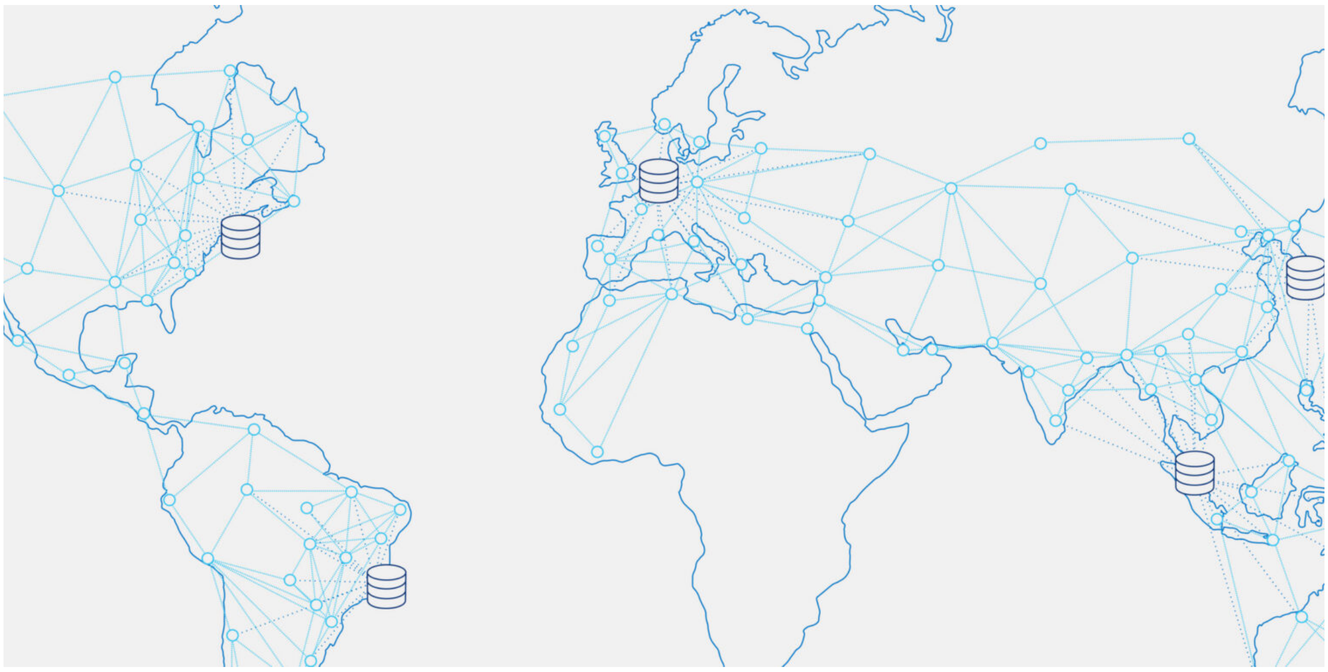


Peer-to-peer video delivery explained: distributed networks from the early internet to the Zettabyte era

By Cyrille Huteau / December 29, 2019

As audiences grow, resolutions increase and quality expectations rise, peer-to-peer content delivery has become a hot topic in the streaming industry. It seems to be an obvious choice; making a solution out of a problem, mesh delivery harnesses traffic peaks – a threat for traditional client-server delivery – to increase capacity and improve viewers' quality of experience. As more and more broadcasters choose to bolster their existing CDN delivery with peer-to-peer technology, we'd like to take a moment to explain what the buzz is all about and how peer-to-peer delivery works.

Peer-to-peer video CDN has been our core business and the center of our technical expertise from the start. Our technology is part of a new generation of WebRTC-based peer-to-peer streaming solutions. But before we talk about modern mesh network delivery, let’s explore how we got here.

Peer-to-peer: from the early days of the internet to the Zettabyte age

Peer-to-peer network architectures are not a novelty; in fact, they have existed since the beginning of the internet. While today client-server models are at the center of the world wide web, it was not always the case. Before it was partitioned, the internet started off as a free, distributed network for sharing information.

As the first network to implement the TCP/IP protocol, *ARPANET* (Advanced Research Projects Agency Network) laid the foundation for the internet in the 1960s. This pioneer network was actually a peer-to-peer one, connecting several American universities including UCLA, the University of Santa Barbara, the University of Utah and Stanford University.

Introduced in 1979, the *Usenet* worldwide distributed discussion system then created the first pre-commercialized internet community, using the *Unix-to-Unix Copy Protoco*l (UUCP). Its news servers operated very similarly to peer-to-peer networks, sharing resources by exchanging them, with no administrator involved.

Early internet protocols like *FTP* and *Telnet* also kept the network largely distributed. While applying a client-server model, they allowed for symmetrical usage patterns by enabling each host to function as both client and server. While not a peer-to-peer protocol on its own, the popular *IRC* chat protocol provided a peer-to-peer service via *DCC* (Direct Client-to-Client) and *CTCP* (Client-to-Client Protocol) extensions. These allowed direct inter-client communications for non-relayed chats and file transfers. Similarly, one of the very foundations of the internet – *DNS* – is based on a distributed architecture; DNS nameservers use peer-to-peer patterns acting as both servers and clients at the same time.

In fact, it was only in the 1980s and 1990s that the cost of the CPU required to serve a growing number of users gave the client-server topology center stage.

Peers Ahoy!

In 1999, *Napster* reignited the distributed internet trend. Along with others that followed, it brought millions of users together to use their increasingly powerful home computers to collaborate and form shared search engines and file systems. In 2000, *Gnutella*, which was accessible via client software such as *LimeWire* and *iMesh*, introduced the next generation of peer-to-peer file sharing. To avoid centralization, it deployed a less efficient method: query flooding, where searches are broadcast throughout the network without requiring an administrating server. 2001 then brought the TCP-based file sharing protocol *BitTorrent* to the world. While its first iteration required central trackers to coordinate between users, later BitTorrent client software managed to avoid centralization with a *distributed hash table* (DHT) used for peer discovery.

Often used for sharing copyrighted content, these networks perverted the original goal of the internet to share resources and data across devices and locations. At this time, peer-to-peer unfortunately became synonymous with piracy.

Peer-to-peer protocols, however, were not used only for illegal file sharing. Many legitimate applications also harnessed distributed architectures to provide better performance for their products and users. Gaming publisher *Blizzard Entertainment*'s use of peer-to-peer networking dates back to the mid 1990s. When *Diablo* was launched in 1996, the game used peer-to-peer for its multiplayer setup: one player acting as a host and the rest as clients. Blizzard's famous *Battle.net* also utilized peer-to-peer for games between hundreds of thousands of concurrent players, with one server responsible for chat and player matching. Later on, Blizzard started using

BitTorrent to distribute updates and patches for games such as *World of Warcraft* via its *Blizzard Downloader*.

Blizzard was (and is) not alone in harnessing peer-to-peer technology for large-scale static file downloads. Other software providers include *Microsoft*, which more recently used peer-to-peer to deliver Windows 10 and Xbox One updates.

In the early 2000s, peer-to-peer design was also at the core of the popular calling application *Skype* (derived from "Sky peer-to-peer"). Requiring no servers, Skype's peer-to-peer protocol allowed millions around the world to communicate over the internet. Though the protocols have changed, Skype is still largely dependant on peer-to-peer, as are many other popular communications apps, notably *Google Hangouts*.

The ins and outs of peer-to-peer video delivery

In the late 1990s peer-to-peer applications were mainly used for content sharing and one-to-one communications. The advent of video streaming in the following decade, however, brought entirely new applications for peer-to-peer.

Requiring more bandwidth than ever before, video streaming created the need for more efficient, scalable distribution. Peer-to-peer technology made perfect sense for this use case for its ability to relieve stress on server infrastructures by connecting viewers watching the same stream at the same time. As audiences began to grow worldwide, it also had significant potential to reduce bandwidth costs and improve quality for viewers by bringing the video source closer to the user.

Peer-to-peer streaming dates back to the late 2000s. One of the first players to take advantage of peer-to-peer architectures for video streaming was Joost, an internet TV provider created by Skype and *Kazaa* founders.

Striking content deals with media giants such as *Viacom*, *Paramount*, *CBS* and *Warner Music*, Joost distributed content via its dedicated desktop player, and later through a Flash-based web player.

Meanwhile, the leading content delivery network (CDN) provider *Akamai* recognized the potential in distributed delivery with its 2007 acquisition of *Red Swoosh*, a peer-to-peer file-sharing company, for $18.7 million. In 2010, this purchase resulted in a new mesh-network based product as part of an exclusive deal to support NFL streaming.

In 2015, Akamai acquired a second company with peer-to-peer offerings, Octoshape. Founded in 2003, the Denmark-based *Octoshape* gained attention when it provided the mesh delivery infrastructure for CNN's coverage of the Obama inauguration.

However promising these peer-to-peer streaming technologies were, they struggled to reach their full potential. Among their biggest limitations, viewers were required to install an additional browser plugin, which was often perceived as intrusive. The plugin hurdle made fewer viewers join the mesh network, which made it naturally less effective.

They may also have been premature: while the demand for online video grew, it had not yet reached the volume we witness today following the mass adoption of smartphones, smart TVs and streaming devices that began in the early 2010s. Nevertheless, these solutions undoubtedly paved the way for a wider market adoption of today's clientless peer-assisted video delivery technologies.

WebRTC: a new age in peer-to-peer video delivery

With ever-growing audiences, a multitude of devices and rising resolutions, video is consuming more bandwidth than ever and is expected to account for over 80% of global IP traffic by 2022.

With this growth, content providers have realized the crucial need for scalable delivery technologies and the limitations of client-server architectures. This is especially true for high profile live-streamed events. When millions of viewers tune in simultaneously, huge traffic spikes are not only costly but can also severely impair quality of experience. The increasing need for a robust video delivery solution, together with the rise of a new technology – *WebRTC* – have ushered in a new age of peer-to-peer streaming solutions.

While HTML5 offered the possibility to render video and audio without a Flash plugin, WebRTC (Web Real-Time Communication) has enabled inter-browser communications and paved the way for a new generation of plugin-free peer-to-peer video delivery technologies. An open-source project released by Google in 2011, WebRTC provides web browsers and mobile applications real-time communication (RTC) capabilities via APIs and therefore enables direct peer-to-peer video and audio communication, without the need to install plugins or download native apps.

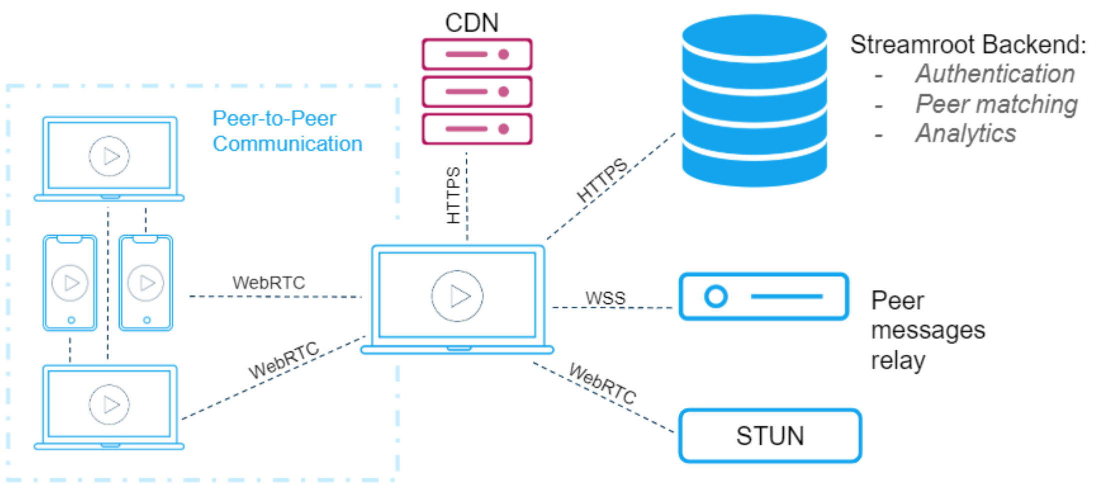
Also available as a C++ library, WebRTC is compatible with *iOS*, *Android* and any other platform. It is supported today by the vast majority of modern browsers, including *Chrome*, *Firefox*, *Opera* and *Safari*. Recently, Microsoft announced that its *Edge* browser will soon be Chromium-based and therefore WebRTC-capable as well.

All of this means that WebRTC-based peer-to-peer video delivery technologies are not only transparent to the end user; with cross-platform compatibilities, they enable much greater proportions of online audiences to participate in peer-to-peer segment exchanges, tremendously increasing performance.

As you can imagine, however, modern peer-to-peer video delivery must also be sufficiently sophisticated to handle the complexity of today's video streaming workflows: HTTP streaming protocols; multi-bitrate streaming; content security and protection mechanisms (tokens, DRM); monetization; and ad insertion solutions, just to name a few. They must also support the multitude of devices that viewers use to stream today, their operating systems and technical capabilities, and work with different ISP configurations across the world. For this reason, WebRTC-based peer-to-peer delivery solutions like Mesh Delivery are often the fruit of years of R&D.

WebRTC-based peer-to-peer video CDN – how does it work?

Contrary to many of the completely decentralized solutions of the past, modern WebRTC-based peer-to-peer streaming solutions are *hybrid systems*. As part of a hybrid delivery network, viewers source video segments from both servers and other viewers watching the same content at the same time. To fully understand how it works, let's take a deeper dive into how our peer-to-peer delivery solution works during a playback session:



When a viewer presses play, the device authenticates via *HTTPS* with our backend, which returns a specific configuration according to the use case and the broadcaster's parameters. The same HTTPS connection is used to pass QoS and traffic data to our customer dashboard for real-time monitoring and analytics.

Our backend also provides the viewer with a list of *peers*: viewers who are currently connected to the same stream, and who can serve as the most efficient sources for obtaining video segments. This list is intelligently selected by our proprietary algorithms based on viewer location, ISP, device and more. It is updated every minute to include the most relevant sources.

At the same time, the viewer’s device prepares to make direct connections with other devices by identifying them through a STUN server and exchanging initial connection information via a relay server. Once this information exchanged, viewers can initiate a direct WebRTC connection to each other and exchange video data.

While the session’s first video fragments are fetched from the origin or caching server to ensure prompt start-up time, the next segments are multi-sourced from the best source available – either from one / several peers or the CDN. Parallel and simultaneous download from multiple sources makes it possible to get video segments from a number of different devices. This allows us to deliver up to 80% of the video traffic via the mesh network and to effectively handle adaptive bitrate use cases.

As a WebRTC-based technology, this solution requires no plugin and is therefore seamless to the end user. Taking it a step further, Mesh Delivery incorporates several additional mechanisms to maximize performance improve streaming quality for viewers. Its peering decisions take into account a variety of variables, including the device type, live or VOD streaming, available memory, users’ internet connection quality and battery status. To avoid affecting user data packages, broadcasters can choose to disable upload for viewers connecting via cellular networks. we also monitor the upload and download speeds of each device and apply congestion control algorithms to best utilize the device’s uplink bandwidth.

Peer-to-peer streaming – the next content delivery go-to solution?

Peer-to-peer has had a long history with its fair share of breakthroughs and obstacles. Today, however, the benefits and necessity of peer-to-peer video delivery are clear, and a growing number of broadcasters are turning to peer-accelerated video technologies to face the delivery challenges of the Zettabyte era. With the intrusiveness of a plugin gone and stigmas fading away, today's peer-to-peer delivery solutions are poised for mainstream market adoption.

In the past year, the industry has shown a tremendous vote of confidence in peer-accelerated delivery. With public broadcasters like France TV and Spain's RTVE opting for our solution and with Lumen offering a mesh delivery platform powered by Streamroot's technology, peer-to-peer is poised to be the next video delivery standard. With a history that dates back to the origins of the web, distributed architectures – the cornerstone of the early internet – are finally regaining their rightful place at the core of online services.