Design and Deployment of a Hybrid CDN-P2P System for Live Video Streaming: Experiences with LiveSky

Hao Yin[1], Xuening Liu[1], Tongyu Zhan[1], Vyas Sekar[2], Feng Qiu[3], Chuang Lin[1], Hui Zhang[2], Bo Li[4]

[1] Tsinghua University, China [2] Carnegie Mellon University [3] ChinaCache Co., Ltd, China

[4] Hong Kong University of Science and Technology, China

ABSTRACT

We present our design and deployment experiences with *LiveSky*, a commercially deployed hybrid CDN-P2P live streaming system. CDNs and P2P systems are the common techniques used for live streaming, each having its own set of advantages and disadvantages. LiveSky inherits the best of both worlds: the quality control and reliability of a CDN and the inherent scalability of a P2P system. We address several key challenges in the system design and implementation including (a) dynamic resource scaling while guaranteeing stream quality, (b) providing low startup latency, (c) ease of integration with existing CDN infrastructure, and (d) ensuring network-friendliness and upload fairness in the P2P operation. LiveSky has been commercially deployed and used for several large-scale live streaming events serving more than ten million users in China. We evaluate the performance of LiveSky using data from these real-world deployments. Our results indicate that such a hybrid CDN-P2P system provides quality and user performance comparable to a CDN and effectively scales the system capacity when the user volume exceeds the CDN capacity.

Categories and Subject Descriptors

C.2.4 [**Distributed Systems**]: Distributed applications

General Terms

Design, Measurement, Performance

Keywords

Content Delivery Networks, Peer-to-Peer, Live Streaming

1. INTRODUCTION

Live video streaming has long been projected as the "killer-app" for the Internet. While this expectation has been in effect for several years now, only in recent years with the deployment of increased bandwidth in the last-mile has this promise finally turned into reality (e.g., [6, 12, 13]).

MM'09, October 19–24, 2009, Beijing, China.

There are two alternative (and competing) technologies for delivering live video streaming to end-users: Content Delivery Networks (CDN) and Peer-to-Peer (P2P) systems. CDNs such as Akamai [1] and Limelight [5] deploy servers in multiple geographically diverse locations, distributed over multiple ISPs. User requests are redirected to the best available server based on proximity, server load etc. CDNs provide end-users with the appearance of traditional client-server approaches, but enable content providers to handle much larger request volumes. Thus, end-users observe higher quality experience and content providers can offer more reliable service. At the same time, ISPs can also benefit from deploying CDN servers in their networks as it reduces the total amount of upstream and transit traffic.

P2P systems solve the scalability issue by leveraging the resources of the participating peers. P2P video streaming systems have attracted a large number of Internet viewers, as demonstrated by the huge popularity of applications such as PPLive [9], Joost [4], CoolStreaming [42]. For example, PPLive uses < 10 Mbps of server bandwidth to serve a 400 kbps video stream to roughly 1.5 million users [23].

Both approaches have their advantages and disadvantages. CDNs provide excellent quality to end-users when the workload is within the provisioning limits. CDNs typically have to provision servers and bandwidth in advance using estimates of the expected workload and are thus inherently constrained by the specifics of their operating regime. Anecdotal evidence, for example, during the recent US presidential inauguration [7], suggests that even popular sites and providers can be overwhelmed by unexpected surges in demand and thus have to deny service to end-users. This scaling constraint becomes especially relevant as users and content providers demand higher quality video, i.e., implying higher operating costs for the CDN.

P2P systems achieve high scalability while keeping the server requirements low. However, the decentralized, uncoordinated operation implies that this scaling comes with undesirable side effects. Typical problems: (1) low stream quality with undesirable disruptions [38], (2) unfairness in the face of heterogeneous peer resources (e.g., high-bandwidth peers become loaded, peers behind NATs do not contribute [14, 27]), and (3) network unfriendliness [14].

A natural question is if we can build a hybrid CDN-P2P architecture that incorporates the best of both technologies and mutually offsets each others' deficiencies. Several researchers have hypothesized and analyzed the potential benefits of such an approach [40, 22] via simulations and trace-driven analysis. However, we are not aware of any real

implementation and deployment that actually demonstrates the benefits of such a hybrid approach.

In this paper, we present the design, implementation, and real-world deployment and evaluation of *LiveSky*, a hybrid CDN-P2P live streaming system developed and deployed by ChinaCache [2]. In designing and deploying LiveSky, we have addressed several key challenges, including:

- A mechanism for dynamic resource scaling that guarantees adequate quality-of-service to end-users.
- Providing good end-user experience such as low startup latency, low stream disruption rate, etc.
- Ease of integration of the hybrid P2P-CDN system with the existing CDN infrastructure.
- Addressing the well-known shortcomings of P2P streaming including high buffering requirement, low stream quality, unfairness in upload contributions, and network-unfriendliness.

An independent and equally valuable contribution of this paper is our analysis of the performance of LiveSky "in the wild". To the best of our knowledge, this is the first study of a widely deployed live streaming system using hybrid CDN-P2P techniques. Several commercial providers have been recently adopting hybrid P2P solutions to scale content distribution (e.g., RedSwoosh [10], Octoshape [8]). However, real-world data on such deployments are hard to obtain. In this light, our evaluation becomes especially valuable for the future development of live streaming technologies and applications. Our measurement results indicate that:

- LiveSky works well even when the client upload bandwidth is restricted and effectively leverages the resources of both CDN and P2P nodes.
- Most clients obtain good quality service and suffer few viewing disruptions, even with a small 15 second media buffer.
- Users see more than 2× shorter startup latency compared to pure P2P approaches.
- LiveSky provides good performance even when the churn –the fraction of peers that join or leave in a specific interval (e.g., 1 minute)–is as high as 10%, effectively insulating most clients from churn-induced disruptions.
- LiveSky effectively offsets the deficiencies of P2P systems. Specifically, LiveSky leverages CDN redirections to make P2P transfers "network-friendly". Also, the presence of a publicly addressable and available CDN node enables NAT traversal, thereby better utilizing the available upload capacity of NAT-ed hosts.

To summarize, the key contributions of this paper are,

- Design and implementation of LiveSky – a hybrid CDN-P2P system for effectively scaling the capacity of a CDN without compromising the reliability and user experience (Section 2).
- Addressing key challenges in integrating the P2P component into the CDN, overcoming the shortcomings of traditional P2P systems for live streaming, and designing an adaptive scaling mechanism to guide the hybrid CDN-P2P operation (Section 3).
- Measurements and analysis of real-world deployments that validate the benefits of a hybrid CDN-P2P architecture (Section 4).

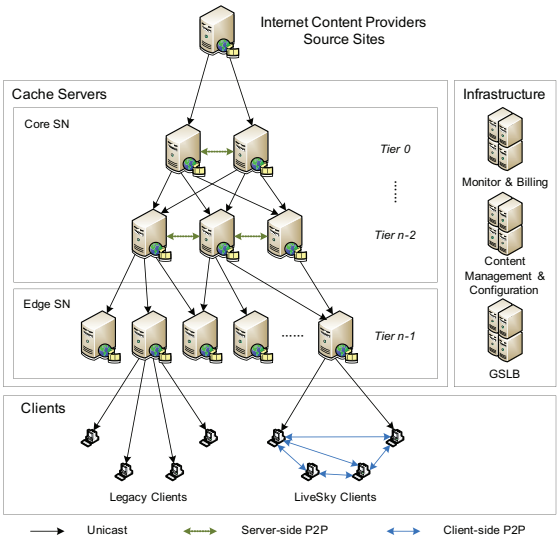


Figure 1: System Architecture

2. SYSTEM OVERVIEW

In this section, we describe the general architecture of the *LiveSky* system. As shown in Figure 1, there are three major components: (1) Management Center (MC) comprising the DNS-based Global Server Load Balance (GSLB) system, content management and configuration system, and monitoring and billing systems; (2) cache servers, referred to as Service Nodes (SN) that deliver video contents from content providers to end users; (c) end hosts which may either be legacy clients, which directly obtain the stream from the edge servers or LiveSky-enabled clients, which can additionally engage in P2P transfers.

System Management: The Management Center (MC) is responsible for efficient control and monitoring of the LiveSky system. The DNS-based GSLB system in the MC redirects user requests to the nearest, lightly loaded server [3]. The MC distributes configurations to the SNs using XML messages; these messages use incremental updates to reduce the communication overhead. The configuration parameters include channel information, source information, operating strategies etc.

2.1 CDN Overlay

The SNs are organized into several tiers, with Tier_0 being closest to the content source and Tier_{n-1} closest to the end users as shown in Figure 1. We refer to the SNs in Tier_{n-1} as *edge* SNs since they are directly responsible for serving end users. The SNs in the remaining tiers are *core* SNs since their primary responsibility is to act as a distribution overlay to deliver the content to the edge SNs. This hierarchical arrangement is typical of many CDN infrastructures to effectively magnify the total system capacity, reduce the load at the content source, and also leverage the benefits of caching requested contents in higher layers.

Each SN is allocated a unique ID. When SN_i boots up, it sends a "alive" message to the MC. The MC then broadcasts the alive message to other SNs. A different SN_j can obtain the attributes of SN_i (e.g., IP address and TCP port information) from the MC to establish a TCP connection with SN_i if necessary.

The server-side distribution mechanism is largely tree-based. However, in order to provide greater reliability in the presence of node or network failures, we allow each SN to retrieve the content either from SNs higher up in the hierarchy (i.e., a lower numbered tier) or from peer SNs in the same tier. Since the edge SNs are responsible for serving end users, they are typically heavily loaded and we disable peering between SNs in the edge tier.

Edge SNs handle client requests and obtain the required contents from the core SNs. Requests from edge SNs are forwarded up the hierarchy until they find a node that has the desired content. To minimize the load at the content source, only Tier_0 SNs retrieve content directly from it.

The goal of the server-side overlay is efficient data distribution with some measures to guard against some node failures and network delays. As the CDN nodes have high availability and are stable, a tree-based overlay with additional peer edges satisfies the goals of providing reliable, yet efficient data transmission.

2.2 System Operation

A client first obtains the URL for the live stream from the content source (e.g., `livesky://domainname/live1`). The GSLB component of the CDN takes into account the client location, the edge SN location, and the edge SN loads to find a suitable edge SN for this client. The client is then redirected to this edge SN using traditional DNS-based redirection techniques [3].

Each edge SN serves multiple roles. First, it acts as a regular server for legacy clients. Second, it serves as a *tracker* for the P2P operation to bootstrap new clients with candidate peers. Third, it acts as a *seed* for the P2P operation for the LiveSky-enabled clients assigned to it. The edge SNs are pre-configured with some decision logic that decides if a new LiveSky-enabled client should be served in CDN-mode or if they should be redirected to the P2P overlay. Finally, the edge SN is used for some optimizations in the P2P operation. Note that the P2P overlays are localized on a per-edge SN basis; i.e., the peers with which a LiveSky enabled node communicates in the P2P mechanism are also assigned to the same edge SN as this node. We discuss these last two roles in more detail in the next section.

2.3 Client Side Distribution

Legacy Clients: As discussed earlier, there are two types of clients: legacy clients which receive contents directly from the edge SNs and LiveSky enabled clients which can either receive contents from the edge SNs or additionally use P2P mechanisms. An important distinction between the legacy and LiveSky clients is that the LiveSky clients can access a higher quality video stream whereas the legacy clients may only be able to access a lower quality stream. This incentivizes users to install the LiveSky client software and encourages widespread adoption. In our experience, we find that typically more than 50% of users have adopted LiveSky.

LiveSky's P2P Mechanism: Recent proposals [41, 39] demonstrate that a hybrid approach combining the multi-tree [36, 16] and mesh [42, 9] schemes achieves both efficient delivery and robustness to churn. We adopt a similar scheme in LiveSky. The video stream is a single bit-rate encoding (i.e., we do not use any layered coding) and is separated into several substreams according to the stream frame id. For example, if the video is divided into six substreams, substream_0 consists of frames $0, 6, 12, 18, \ldots$, substream_1 consists of frames $1, 7, 13, 19, \ldots$ and so on. Peers are organized in a tree-based overlay on a per-substream basis. This ensures that all nodes contribute some upload bandwidth. Additionally, in order to be robust to network or node failures, peers also use a mesh-style pull mechanism to retrieve missing frames for continuous playback.

3. ADAPTATION AND OPTIMIZATIONS

In this section, we present a more detailed discussion of two key aspects of the LiveSky system: (1) system adaptation in the presence of unexpected load to achieve an efficient tradeoff between user quality, CDN cost, and system scalability, and (2) optimizations to address some of the shortcomings of P2P for effective live video delivery.

3.1 Adaptive Scaling

There are natural tradeoffs in a hybrid CDN-P2P environment between the operating cost of the CDN, the perceived user quality (e.g., media rate, delay between the video source and the end user), and the overall number of end-users that can be supported by the system at a given time.

The key challenge is to adapt the hybrid CDN-P2P operation depending on the working environment so that the users observe good performance and at the same time the operating costs of the CDN are not too high. To this end, we present an analytical model to understand the tradeoffs involved. The objective of the analysis is to guide the operation of the LiveSky system, especially to control the number of end-users served directly by the CDN.

The working environment for a hybrid CDN-P2P system is defined by several important parameters such as (1) the media playback rate, (2) the total number of end-users, (3) the bandwidth capacity of the edge SNs, (4) bounds on user quality defined as the delay between the video source and the end user, and (5) characteristics of end-users such as their upload bandwidth capacity and churn rates. Given the working environment, the control parameter is the number of end-users that are served directly by the edge SNs. Our analysis models the relationships between these parameters in order to derive guidelines for system operation.

Assumptions: In order to make the analysis tractable, we make four simplifying assumptions.

- First, we model the P2P overlay as a single tree, to avoid the complexity of modeling the hybrid mesh-multitree mechanism. Since our actual overlay is more efficient than a tree, this is a *conservative* assumption that underestimates the system performance. Consequently, we model the delay between the source and a user in terms of the *level* of that user in the tree.
- Second, we imagine that the stream is divided into multiple equal-length segments, and consider this segment as a basic unit of operation. All clients at the same level in the P2P tree receive a specific segment at the same time.
- Third, we assume that the total upload bandwidth of clients in level k of the P2P tree is always larger than the download bandwidth requirement of clients in level $k+1$. This means that clients in this analytical model never need to *rebuffer*. In other words, they do not experience buffer underflows, where the media buffer does not have any content for the higher-layer application to render.

Notation	Definition
ρ	The average fraction of full streaming rate that each supplying peer contributes during a session
k	Level in the P2P overlay; peers in lower numbered levels receive the content earlier and serve it to peers in the succeeding level. Especially, peers in level 1 directly receive the content from the CDN server
K_0	Maximum number of levels
N_c	CDN server capacity
P_0	Total number of clients for this media
N(k)	Upload capacity of clients in level k , $N(0) = N_c$
P(k)	Number of clients in levels $(k+1)\dots K_0$, P(0)=P_0
σ	Leave rate expressed as a fraction of peers that leave in a specific interval
λ	Join rate expressed as a fraction of peers that join in a specific interval

Table 1: Notation used in our analytical model

- Finally, we only consider aggregate measures (i.e., population and time averages) to model the end-user properties such as upload capacities and churn rates.

Since our goal is to provide an approximate guideline for operation, we believe that these simplifying assumptions are reasonable. In the following discussion, we adopt and extend the analysis framework of Xu et al. [40].[1]

In LiveSky, each edge SN independently decides whether to serve a new client directly or redirect it to a peer which has already joined the system. We make this explicit design choice to have each edge SN operate independently to minimize the changes to the existing CDN infrastructure and ensure ease of integration of the P2P component into the CDN. In addition, this also implicitly ensures some form of "fate isolation" across edge SNs in localizing the effects of network or node failures and thus improving the overall robustness of the system (e.g., avoiding global oscillations or cascading effects). Thus, for the following analysis, we focus on the properties of a single edge SN.

Basic Analysis with No Churn: Suppose the edge SN has sufficient capacity to serve only N_c concurrent streaming sessions. Let P_0 be the total number of clients assigned to this SN that wish to receive this live video stream.

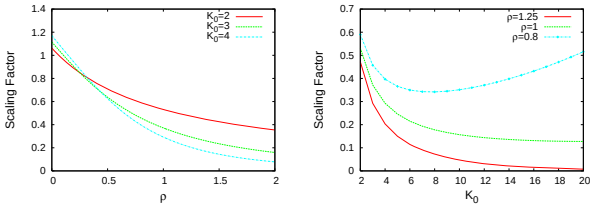
Let ρ denote the average fraction of full streaming rate that each supplying peer contributes during a session. Let $N(k)$ represent the total P2P streaming capacity of peers in level k. Note that as a specific case, $N(0) = N_c$. Since we have a total capacity of $N(k-1)$ at level $k-1$ and peers can support ρ sessions on average, we have

$$N(k) = \rho N(k-1) \tag{1}$$

Let $P(k)$ be the number of clients in levels $(k+1)\dots K_0$. Then,

$$P(k) = P(k-1) - N(k-1) \tag{2}$$

[1]There are a few differences between the analysis of Xu et al. [40] and that presented here. First, we explicitly model the upload bandwidth restrictions on peers and the download streaming requirement at each level. Second, our model of node joins/leaves is more suited to our operational environment. In their framework, peers leave the P2P system once they have contributed some threshold capacity; in our case, peers leave at random.



(a) Vs. ρ (b) Vs. K_0

Figure 2: Relationship between the scaling factor S (fraction of nodes served directly by the CDN node) and ρ (average client upload capacity relative to media rate) and K_0 (number of levels in the P2P tree)

Combining the two above equations we have,

$$P(k) = \begin{cases} P_0 - \frac{N_c(1-\rho^k)}{1-\rho} & \text{if } \rho \neq 1 \\ P_0 - k \times N_c & \text{if } \rho = 1 \end{cases}$$

Let K_0 denote the maximum level in the P2P overlay; i.e., a parameter to capture the maximum acceptable source to user delay. Then, by definition $P(K_0) = 0$; i.e., it is not acceptable to have clients more than K_0 levels away from the CDN node. This means that,

$$P(K_0) = 0 = \begin{cases} P_0 - \frac{N_c(1-\rho^{K_0})}{1-\rho} & \text{if } \rho \neq 1 \\ P_0 - K_0 \times N_c & \text{if } \rho = 1 \end{cases} \tag{3}$$

We define the *scaling factor*, $S = \frac{N_c}{P_0}$, the fraction of the clients directly served by the CDN. Then, from (3) we have,

$$S = \frac{N_c}{P_0} = \begin{cases} \frac{1-\rho}{1-\rho^{K_0}} & \text{if } \rho \neq 1 \\ \frac{1}{K_0} & \text{if } \rho = 1 \end{cases} \tag{4}$$

Analysis with Node Churn: We model the node churn using two parameters: the average leave rate σ and the average join rate λ. σ and λ are expressed as a fraction of the total number of current peers. We can modify (1) and (2) to include node churn as:

$$N(k) = N(k-1) \times (1-\sigma) \times \rho \tag{5}$$

$$P(k) = P(k-1) - N(k-1) - P(k-1) \times \sigma + P_0(1+\lambda-\sigma)^{k-1}\lambda \tag{6}$$

Proceeding along similar lines, we can show that

$$S = \frac{N_c}{P_0} = \begin{cases} \frac{(1-\rho)(1+\lambda-\sigma)^{K_0}}{(1-\rho^{K_0})(1-\sigma)^{K_0-1}} & \text{if } \rho \neq 1 \\ \frac{(1+\lambda-\sigma)^{K_0}}{K_0(1-\sigma)^{K_0-1}} & \text{if } \rho = 1 \end{cases} \tag{7}$$

Examples for illustration: Figures 2(a) and 2(b) show how the scaling factor S varies as a function of ρ and K_0 respectively based on the analysis above. We use a constant peer arrival rate $\lambda = 3.6\%$ and a constant peer leave rate $\sigma = 2.4\%$. As expected, when the available P2P bandwidth ρ increases, the system scaling improves and a smaller fraction of the users need to be served by the CDN in Figure 2(a). When $\rho > 1$, by relaxing the delay constraint, i.e., increasing K_0, we can reduce the CDN contribution. When $\rho = 1$, the system cannot be scaled arbitrarily, and the scale levels off as a function of K_0. For $\rho < 1$, as the delay bound K_0 is increased, the CDN contribution initially decreases, but subsequently increases beyond a certain critical value.

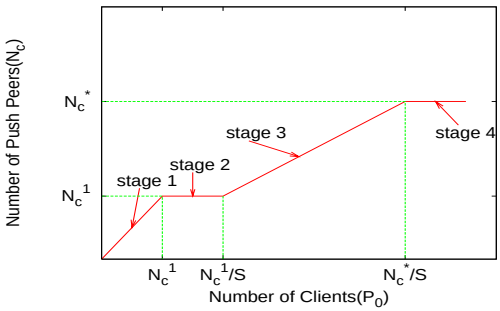


Figure 3: Different stages in the operation of an edge SN in LiveSky as a function of the number of clients

Using the analysis to guide LiveSky operation: We use the above analysis in conjunction with experiences gained from trial deployments to derive configuration parameters to aid the decision process of each SN. The key is to control N_c, the number of nodes directly served by the edge node, depending on the current load P_0. Each edge SN is *pre-configured* with the decision logic to determine N_c depending on P_0. We briefly describe the decision logic below.

Suppose, we have estimates of ρ, λ, and σ. These can be obtained from understanding user behaviors and available bandwidth characteristics from earlier test deployments and other studies. We also choose a suitable value of K_0, the maximum number of levels in the client-side P2P based on an upper bound on the acceptable source-to-user latency and also on observed performance in field trials. In our deployment, we use $K_0 = 2$.

Each edge SN can be in four stages (Figure 3):

- Stage 1, $P_0 \leq N_c^1$: All clients retrieve the content directly from edge server.
- Stage 2, $N_c^1 \leq P_0 \leq \frac{N_c^1}{S}$: New clients are served by one of the first N_c^1 peers, i.e., we switch from the $K_0 = 1$ mode to the $K_0 = 2$ mode.
- Stage 3, $\frac{N_c^1}{S} \leq P_0 \leq \frac{N_c^*}{S}$: Half the new clients are served directly, rest are redirected to peers.
- Stage 4, $P_0 \geq \frac{N_c^*}{S}$: The edge SN hits its capacity limit N_c^*. New clients have to be redirected to existing peers. Since $K_0 = 2$, we have $S \approx 0.5$ this means that when $P_0 \geq 2 \times N_c^*$, new clients are redirected to other less loaded edge SNs.

The specific thresholds (N_c^1 and $\frac{N_c^1}{S}$) when the edge SN transitions between stages are based on the bounds suggested by (7). N_c^* is simply the total bandwidth capacity of the edge SN divided by the media rate. Since the bound does not account for extra load due to rebuffering events (discussed next in stability under dynamics), we make sure that the actual values of N_c^1, $\frac{N_c^1}{S}$ are slightly higher than these analytical bounds. While our current deployment uses $K_0 = 2$, we can extend our system to use larger K_0 values as workloads become even larger and client-side P2P technologies evolve over time.

3.2 Improvements in the P2P layer

Locality: Each edge SN keeps track of clients currently assigned to it (and none others). Each client learns about other peers assigned to its designated edge SN. Since the CDN redirection already takes into account the client and SN locations when assigning a client to an edge SN, this automatically ensures that clients mostly peer with other clients in the same region. This can be viewed as a more direct implementation of a recent proposal for localizing P2P transfers [17].[2]

Upload fairness and NAT handling: A host behind a NAT can receive data but often cannot serve other hosts. Thus, NAT-ed hosts may not contribute any capacity to the system. For example, measurements from CoolStreaming [27] show that users behind NATs contribute less than one-sixth of the stream rate. This is a serious concern–in many real deployments, a significant fraction of the hosts are behind NATs. Measurements from PPLive [24] show that 60%-80% of hosts in China are behind some kind of NAT; our measurements (Section 4.6) reveal that more than 90% of the peers in LiveSky are behind NATs.[3] Thus, it is necessary to incorporate NAT traversal mechanisms for the system to scale to a large user population.

We adopt well-known techniques such as STUN [34] to deal with such connectivity restrictions. In doing so, we leverage the edge SN to serve as an intermediary to facilitate NAT traversal. STUN uses UDP to achieve efficient transmission between clients that behind NATs. We add application-layer ACKs to ensure reliable data delivery over UDP. We also adopt upload bandwidth restriction mechanisms to balance the upload contribution across hosts. For example, in our implementation, we limit the upload bandwidth of each peer to be 150% of the stream bitrate.

Stability under dynamics: Node churn can cause intermittent disruptions in data delivery. This can cause frequent interruptions, i.e., the media player starts to *rebuffer* because it is waiting for video segments to render, and affect the user viewing performance. However, since LiveSky is a hybrid CDN-P2P system, it can leverage the edge SNs to minimize the impact of such disruptions without requiring heavy-weight mechanism or using large buffers. The key idea is that peers who suffer performance degradation due to churn (e.g., a parent leaves or the connection is poor) can retrieve data directly from the edge SN until the overlay adapts to the churn and these peers can find suitable parent nodes. This ensures continuous video playback with minimal overhead.

Fast startup: Another common problem in P2P systems is the *startup delay* incurred by new clients. Many P2P streaming systems take at least 30 seconds before the video playback starts [26]. There are two optimizations in LiveSky to reduce the startup delay. First, we reduce the client buffer size to 15 seconds. Second, LiveSky leverages the edge SNs to provide fast startup. When a client first sends a request to a edge SN, the SN responds immediately reply it with a pre-specified number of video segments. As the client video buffer is filled with this content, it starts to play the video. At the same time, it joins the P2P overlay. Once this is done, it no longer needs the data coming directly from the SN. Thus, the client has fast startup and at the same time can transition into P2P mode smoothly.

[2]Ono [17] indirectly infers locality information based on "black-box" observations of CDN redirections. Since we already have this information, our approach is more direct. Alternatively, we can use network coordinates (e.g., [19]) for localizing P2P transfers; we leave this for future work.

[3]LiveSky only served the users in China and PPLive is mainly used in China.

4. DEPLOYMENT AND EVALUATION

There has been considerable interest from researchers and industry alike on deploying hybrid CDN-P2P services [40, 22]. While there have been several measurement studies analyzing pure P2P and pure CDN systems for live streaming and VoD services [20, 21, 24], there is little understanding of how a hybrid P2P-CDN performs in the wild. We believe that this is the first analysis of a commercially deployed hybrid CDN-P2P system for live streaming.

We answer the following questions in this evaluation:

- How does the adaptive scaling work in practice? (Section 4.3)
- What is the performance seen by end-users in terms of the *rebuffering rate* and *startup delay*? How are these performance metrics affected by churn? Do the stability control measures work well in practice? (Section 4.4)
- How effectively can we localize P2P transfers? (Section 4.5)
- Do clients behind NATs contribute adequately? (Section 4.6)

4.1 Deployment

The growth in number of users has been quite rapid since LiveSky has been built and deployed. Figure 4 shows the scale of the system in terms of the peak number of users served over several different events. For the purposes of this study, we focus on the largest event (in terms of total number of users), the 17th CPC National Congress [11] on Oct 22, 2007.[4] During this event, ChinaCache deployed about 500 edge servers in 8 districts each for China NetCom and China TeleCom, the two largest commercial ISPs in China. There were 50 core SNs distributed throughout the districts. These retrieve the content from the source and other core SNs in nearby districts using the server-side distribution discussed in Section 2. Around the core SNs, there are over 400 SNs in the edge layer, scattered within each district. Figure 5 shows the actual location of the core and edge SNs for the event.

The peak number of clients for this event was more than 145,000 with an aggregate bandwidth contribution of roughly 34 Gbps (58.6% of the users) from the CDN nodes and roughly 17 Gbps from the P2P nodes. This event exhibited a flash-crowd like effect around 11am within a few minutes of the beginning of the stream (Figure 6). The average viewing time of the users in the event was 13.5 minutes. Since the event itself only spanned a few minutes, this means that many users stayed and watched a significant portion of the stream. This also suggests that LiveSky was able to provide good user performance even with the sudden surge in demand during the flash-crowd.

4.2 Measurement Methodology

Our measurements are collected from both edge SNs and peers. Each peer reports aggregated statistics to its assigned SN. The SN collects additional information on client arrival and leave patterns. Each edge SN subsequently sends these reports to a log server in the MC. To reduce the overhead of the measurement traffic, each host (peer and edge SN) performs local summarization where appropriate before sending the data to the log server.

There are three types of measurements collected:

1. Per-minute data on total number of active users, number of new users, and number of users leaving the system collected at the edge SN.
2. Client join/leave information including the time a client started, the first login response received, first data packet received, time for the video buffer to fill up initially, and time when the playback started etc.
3. Periodic per-minute reports from each client on the set of peers it currently interacts with, the total number of bytes uploaded and downloaded, playback quality etc. Each client report contains both the global IP and a private IP if the client is behind a NAT.

There are three potential sources of error. First, there can be skews in the clocks across the different nodes in the system. However, this is not a serious issue because we are only interested in coarse-grained metrics rather than fine-grained temporal analysis. Second, the log server may get overloaded and thus some measurements may be lost. We ensure that the clients and edge SNs store such unsuccessful reports and send them at a later time. Third, some clients may leave abruptly, without generating a leave message. However, less than 5% of nodes showed such behavior and this does not bias our results.

Each edge SN operates in isolation and clients in different regions are naturally redirected to local edge SNs. Thus, we can effectively study the system performance and dynamics

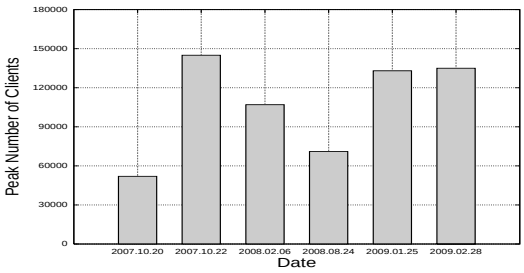


Figure 4: Peak number of clients served by LiveSky during several events since 2007

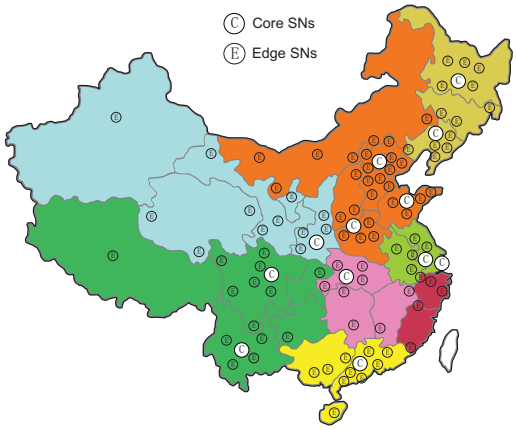


Figure 5: Deployment of the LiveSky CDN nodes in China for the 17th CPC National Congress

[4]The CPC National Congress is a high-impact event of tremendous national interest within China comparable to the US Presidential Inauguration.

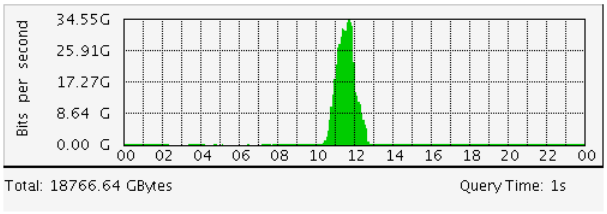


Figure 6: Flash crowd during the 17th CPC Congress on Oct 22nd, 2007

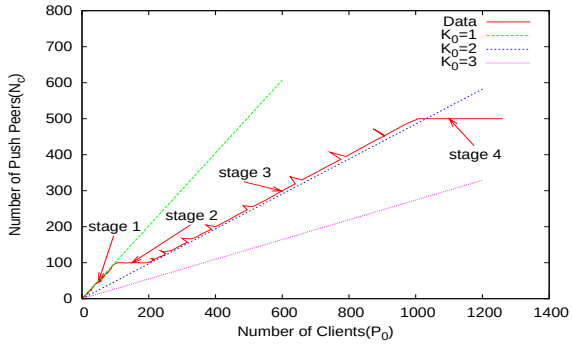


Figure 7: Adaptive scale control at a busy SN

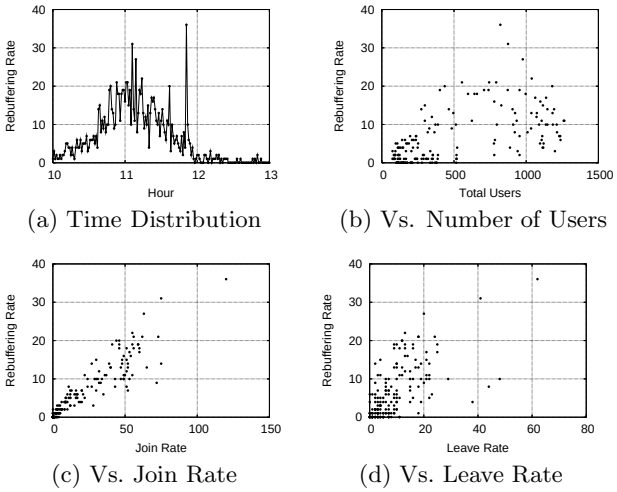
from the perspective of each SN in isolation. Due to skew in the population and viewership, edge SNs see varying request workloads. Some operate in pure CDN mode (i.e., Stage 1) while others switch to the mixed CDN-P2P operation (i.e., Stages 2–4). For the following analysis, we focus on the interesting case: a typical "hotspot" edge region in Beijing that received a sufficiently large request load to cause it to transition into different operating modes. In fact, over 60% of the users in this region were served from the P2P component of LiveSky, compared to the global average of 40%.

4.3 Adaptive Scaling

Figure 7 shows how the adaptive scale control discussed in Section 3.1 works in practice at the Beijing hotspot. As discussed earlier, the key parameter is N_c, the number of *push* peers served directly by the SN. The adaptive control decides a suitable value depending on the offered load (P_0, the x-axis). Here, we configured $N_c^1 = 100$. The bandwidth capacity of the SN is 200Mbps, which at a media rate of 400Kbps translates into $N_c^* = 500$. For reference, we show the expected N_c curves for different values of K_0. We see that the scale control closely follows the expected guidelines, which restrict the minimal N_c suggested by the analysis. (The minor glitches in the curve arise because of practical issues in accurately estimating the total number of clients. The total number of Push peers can be accurately estimated as these clients maintain a persistent connection with the edge SN. However, not all the clients in P2P mode maintain a connection with the edge SN. Some of these clients may leave the system without notifying the edge SN, causing some discrepancy in estimating the total number of clients.)

4.4 User experience

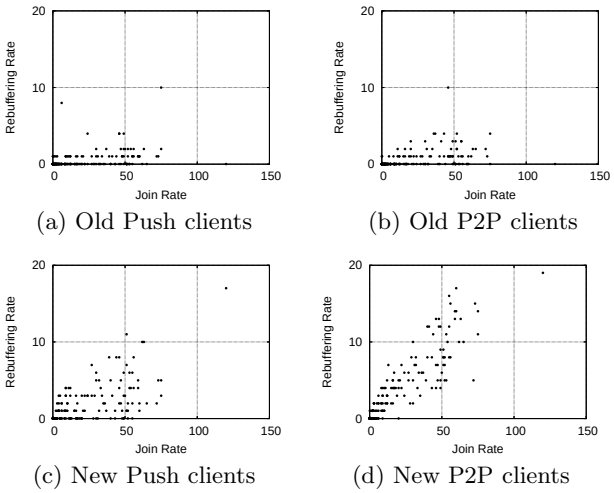
Startup Delay: Anecdotal evidence suggests that the startup delay is a crucial factor in user quality – users are likely to get frustrated and leave if they perceive high startup delays [29]. In LiveSky, 85% of the clients wait less than 15s



(a) Time Distribution (b) Vs. Number of Users

(c) Vs. Join Rate (d) Vs. Leave Rate

Figure 8: Rebuffering rate over time



(a) Old Push clients (b) Old P2P clients

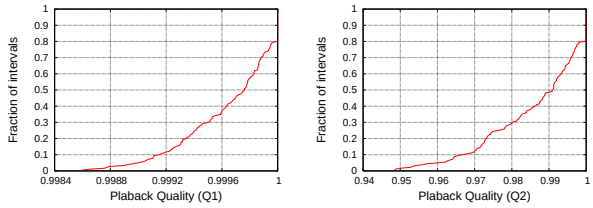
(c) New Push clients (d) New P2P clients

Figure 9: Correlation between join rate and rebuffering rate for different classes of clients

(not shown) for playback to commence from the time the user clicks on the stream hyperlink. Measurements from P2P streaming systems like CoolStreaming show that most clients observe startup delays greater than 30s [27]. Thus, is clear that the hybrid system architecture provides significantly faster startup performance.

Rebuffering Dynamics: We define the *rebuffering rate* as the number of clients that rebuffered for playback per minute. We believe that this metric more accurately captures the quality of the user's viewing experience compared to metrics proposed in previous work. For example, previous work on P2P streaming [27] primarily captures user experience through the failure rate, where a failure event is defined as the inability of the user to start playing the video. Such a coarse-grained measure only captures the user experience during joins; it does not capture the quality when the user is viewing the event.

In Figure 8, we show the rebuffering rate over time and also correlate this with the number of users, the join rate, and the leave rate. While there is a reasonably strong correlation between the rebuffering rate and join rate, there is little or no correlation with the remaining factors.



(a) Q1: Buffered time (b) Q2: Buffering events

Figure 10: Metrics to capture playback quality

Given that the join rate is the dominant factor affecting the rebuffering rate, we do a more fine-grained analysis of the relationship between the rebuffering rate and the join rate. For this, we classify clients along two dimensions: (1) how long they have been in the system (e.g., old vs. new), and (2) whether they are being served by the CDN node or by the P2P system (Push vs. P2P). Figure 9 correlates the rebuffering rate for each of these 4 classes of clients to the aggregate join rate. There is little or no correlation between the rebuffering rate of old users, both Push and P2P, and the join rate. There is a slight correlation between rebuffering rate of the new Push clients and the join rate. There is however, a stronger correlation between the rebuffering rate of new P2P users and the join rate. Even this correlation is substantially less than that observed in P2P-only streaming systems [27]. This suggests that LiveSky is more effective at insulating users from churn-induced effects compared to pure-P2P systems.

Aggregate Quality Indices: We define two quality indices to measure the users' viewing quality in a given measurement interval. The first index is $Q_1 = \frac{T_P}{T_P+T_B}$, where T_P is the total (across all clients) time spent in media playback and T_B denotes the total time spent in buffering in this measurement interval. The second index is $Q_2 = 1 - \frac{N_B}{N_T}$, where N_T is the total number of clients and N_B is the number of clients that experienced some buffering in this measurement interval. Ideally, we want these two quality indices to be as close to 1 as possible, i.e., perfectly smooth playback for all users.

Figure 10 shows the distribution of these two playback quality indices over the entire duration of the live stream. We see that even the worst case values of these quality measures are greater than 0.95 and that the median values for Q_2 and Q_1 are 0.9998 and 0.99 respectively. These show that LiveSky delivers high quality viewing experience.

We make three main inferences from the above results: (1) the clients that have already joined the system are effectively insulated against the dynamics of new joins, (2) the impact of join dynamics on the user quality is significantly lower compared to existing P2P alternatives, and (3) the system provides high quality user viewing experience even in a dynamic environment.

Effect of increasing buffer size: In our deployed system, we use a 15 second long client buffer. We evaluated the potential benefits of increasing the buffer size based on traces collected from the event. Suppose the actual rebuffering time for some client was X seconds. We assume that using a buffer size corresponding to $Y > X$ seconds will alleviate this rebuffering event. Figure 11 shows how the number of rebuffering events per client decreases as we increase the buffer size. For example, when the buffer size is increased from 15s to 30s, the clients percentage with no rebuffering

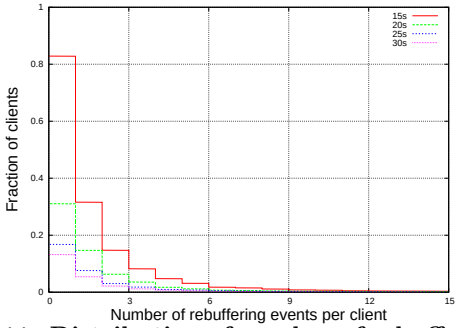


Figure 11: Distribution of number of rebuffer events per client

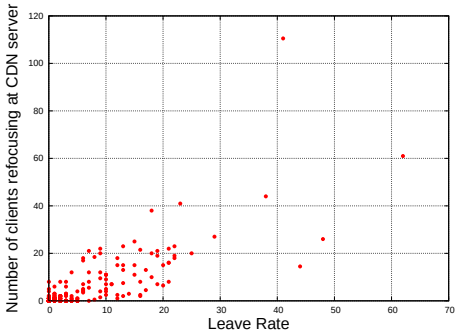


Figure 12: Leveraging the CDN node to minimize disruptions caused when peers leave

increases significantly from 17% to 82%. Beyond a buffer size of 20-25 seconds, we observe that there is a diminishing returns property.

Stability Measures: In Section 3.2 we discussed how LiveSky can provide continuous playback by using the edge SN to temporarily serve nodes that suffer disruptions in churn. Figure 12 confirms that this optimization does help in practice. The figure shows a clear correlation between the number of clients that refocused on the edge SN to obtain the media and the leave rate. Using the edge SN to temporarily serve these affected nodes ensures continuous playback even in the presence of node churn.

4.5 Locality Analysis

Since our P2P overlay is localized on a per-SN basis, we can evaluate the locality-awareness of the P2P transmissions in LiveSky by analyzing the effectiveness of the DNS-based redirections used by the CDN. For example, if the CDN redirection was accurate in assigning 90% of the clients located in Shanghai to the edge SN located in Shanghai, then most of the P2P transmissions associated with this edge SN will be localized. The CDN redirection maps the IP address of the local DNS server of the client to a geographic location using a IP to location database and redirects the client to a suitable edge SN. There are two potential sources of error in this redirection: (1) clients may have misconfigured their local DNS server (e.g., using a static DNS server instead of using a DNS server provided by the upstream ISP) and (2) the IP to location mapping might be out of date. We analyze the accuracy of the redirection by checking if a client's public IP address maps to the same region/ISP as the edge SN assigned to it by the GSLB. In more than 80% of the cases the client is assigned to an edge SN in the same ISP. Thus, most of the P2P traffic in LiveSky is localized within the ISP and this ensures a "network-friendly" operation.

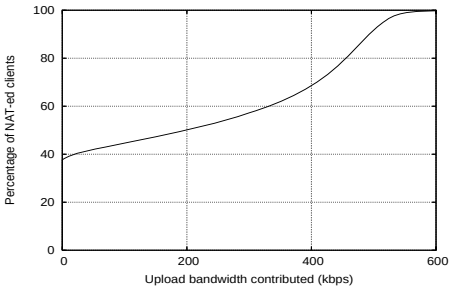


Figure 13: Contribution of NAT-ed clients

4.6 Contribution of NAT-ed clients

Each LiveSky P2P client reports both its private IP address (e.g., 192.168.*) to the edge server and the edge server can obtain its public IP address from the received packets, we can determine if this client is behind a NAT. During the Chinese Congress event, more than 90% of end hosts are behind NATs. Figure 13 shows the distribution of upload bandwidth contribution of NAT-ed clients during this event. While 37% clients behind NAT could not provide upload capacity, the majority of NAT-ed clients contribute significantly. For example, given a video bitrate of 400kbps, nearly 30% of NAT-ed clients contribute 1× the video rate. The CDF ends at 600kbps only because we manually limited the upload contribution to be 1.5× the media rate in the client software. Compared to measurements from prior P2P systems [27] this result suggests a greater contribution from NAT-ed hosts.

5. RELATED WORK

Architectures for P2P streaming: There are two candidate P2P streaming technologies: tree-based (e.g., [18, 16]) and mesh-based (e.g., [42, 9]). The tree vs. mesh question has been a long-standing debate in the research community [36, 31]. Tree-based schemes follow a *push* model and provide low source-to-user delay. However, they have poor performance under churn and do not leverage the available upload capacity effectively since majority of the peers are leaf nodes [29]. Multi-tree constructions alleviate some of these concerns [36, 16]. Mesh schemes [42, 28, 30] follow a *pull* model. Peers typically exchange membership and content information (i.e., the "chunks" each peer currently has to offer) through gossip-like protocols [15]. Mesh approaches are inherently robust to churn and also result in a more equitable use of the upload capacities of peers. However, mesh-based overlays may not deliver the relevant video segments in time and/or have to typically use large buffers to offset the effect of having to wait for video segments.

None of these approaches completely solve the problems arising from the dynamic operating environment, motivating recent proposals for hybrid P2P approaches combining multi-tree and mesh systems (e.g., [41, 39, 37, 36]). LiveSky currently uses these techniques and we can leverage future developments in this area for improving the client-side P2P component.

Hybrid CDN-P2P systems: Xu et al. [40] presented one of the early analysis on hybrid CDN-P2P architectures. They perform an in-depth analysis of the system dynamics which involves a transition between a CDN mode to a P2P mode (similar to our multi-stage approach in Section 3.1) and use simulations to demonstrate that such a hybrid approach is cost-effective. Huang et al. [22] present the potential savings in using hybrid CDN-P2P systems for two major CDNs: Akamai and Limelight. Other researchers have also recommended the use of hybrid CDN-P2P systems (e.g., [25, 32, 33]). Most of this work is based on the simulation or traces collected from pure CDNs. While there are several industry efforts toward implementing hybrid CDN-P2P architectures (e.g., [10, 8]), we are not aware of any published work with respect to the actual system design, implementation, and deployment aspects or real-world measurements of such systems. Thus, we hope that our experiences with LiveSky will serve as a valuable contribution to foster future research in this area.

Measurements of deployed streaming systems: There are many measurement studies of commercial P2P live streaming systems. Hei et al [20] study PPLive using traces collected both by active crawling and passive sniffing. Ali et al. [14] study PPLive and SopCast by passively collecting network traces from clients. Silverston et al. [35] compare four popular P2P streaming applications: PPLive, PPStream, SopCast, and TVAnts using passive sniffing during the broadcast of the 2006 FIFA World Cup. These are largely "black-box" measurements as they are constrained by the inability to instrument the client software or reverse-engineer the control protocols used by these systems. Li et al. [27] analyze CoolStreaming and Wu et al. [38] study UUSee using a more "white-box" approach similar to our measurements by deploying an ActiveX component at peers, which reports peer statistics periodically to a logging server. Such measurement studies have provided valuable insights into network properties, user behavior patterns, and system dynamics to guide the future development of P2P streaming systems. To the best of our knowledge, our work represents the first extensive measurement study on a commercial hybrid CDN-P2P live streaming system, and we hope that our analysis will also serve a similar role.

6. CONCLUSIONS

There has been tremendous interest in both academic and industrial research communities on combining the benefits of traditional CDN architectures and P2P systems for live streaming. However, there have been few real systems and deployments that validate the promise of such a hybrid CDN-P2P architecture. Consequently, we do not have a real understanding of how such an architecture performs "in the wild". The contribution of this paper is a practical one – it presents our experiences in designing and deploying a real-world hybrid CDN-P2P system for live streaming called LiveSky that helps bridge this gap.

In designing LiveSky, we use existing P2P technologies and integrate them with minimal changes to the existing CDN infrastructure. In doing so, we make specific design choices to ensure that the system scales with the number of users, at the same time provides the users with a good viewing experience, i.e., low startup delay and minimal disruptions due to rebuffering, even under churn. We leverage the presence of the infrastructure nodes and the CDN redirection mechanisms to address some well-known deficiencies in P2P systems such as enabling users behind NATs to contribute upload bandwidth and localizing P2P traffic to enhance the network-friendliness.

Our evaluations demonstrate that LiveSky can provide users with good quality even with commodity P2P technologies. There are four natural directions for future work. First, improving P2P technologies for live streaming and VoD like applications is still an ongoing area of research. LiveSky can leverage developments in this area for providing even better performance at scale. For example, scalable coding techniques (i.e., base layer + enhanced layer style codes), can better adapt to node heterogeneity. Second, we can enhance our analysis framework for adaptive scaling to better understand the tradeoffs between scale, quality of service, and CDN operating costs in our analysis, and also extend the analysis to mesh-style P2P overlays. Third, we plan to incorporate better security features e.g., to only allow authorized users to access the stream, ensuring that peers do not intentionally deny service etc. Finally, we would like to achieve more fine-grained instrumentation of clients to be able to better diagnose the root causes of poor client performance.

7. ACKNOWLEDGMENTS

This work is supported by the Project 60673184, 60873254 supported by NSFC, the Project 2007AA01Z419 supported by 863 Program, the Project 2008CB317101 supported by 973 Program, and the Tsinghua-ChinaCache CDN Research Institute Program. Vyas Sekar and Hui Zhang were supported in part by NSF award ANI-0331653. The authors would like to thank Huanying Zou, Kunlong Wang, and Tongqing He for their excellent technical support and expert reviewers for critically reviewing the manuscript.

8. REFERENCES

[1] Akamai. http://www.akamai.com.
[2] ChinaCache. http://en.chinacache.com.
[3] ChinaCache CDN Principle. http://en.chinacache.com/viewtechnique.asp?id=16.
[4] Joost. http://www.joost.com.
[5] Limelight. http://www.limelightnetworks.com.
[6] Live Streaming Continues Momentum With March Madness. http://www.mediapost.com/publications/?fa=Articles.showArticle\&art_aid=101750.
[7] News Sites Struggle to Stream Obama Video. http://bits.blogs.nytimes.com/2009/01/20/news-sites-struggle-to-stream-obamas-innaguration-speech/?apage=1.
[8] Octoshape. http://www.octoshape.com/.
[9] PPLive. http://www.pplive.com.
[10] Redswoosh. http://www.akamai.com/client/.
[11] The 17th CPC National Congress . http://www.chinaview.cn/17thcpc/.
[12] The Numbers Are In, Live Video Online Is Blowing Up. http://www.readwriteweb.com/archives/live_video_big.php.
[13] The Obama Inauguration Live Stream Stats . http://newteevee.com/2009/01/20/the-obama-inauguration-live-stream-stats/.
[14] S. Ali, A. Mathur, and H. Zhang. Measurement of Commercial Peer-to-Peer Live Video Streaming. In *Proceedings of International Workshop on Recent Advances in Peer-to-Peer Streaming*, 2006.
[15] S. Banerjee, S. Lee, B. Bhattacharjee, and A. Srinivasan. Resilient Multicast using Overlays. In *Proceedings of ACM SIGMETRICS*, Oct. 2003.
[16] M. Castro, P. Druschel, A. M. Kermarrec, A. Nandi, A. Rowstron, and A. Singh. SplitStream: High-Bandwidth Content Distribution in Cooperative Environments. In *Proceedings of ACM Symposium on Operating Systems Principles (SOSP)*, 2003.
[17] D. R. Choffnes and F. E. Bustamante. Taming the Torrent: A Practical Approach to Reducing Cross-ISP Traffic in Peer-to-Peer Systems. In *Proceedings of ACM SIGCOMM*, Aug. 2008.
[18] Y. Chu, S. G. Rao, S. Seshan, and H. Zhang. A Case for End System Multicast. In *Proceedings of ACM SIGMETRICS*, June 2000.
[19] F. Dabek, R. Cox, F. Kaashoek, and R. Morris. Vivaldi: A decentralized network coordinate system. In *Proc. of ACM SIGCOMM*, 2004.
[20] X. Hei, C. Liang, J. Liang, Y. Liu, and K. W. Ross. A Measurement Study of a Large-Scale P2P IPTV System. *IEEE Transactions on Multimedia*, 9(8):1672–1687, Dec. 2007.
[21] C. Huang, J. Li, and K. W. Ross. Can Internet Video-on-Demand be Profitable? In *Proceedings of ACM SIGCOMM*, Aug. 2007.
[22] C. Huang, A. Wang, J. Li, and K. W. Ross. Understanding Hybrid CDN-P2P: Why Limelight Needs its Own Red Swoosh. In *Proceedings of NOSSDAV*, May 2008.
[23] G. Huang. Keynote: Experiences with PPLive. In *Proceedings of ACM SIGCOMM P2P-TV Workshop*, Aug. 2007.
[24] Y. Huang, T. Z. J. Fu, D.-M. Chiu, J. C. S. Lui, and C. Huang. Challenges, Design and Analysis of a Large-scale P2P-VoD System. In *Proceedings of ACM SIGCOMM*, Aug. 2008.
[25] T. Karagiannis, P. Rodriguez, and K. Papagiannaki. Should Internet Service Providers Fear Peer-Assisted Content Distribution. In *Proceedings of ACM SIGCOMM IMC*, 2005.
[26] B. Li, Y. Qu, Y. Keung, S. Xie, C. Lin, J. Liu, and X. Zhang. Inside the New Coolstreaming: Principles, Measurements and Performance Implications. In *Proceedings of IEEE INFOCOM*, 2008.
[27] B. Li, S. Xie, G. Y. Keung, J. Liu, I. Stoica, H. Zhang, and X. Zhang. An Empirical Study of the CoolStreaming System. *IEEE Journal on Selected Areas in Communications*, 25(9):1627–1639, Dec. 2007.
[28] X. Liao, H. Jin, Y. Liu, L. M. Ni, and D. Deng. AnySee: Peer-to-Peer Live Streaming. In *Proceedings of IEEE INFOCOM*, Apr. 2006.
[29] J. Liu, S. G. Rao, B. Li, and H. Zhang. Opportunities and Challenges of Peer-to-Peer Internet Video Broadcast. *Proceedings of the IEEE*, 96(1):11–24, Jan. 2008.
[30] N. Magharei and R. Rejaie. PRIME: Peer-to-Peer Receiver-drIven MEsh-based Streaming. In *Proceedings of IEEE INFOCOM*, 2007.
[31] N. Magharei, R. Rejaie, and Y. Guo. Mesh or Multiple-Tree: A Comparative Study of P2P Live Streaming Services. In *Proceedings of IEEE INFOCOM*, 2007.
[32] D. Pakkala and J. Latvakoski. Towards a Peer-to-Peer Extended Content Delivery Network. In *Proceedings of 14th IST Mobile and Wireless Communications Summit*, June 2005.
[33] P. Rodriguez, S. M. Tan, and C. Gkantsidis. On the Feasibility of Commercial, Legal P2P content Distribution. *ACM SIGCOMM CCR*, 36(1), 2006.
[34] J. Rosenberg, J. Weinberger, C. Huitema, and R. Mahy. STUN - Simple Traversal of User Datagram Protocol (UDP) Through Network Address Translators (NATs). *IETF RFC 3489*, 2003.
[35] T. Silverston and O. Fourmaux. Measuring P2P IPTV Systems. In *Proceedings of NOSSDAV*, June 2007.
[36] V. Venkataraman, P. Francis, and J. Calandrino. Chunkyspread: Multi-tree Unstructured Peer-to-Peer Multicast. In *Proceedings of IPTPS*, Feb. 2006.
[37] F. Wang, Y. Xiong, and J. Liu. mTreebone: A Hybrid Tree/Mesh Overlay for Application-Layer Live Video Multicast. In *Proceedings of International Conference on Distributed Computing Systems (ICDCS)*, 2007.
[38] C. Wu, B. Li, and S. Zhao. Diagnosing Network-wide P2P Live Streaming Inefficiencies. In *Proceedings of IEEE INFOCOM*, Apr. 2009.
[39] S. Xie, B. Li, G. Y. Keung, and X. Zhang. Coolstreaming: Design, Theory, and Practice. *IEEE Transactions on Multimedia*, 9(8):1661–1671, May 2007.
[40] D. Xu, S. Kulkarni, C. Rosenberg, and H. Chai. Analysis of a CDN-P2P Hybrid Architecture for Cost-Effective Streaming Media Distribution. *Multimedia Systems*, 11(4):383–399, Mar. 2006.
[41] M. Zhang, J. Luo, L. Zhao, and S. Yang. A Peer-to-Peer Network for Live Media Streaming – Using a Push-Pull Approach. In *Proceedings of ACM International Conference on Multimedia*, 2005.
[42] X. Zhang, J. Liu, B. Li, and T. S. P. Yum. CoolStreaming / DONet: A Data-driven Overlay Network for Peer-to-Peer Live Media Streaming. In *Proceedings of IEEE INFOCOM*, Mar. 2005.