The history of streaming media

What happens when you watch that movie over the internet? **Julian Bucknall** explains

What's covered

▶ STREAMING MEDIA PAST AND PRESENT

These days, everyone knows about streaming video. From cat videos on YouTube to live video broadcasts of important events, the modern online video watcher is well used to viewing video on demand, so much so that the DVD industry is in decline. But now does the whole streaming video process work? What needs to happen between the producer and the consumer?

Recently, a *TechRadar* article revealed that the BBC is going to live-stream 24 live HD Olympic events simultaneously from its sports website, alongside its normal digital broadcasts over the air (http://bit.ly/MEPTtL). Although I'm sure, like me, you have a vague appreciation of what streaming is – after all, watching movies and TV shows over the internet is all part and parcel of the 2012 always-on society – the truth is even more peculiar than you might expect. In a way, it's amazing it works at all.

The earliest reference to what we might recognise as 'streaming media' was a patent awarded to George O Squier in 1922 for the efficient transmission of information by signals over wires. At the time, broadcast radio was just starting up, and required expensive and somewhat temperamental equipment to transmit and receive. Squier recognised the need to simplify broadcasting, and created a company called Wired Radio that used this invention to pipe background music to shops and businesses. Later he decided to ape the Kodak brand name by renaming the company Muzak. This was the first successful attempt to multicast media (that is, transmit one signal over a cable to several receivers simultaneously).

Digital streaming

That was pretty much it for broadcast (radio and TV) and multicast (Muzak) until the age of computers, especially personal computers. It wasn't until the late 1980s or early 1990s that computers had the hardware and software that was capable of playing audio and displaying video.

The main issues that remained were a CPU powerful enough to render video, and a data bus wide enough to transmit video

Spotlight on... Seekpoints and keyframes

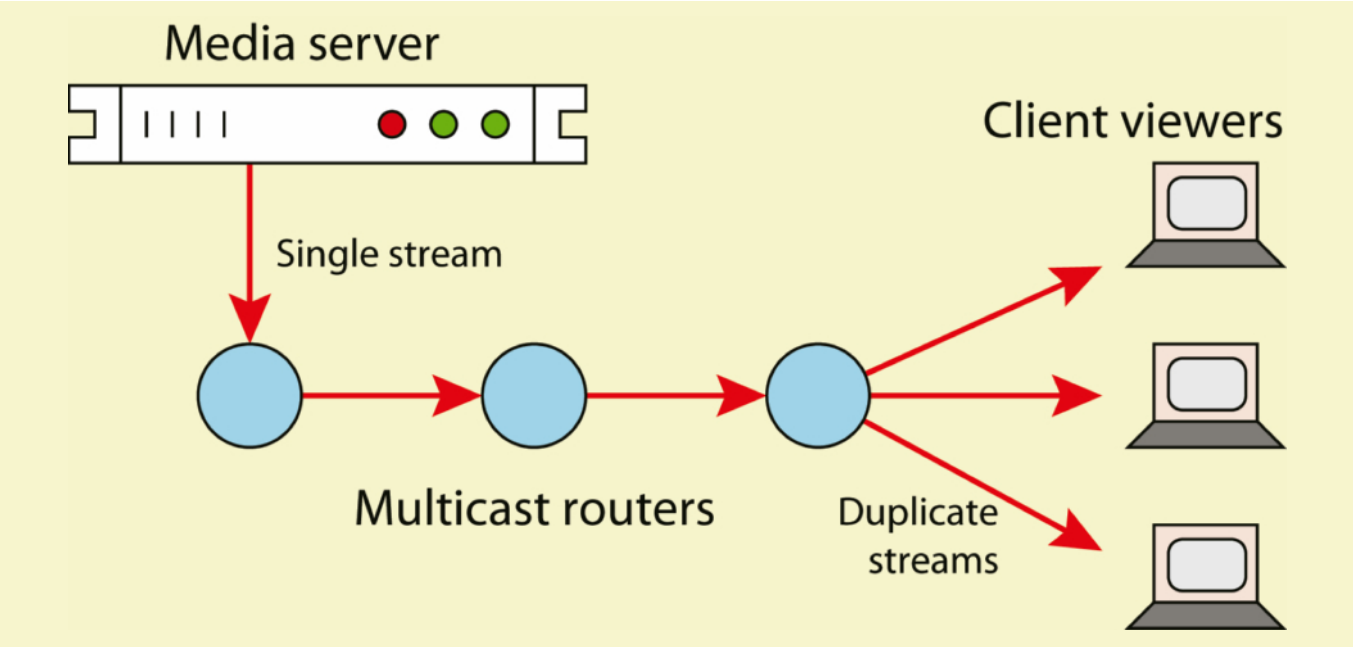
Pseudo-streaming generally allows for the user to pick a particular point from which to continue viewing the video. Since the video is usually highly compressed, there's a problem: the point chosen to seek to is in all probability part of a compressed chunk of data. The user would see garbage and not the actual video.

For this feature to work, the video must be encoded with a list of seekpoints. These seekpoints define a position in the video (either as a byte reference or as an offset time) that define keyframes of the video. A keyframe is a point at which the compression algorithm starts afresh to encode the frames of the video, and thus references a point at which the viewer can correctly load a complete frame and then the subsequent delta frames. If a user selects a particular point, say on the video playback progress bar, the viewer will choose the seekpoint nearest to that selection, and the playback continues from that particular keyframe.

The more seekpoints there are, the finer the granularity of selecting a position to resume video playback, and the less likely it is that the user will notice that the video starts at a slightly different position than that selected. Of course, the more seekpoints there are, the less efficient the compression ratios are and the more data needs to be sent.

The video encodes these seekpoints as metadata at the start of the video file. The viewer will download and store the list before playback commences, thus allowing them to select a playback point at any time.

IPR2025-00252
SportsCastr Inc. EX-2022



▲ **Figure 1: A multicast network distributes media with little bandwidth loss**

data to the video adaptor and monitor, as well as the network bandwidth (this was the age where the best access to networks was through a 28.8Kb modem).

In fact, for a while the only option available was to download the media as a file from some remote server and play it once the file was fully downloaded. Consider the problem: a PC usually had an XGA monitor with a resolution of 640 x 480 pixels at 16 bits per pixel. Video, though, was 320 x 240 pixels. At a video refresh rate of 24 frames per second, the data bus on the PC had to process 320 x 240 x 2 (bytes per pixel) x 24 bytes per second, which works out at about 3.5MB per second.

Several things had to come together before streaming media could happen. First of all, the video itself had to be compressed to reduce the footprint of the media file on disk. At 3.5MB per second, a one minute video would take up 200MB on the hard drive - an amount of space that frankly was not readily available on most PCs of the time. The CPU had to be able to decompress the video data in real time and render frames at the correct frame rate. The data bus of the PC had to be able to handle transferring that amount of data to the video subsystem, and the latter had to be able to refresh the monitor at the correct frame rate. By the mid-1990s, the requisite stars had aligned.

Codecs

A codec is, at its most basic, a compression algorithm for media. The most famous codecs for images are JPEG and PNG, the most popular ones for audio are MP3 and AAC, and video generally uses H.264 and the various MPEG codecs. With regard to streaming, all the video codecs used are lossy compression algorithms. Some are very simple and don't compress too well: in essence each frame of the video is compressed as an individual JPEG image.

The main ones used in streaming, though, use inter-frame compression. Here the codec compresses static parts of the video once (a keyframe), and in subsequent frames only compresses the changing parts of the scene (delta frames – see Figure 2). This type of compression is much more efficient than the simple case, but it does lead to problems when seeking within a video (see 'Spotlight on seekpoints and keyframes' for more information). Newer codecs such as H.264 use different types of delta frames to achieve even better compression. Most codecs are patented and licensed for use.

Multicasting

In 1992, an experimental network was born: the Mbone. This was a virtual network superimposed on the normal internet whose main purpose was multicasting. Multicast in this scenario is a technology that allows data to be streamed efficiently from one server to several receivers simultaneously. An example of a situation that benefits from multicast is an internet radio station. Such a station will present a stream of music data that users can subscribe to, but all users will hear the same stream. From the internet radio station's viewpoint, all it needs is a single low-bandwidth connection to the multicast backbone, and the rest of the transmission and eventual duplication of the data stream is done by the nodes in the internet. Increasing the number of listeners wouldn't impact the internet radio station too much at all.

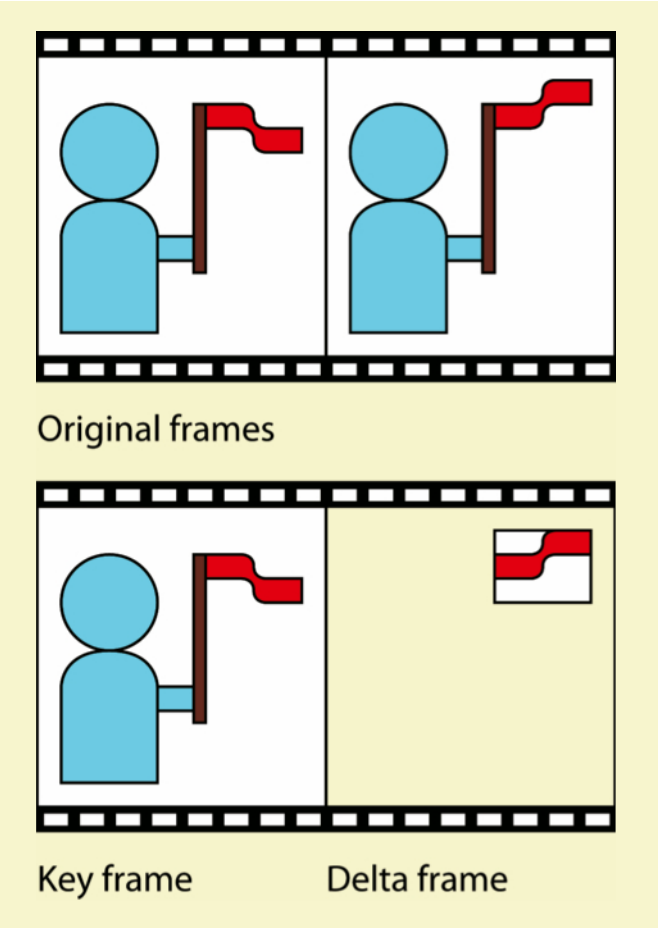
The corresponding technology is known as unicast, and this is what we use when we watch a YouTube video or a movie online: one server sending a data stream over the internet to a single receiver, namely our PC. To continue our example, an internet radio station wouldn't benefit from unicast since it would have to transmit a data stream to every listener. Increasing the number of listeners would require increasing the station's server and network capacities.

The issues with multicasting are several-fold. First of all, it requires special routers as nodes on the network to pass the single data stream on. It has to build up a tree of these special routers, so that it (or the network) can program those routers so that only a single data stream is passed between them. Obviously, only multicast routers can be linked in this tree. This is generally known as tunnelling – the special routers tunnel the multicast data stream between them over the normal internet. Then, each receiver must be able to identify its nearest multicast router so that it can receive a unicast of the data stream from that router. The router acts as a duplicator of data – see Figure 1. The other main issue was touched on in our example of an internet radio station: multicast poses problems with regard to paying for it, especially with regard to ISPs' costs. With a multicast internet radio station, the station's local ISP only passes through a single data stream, regardless of how many listeners there are. The data duplication is done by the routers that are geographically far from the transmitter.

Although Mbone was successful as a research project – it was even used to multicast a Rolling Stones concert at the Cotton Bowl in Dallas – it never really caught on publicly. These days it's mostly used for video conferencing.

Streaming

By the late 90s, streaming video had started to become ▶



▲ Figure 2: Inter-frame compression showing keyframe and delta frame

the norm. Unlike in previous years, where the video had to be downloaded in its entirety before viewing, streaming is characterised by playing the video data as it's received. First, this requires a special compressed video format to facilitate play while downloading. The viewer has to buffer enough data to play should there be some network contention; a few seconds' worth, say. The protocol between viewer and remote media server must allow for renegotiating the resolution of the media should the latency or bandwidth of the network change. If the network latency increases and/or the bandwidth decreases, a lower resolution may be more acceptable than introducing stuttering to the user's playback experience.

Before the turn of the millennium, there were several competing streaming video viewers available. The first was Real Player, which was launched in 1997 and had been demonstrated from 1995. Microsoft implemented streaming video playback in Windows Media Player in 1999, as did Apple with QuickTime. These streaming viewers required websites to install the corresponding media servers in order to provide properly formatted streaming video for playback, and so, for a few years, users had to contend with the possibility of needing to install three incompatible viewers in order to view content.

This state of affairs continued until about 1992 when Macromedia Flash became prevalent. In essence, alongside animation, programmability, games and so on, it provided a multi-platform, multi-browser streaming viewer, free of charge, and free of the vendor lock-in that characterised its predecessors. Flash became so successful that it was available on the vast majority of PCs, and formed the basis of streaming sites, such as YouTube, Vimeo and so on (Netflix uses Microsoft's Silverlight streaming viewer).

Nowadays, there has been a move away from Flash as a streaming viewer; it requires some fairly intensive CPU resources and therefore compromises the battery life of mobile devices such as smartphones and tablets.

Pseudo-streaming

Nowadays, video streaming tends to split into two camps: there's what might be called pseudo-streaming and there's streaming proper. Pseudo-streaming is characterised by downloading an actual file and playing that file as it's being downloaded. YouTube videos tend to be of this variety; you download a video file (and save it temporarily), and play it back during the download. Since the complete file is downloaded, replaying a YouTube video tends to be very quick: there's no more data to download. The file is, however, managed by the viewer and will be deleted once the user moves away to another video.

The media server is different for pseudo-streaming as well. In essence, it operates as a big peer to peer file server: it stores a set of files and will send one as fast as possible to a client requesting it.

Nevertheless, pseudo-streaming allows for seeking to a particular point in the video, without having to download all the video data in between. Pseudo-streaming also uses plain HTTP as a delivery protocol, meaning that it is available on local corporate networks that may block other ports.

Real streaming, on the other hand, is characterised by a data-buffering viewer (all data is kept in memory), with no file being saved on disk. Real streaming also allows for automatic resolution changes (say from 720p to 480p or vice versa) to contend with real-time changes to the network throughput or latency, whereas pseudo-streaming has no such feature. Of course, with some YouTube videos you can elect to view the video in a higher or lower resolution, in which case the video resumes at the changed resolution. For this to work though, the video must have been uploaded at those different resolutions in the first place – the server, in effect, has to store multiple resolution versions of the video.

Media servers that provide real streaming use a different protocol and port to provide video and audio streams. A common protocol used is RTMP (Real-Time Message Protocol, an Adobe standard used by Flash streaming), where the port used is 1935 (HTTP's is 80). There are other variants, including one that tunnels streams through HTTP. There are also other protocols in use such as RTSP (Real-Time Streaming Protocol), which uses RTP (Real-time Transport Protocol) and RTCP (Real-Time Control Protocol). These protocols break up the streams (generally there are more than one, such as a video and an audio channel) into very small packets and then transmits them to the client viewer.

All in all, streaming video has come a long way. Nowadays it's a big part of modern online society, from cat videos all the way to live HD broadcasts of the Olympics. In the audio space it's all Spotify and Pandora, the new individualised internet radio stations. In the future? All that and more. ■

HTML5

Up until two or three years ago, Adobe Flash ruled streaming video. Then came the iPad and, famously, its lack of support for Flash. Instead, Apple championed HTML5 video for playing back video on its tablet.

From that point on, HTML5 video was the new standard and the use of Flash began to decline. In order to support HTML5 video, a user must have an HTML5-compatible browser (all mobile devices have such a browser, and nowadays so do laptops and desktops), and that browser must have installed the codec used to compress the video. At the start, the latter requirement was problematic: Chrome and Firefox chose to implement the WebM codec, whereas Safari and IE9 chose H.264 instead. This meant some issues for video providers since they had to provide two differently-encoded videos, but in reality H.264 is almost certainly going to win those particular wars – most mobile devices use it.

Despite its appeal, HTML5 video still doesn't support much needed commercial technologies, such as adaptive streaming for live content, content protection for premium channels, or playback locking for advertising. Nevertheless, it's certain to become the streaming video viewer of the future.

Page 3