This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE P2P 2012 proceedings

Peer2View a Peer-To-Peer HTTP-Live Streaming platform

Roberto Roverso[1,2], Sameh El-Ansary[1], Seif Haridi[2]
[1] Peerialism Inc., Sweden, [2] KTH-Royal Institute of Technology, Sweden,
{roberto, sameh}@peerialism.com, haridi@kth.se

***Abstract*—Peer2View is a commercial peer-to-peer live video streaming (P2PLS) system. The novelty of Peer2View is threefold: i) It is the first P2PLS platform to support HTTP as transport protocol for live content, ii) The system supports both single and multi-bitrate streaming modes of operation, and iii) It makes use of an application-layer dynamic congestion control to manage priorities of transfers. Peer2View goals are to achieve substantial savings towards the source of the stream while providing the same quality of user experience of a CDN.**

I. Introduction

In the last years, significant efforts have been made in tackling the problem of distributing live content over peer-to-peer networks. Both industry and academia have been trying to come up with solutions to efficiently exploit spare upload bandwidth of participating hosts to offload the broadcasting origin and thus save on distribution costs. We find successful examples in commercial systems like NewCoolstreaming [1] and PPlive [2]. On the academic front instead, there have been several attempts to build open source frameworks for streaming, such as Tribler [3], but also to understand theoretical limits of different overlay structures in terms of bandwidth utilization.

Most of the aforementioned research and developed systems assume a push mode of operation typical of older streaming technologies, such as RTSP/RTP. In this model, a player requests the playback of a stream from the streaming server, the server then delivers the stream over UDP at a pace determined by feedback information from the player, such as acknowledgements or error notifications.

Lately however, the industry has moved on to other live streaming technologies based on a pull model that utilizes HTTP as transport protocol. All companies who have a major say in the market including Microsoft, Adobe and Apple have adopted adaptive HTTP-streaming as the main approach for live broadcasting. This shift to HTTP has been driven by a number of advantages: i) Routers and firewalls are more permissive to HTTP traffic compared to the RTSP/RTP ii) HTTP caching for real-time generated media is straightforward like any traditional web-content iii) The Content Distribution Networks (CDNs) business is much cheaper when dealing with HTTP downloads.

In adaptive HTTP live streaming, the mode of operation is significantly different from RTSP/RTP in the way that it is the player which periodically pulls parts of the content from the streaming server. The latter instead simply encodes the content and avails it as small HTTP files. The same stream is usually encoded in a number of different qualities (bitrates) both for audio and video content. It is then up to the player to decide which quality to request and the pace at which fragments should be retrieved.

Players are usually proprietary and closed source. As a consequence, it is extremely difficult to make assumptions about their internal heuristics and, in particular, about the period between consecutive fragment requests, the time at which the player will switch rates, or how the audio and video will be interleaved.

In Peer2View, in order to cope with the unpredictability of different player implementations and versions of those, we treat the problem of reducing the load on the source of the stream the same way it would be treated by a Content Distribution Network (CDN): as a *caching problem*.

II. Overview

In this section, we provide a general description of Peer2View and we highlight some of the most interesting features of the platform. Our previous work on SmoothCache [4], Peer2View's research prototype, discusses more in detail the functioning of the platform.

Distributed caching. The design of the HTTP live streaming protocol was made such that every fragment is fetched as an independent HTTP request that could be easily scheduled on CDN nodes. The difference in Peer2View is that the caching nodes are consumer machines and not dedicated nodes. The player is directed to order from our local P2PLS agent which acts as an HTTP proxy. All traffic to/from the source of the stream as well as other peers passes by the agent. Upon a content fragment request, Peer2View does its best to timely retrieve the data requested by the player from other peers in the overlay. If a fragment cannot be retrieved from any other peer on time, the agent downloads the missing amount of data from the CDN. We engineered this process in such a way to make the agent totally transparent to the player and the CDN. In this manner, our platform is able supports all HTTP-based live steaming players.

Overlay Construction and Maintenance. Peer2View is implemented as a completely self-organizing system based on a mesh overlay network. When joining the system, peers are introduced to other participants by a tracker. After that, they utilize gossip to build a random overlay. Gossip is also used for dissemination of live peer characteristics such as maximum/average throughput, connectivity information, playback quality, buffering point and network location.

978-1-4673-2862-3/12/$31.00 ©2012 IEEE

Fig. 1. Live Peer2View Statistics

Peers choose partners for the delivery of the stream by sampling their local view and by ranking neighbors according to the aforementioned characteristics. By means of a distributed heuristic, peers are placed proportionally closer to the source of the stream depending on their upload capacity to improve efficiency of the distribution.

Transport. Peer2View is built on top of a UDP-based transport library which provides security as well as reliability. Peers are authenticated towards a central authority upon joining the system. All traffic is encrypted through SSL and end-to-end integrity of data is guaranteed by signing all data transferred data chunks. The library implements the same flow control and error correction of TCP. Regarding the congestion control, we use the DTL dynamic congestion control protocol [5], which allows changing priority of transfers at runtime. Priority levels range from lower-than-best-effort, designed to yield to TCP, up to four times more aggressive than TCP.

The library utilizes the state-of-the art NATCracker [6] traversal scheme to provide connectivity between peers in presence of network address translators. In our deployments, we have observed an average bi-directional connection probability of around 81%, whereas one-way connectivity could be achieved in 8% of cases with a remaining failed establishment rate of 11%.

Proactive caching. Peer2View strives to increase the overlay utilization by retrieving fragments ahead of time, before the player actually requests them.

In order to avoid contention between transfers of fragments closer to the playback deadline and the ones being pre-fetched, Peer2View pre-fetches using the the lowest transfer priority level of DTL. As the fragment being pre-fetched comes closer to playback, the priority is increased to speed up the transfer.

III. Performance

We measure performance according to two metrics: savings towards the source of the stream and quality of user experience. Savings are evaluated by accounting the amount of data retrieved from P2P network over the total amount consumed by the peers.

Quality of user experience (QoE) is measured first using cumulative buffering, i.e. how long the player spent waiting for content. This time includes both the initial startup delay and the time the player paused the playback because of lack of content. Second indicator of QoE is the bitrate a peer was able to play when using Peer2View compared to the one it would play going directly to the CDN.

Our evaluation of Peer2View conducted on the open Internet using both automated tests on a 5000-large test network contributed by volunteers and in collaboration with our commercial partners suggests that savings range normally between 85% to 90% using a multi-bitrate stream of 3 qualities: $380kbps$, $700kbps$ and $1.4Mbps$.

In all tests, we noticed nearly the same level of QoE of the CDN, with only 0.5% to 3% of the peers experiencing a larger cumulative buffering time than the CDN and 1% to 2.5% of the peers playing a lower bitrate when using both CDN and P2P compared to the case when playing directly from the CDN.

IV. Demonstration

During the demonstration we expect to conduct live runs on our test network. Our graphical tools allows us to observe the live evolution of a number of metrics, such as the ones shown in Figure 1, and the shape of the overlay network over time.

Interested participants will be welcome to join the demo by installing our client and play the stream with Peer2View.

References

[1] Li, B. et al., *Inside the New Coolstreaming: Principles, Measurements and Performance Implications*, INFOCOM, 2008

[2] A. Shahzad , A. Mathur and H. Zhang, *Measurement of Commercial Peer-to-Peer Live Video Streaming*, Workshop on Recent Advances in P2P Streaming, 2006.

[3] N. Zeilemaker, M. Capota, A. Bakker, J. Pouwelse, *Tribler: Search and Stream*, IEEE P2P 2011, Kyoto, Japan

[4] R. Roverso, S. El-Ansary and S. Haridi, *SmoothCache: HTTP-Live Streaming Goes Peer-to-Peer*, NETWORKING 2012, May 2012, Prague.

[5] R. Reale, R. Roverso, S. El-Ansary and S. Haridi, *DTL: Dynamic Transport Library for Peer-To-Peer Applications*, IDCDN 2012, January 2012, Hong Kong, China.

[6] R. Roverso , S. El-Ansary and S. Haridi, *NATCracker: NAT Combinations Matter*, ICCCN '09, August 2009, San Francisco (CA), United States.