct redundant networks of relay servers that have multiple paths to the same destination.

Here's the important point that makes this work: A receiver can listen to the same stream from more than one source at a time. Because RealServer can listen to the same stream coming in from multiple network paths, you can build a highly redundant video distribution system with no single point of failure. Figure 3 shows one example of what

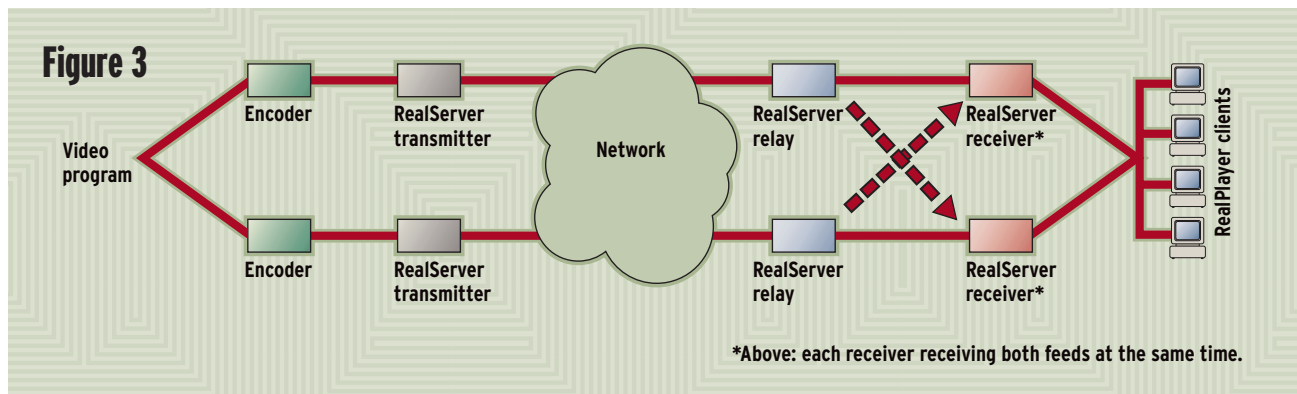


Figure 3:
Redundant video distribution system.

this might look like. A receiver server getting duplicate packets for a single stream will simply throw away any redundant data. In this configuration, a network failure, power failure or system failure at just about any point will not affect playback for viewers.

Take a Deep Breath

So by now, your CEO ought to be very happy with the deployment plan. We've figured out how to get a reliable, redundant stream to the edges of the network, and we're optimizing bandwidth on the company network at every opportunity. In fact, using the tools I just described, you could assemble many different deployment scenarios.

There is some overlap in the feature sets of RealProxy and RealServer 8. Both can provide a way to do live stream splitting and distribution. Both can optimize bandwidth over your network pipes and help your video infrastructure scale. One way to think about the difference is that RealProxy requires no prior relationship at all between the source server and the proxy. But it does require special configuration of the RealPlayer clients or network switches. Then it simply redistributes live or on-demand streams as needed from any source, counting the repeated streams against that source's license. RealServer, on the other hand, can be used to build redundant and self-scaling networks of servers for webcasts across all kinds of network connections, including unidirectional ones.

There's one need that neither product meets at this time – push for on-demand content. Suppose we have an on-demand video (let's say at 220Kbps) that we know will be viewed in all of our corporate branch offices tomorrow morning. Let's also suppose that some of our smallest branch offices have only a dual ISDN connection (at 128Kbps) to the corporate network. We cannot pull that video stream over that connection in real time, so RealProxy will not help us. And with RealServer, we'll have to manually move a copy of the file over to that server beforehand. Then we'll need a way to direct RealPlayers in that office to point at that server to get the video.

The ideal solution here would be something that's a hybrid of the server and the proxy products. We want to push content to the remote servers or proxies during off hours, and have the clients in that office automatically get the local copy when they play the video.

Despite this gap, the capability to set up distributed video delivery systems with off-the-shelf software is a huge win.

See? It's not even lunchtime yet, and you've already got a plan for getting the CEO's face and voice out to those 8000 desktops. Go ahead and take the rest of the day off. *smm*

Larry Bouthillier is Director of Software Development at Harvard Business School and teaches, writes and consults on topics related to creating and managing dynamic multimedia content.