On HTTP Live Streaming in Large Enterprises

Roberto Roverso
Peerialism AB
KTH - Royal Institute of Technology
Stockholm, Sweden
roberto@peerialism.com

Sameh El-Ansary
Peerialism AB
Stockholm, Sweden
sameh@peerialism.com

Mikael Högqvist
Peerialism AB
Stockholm, Sweden
mikael@peerialism.com

ABSTRACT

In this work, we present a distributed caching solution which addresses the problem of efficient delivery of HTTP live streams in large private networks. With our system, we have conducted tests on a number of pilot deployments. The largest of them, with 3000 concurrent viewers, consistently showed that our system saves more than 90% of traffic towards the source of the stream while providing the same quality of user experience of a CDN. Another result is that our solution was able to reduce the load on the bottlenecks in the network by an average of 91.6%.

Categories and Subject Descriptors

C.2.2 [**Computer-Communication Networks**]: Network Protocols; C.2.4 [**Computer-Communication Networks**]: Distributed Systems

General Terms

Algorithms, Performance

Keywords

HTTP live, private networks, distributed caching, content delivery network

1. INTRODUCTION

In the last years, the streaming industry has witnessed a shift from the RTP/RTSP standard towards HTTP live, a set of protocols which all utilize HTTP for delivery of live and on-demand content in IP networks [1]. Microsoft, Adobe and Apple have developed players which embrace HTTP-streaming and the concept of adaptive bitrate switching as the main approach for broadcasting. HTTP live has been adopted by content services and creators like Netflix, Hulu and the BBC with support across all platforms and OSs, including computers, tablets and smart phones.

The main challange of delivering HTTP-based live streams is the unicast nature of the HTTP protocol. This creates a potential bottleneck at the source of the stream with a linear increase in bandwidth demand as the number of viewers increases. A natural approach, which is also the primary solution to handle capacity issues for normal HTTP traffic, is to introduce caching. For HTTP live this is the only alternative since there is no multicast support.

While caching of HTTP live via Content Distribution Networks (CDNs) is common for the open Internet, it is challenging to deploy efficiently within private networks such as those operated by corporations or other entities. Larger private networks interconnect multiple network segments representing geographically distributed offices with fixed VPN links. Inside an office, the network is constructed with high capacity links and switches. Traffic from and to the public Internet is routed through a gateway link of fixed capacity or through one or more segments until a gateway link is reached.

The main bottlenecks in private networks are the VPN and gateway links which are typically not dimensioned to sustain the load of delivering one or more streams to a large amount of viewers at the same time. While a private CDN would dramatically improve the efficiency of HTTP live performance, it is hard to deploy and manage since it requires *i*) new or upgraded caching hardware to support HTTP live *ii*) that each network segment is covered to handle the bandwidth load of all potential viewers, *iii*) handling of heterogeneous network infrastructure resulting from network changes such as company acquisitions and mergers.

In this work, we are exploring a software-based CDN for HTTP live in private networks. We leverage the experience acquired in our previous research on improving the efficiency of delivery of HTTP streams over the Internet, where we proposed a peer-to-peer distributed caching approach [4]. Based on the same principles, we design an overlay which minimizes inter-segment traffic. In doing so, we enable efficient HTTP live streaming without the need of deploying and managing expensive and specialized hardware. In addition, we evaluate our solution in real-world deployments in private networks which allows us to present unique insights into the problem at hand.

2. CHALLENGES

In this section, we identify a set of challenges and requirements which a distributed caching system must satisfy to be a viable solution for HTTP live streaming in a private network.

Locality and structure awareness. The overlay must be constructed to follow the physical structure of the private

SIGCOMM'13, August 12–16, 2013, Hong Kong, China.
ACM 978-1-4503-2056-6/13/08.

network in order to keep traffic within local segments and offload gateway and VPN links.

Bitrate switching. An HTTP live player dynamically switches between different bitrates for the same stream depending on the available bandwidth and host rendering capabilities. Our solution must therefore be able to quickly handle bitrate changes.

Vendor neutrality. Support different players and protocols such as Microsoft's Smooth Streaming, Adobe's HTTP Dynamic Streaming and the upcoming standard MPEG-DASH.

Politeness. The delivery and relaying of content should not interfere with other activities of the user or network traffic generated by critical services.

Finally, in our industrial experience, we have found that **quality of experience (QoE)** is a feature that content owners are not willing to compromise on. Our service should then strive to improve efficiency of distribution while providing QoE which matches the one of private CDNs.

3. SYSTEM OVERVIEW

In HTTP live streaming protocol every fragment is fetched as an independent HTTP request that could be scheduled on caching servers. The difference in our solution is that the caching is done at desktop machines instead of dedicated servers. The HTTP player is directed to a local caching 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, the caching agent tries 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 portion of the fragment from the source of the stream, e.g. a public CDN. By falling back to the source, we guarantee that all fragments are delivered on time even if the overlay network cannot retrieve such fragments, thereby guaranteeing the desired level of QoE. This process is engineered to make the agent totally transparent to the player and the streaming infrastructure. In this manner, our platform can support all HTTP-based live streaming protocols.

Overlay Construction. Our distributed caching system is implemented as a 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 build a random overlay which is used for dissemination of live peer information, e.g. throughput and playback quality. A network segment id is provided by a central registry, with a mapping provided by the network's owner. Peers choose their neighbours by sampling their local view and by ranking peers according to the aforementioned information. Peers make sure to partner with nodes which are retrieving different bitrates, in order to adapt quickly to player bitrate switches.

One or more peers in a segment are promoted to act as live caches for all others in the same segment. The promotion process is implemented either with the help of a locality-aware central service or by means of a distributed K-leader election algorithm similar to [2]. In order to determine the peers to be promoted, we utilize an absolute ranking based on metrics such as computational load, bandwidth and connectivity.

Delivery. Promoted peers are tasked with pre-fetching content ahead of all other peers in the same segment. The pre-fetching happens either from the source of the stream or from other nodes outside their segment. We manipulate the pre-fetching in a way that promoted peers retrieve the stream from the CDN only if their segment has a gateway link, as the content is typically provided by a source external to the private network.

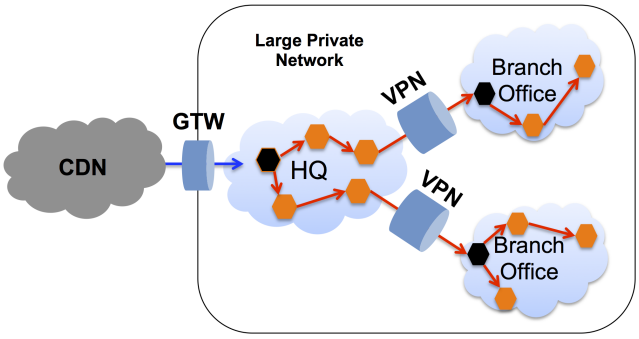

Figure 1: HTTP live delivery in a private network with our solution

As soon as a fragment is pre-fetched, other nodes in the segment start to retrieve it from the promoted peer using lower-than-best effort priority [3], as not to interfere with other critical traffic in the network.

A sample delivery overlay is shown in Figure 1, promoted peers are highlighted in black, while the others in orange. Arrows denote the traffic's flow across the gateway (GTW) and VPN links, as well as across the network segments.

4. PRELIMINARY RESULTS AND DEMO

We have conducted a number of pilot deployments of our system in large corporations located in the US and in Sweden. The largest deployment comprised of 48.000 peers on 89 network segments. Tests conducted with 3000 concurrent viewers consistently showed that our system saves more than 90% of traffic towards the source of the stream. Another result is that our system was able to reduce the traffic on the bottlenecks in the network by an average of 91.6%.

In the demo, we will provide extensive and interactive visualisation of the aforementioned data and we will showcase a live stream distributed with our system to a number of on-site desktop machines and mobile devices. More information about our system can be found at [5].

5. REFERENCES

[1] S. Akhshabi, A. C. Begen, and C. Dovrolis. An experimental evaluation of rate-adaptation algorithms in adaptive streaming over HTTP. In *Proc. of ACM MMSys*, 2011.

[2] V. Raychoudhury, J. Cao, and W. Wu. Top k-leader election in wireless ad hoc networks. In *Proc. of IEEE ICCCN*, 2008.

[3] R. Reale, R. Roverso, S. El-Ansary, and S. Haridi. DTL: Dynamic Transport Library for Peer-To-Peer Applications. In *Proc. of ICDCN*, 2012.

[4] R. Roverso, S. El-Ansary, and S. Haridi. Smoothcache: Http-live streaming goes peer-to-peer. In *Proc. of IFIP NETWORKING*, 2012.

[5] Peerialism AB. http://www.peerialism.com.