Live Video Streaming Service over Peer to Peer Network: Design, Implementation and Experience*

Yun Tang[1] ,Lifeng Sun[2], Meng Zhang[3], Shiqiang Yang[4] and Yuzhuo Zhong[5],

Tsinghua University, Beijing 100084, P.R.China

Summary
Providing live video streaming service over peer to peer network to a large population of end users remains challenging and interesting, due in large part to the scalability of "self-growing" community of peers, high demanding bandwidth over fluctuant and heterogeneous underlay network, system stability to high churn rate as well as stringent deadline for continuous playback. Although there have been many research efforts advocate various algorithms with respect to application layer multicast, overlay or peer to peer network, few existing works indeed study the impact of the congestion on data transmission paths in such networks which is the key to the performance of streaming applications. Besides, it is more important to investigate practical issues in this arena rather than to follow prescribed theoretical assumptions. Thus in this paper we first consider the spanning tree in multicast as the representative transmission mode and then develop a statistical link model to quantitatively analyze the impact of congestion on the system performance. Since the tree-based structure is vulnerable to network congestion and peer dynamics, while link dependence favors system stability, in the design and implementation of a practical live video streaming system, we adopt a gossip-based overlay construction protocol to accommodate topology change and a composite but efficient streaming mechanism to stream video contents with lower delay than simplex data-driven scheme. The experimental results over PlanetLab indicate that this architecture offers continuous playback, demanding bandwidth and low latency. More interestingly, with the service to totally more than *500,000* users and maximum *15,239* concurrent users provided by only one common streaming server in Feb. 2005, we further exhibit insightful statistical results to reveal system properties, and thus demonstrating that it could scale to reliably support a large and highly dynamic population of end users in the Internet..

Key words:
Peer to peer, Statistical link model, Live video streaming, Practical experiences.

Introduction

The growth of Internet and prevalence of broadband access in past few years have created a great marketplace for the interplay of various information and services. With the turn of century many research pioneers have recognized the potential of streaming live video contents to a large population of end users. They advocate application-layer multicast, overlay or peer to peer network, which all exploit collaborations between users as remedies to traditional client-server approach without the need of network infrastructure support. Although peer to peer network is naturally regarded as the most perspective means since it inherits the effectiveness of group communication, in contrast to file sharing applications, four characteristic challenges distinguish the arena of large-scale live video streaming service, that is, scalability for large "self-growing" community of users, high demanding bandwidth over fluctuant and heterogeneous network, system stability to high churn rate and comfortable visual quality of service in terms of stringent playback deadline.

The universal agreement on peer to peer network for streaming applications is that end users, also called peers or end nodes, self-organize into a logical overlay at application layer and then applications provision data or services along the edges of the overlay through unicast connections. Currently, there are a number of proposed overlay structures and service management protocols in peer to peer based approaches for media contents dissemination, which fall into the following categories. In the first category, such as NARADA[1], RMX[2], Overcast[3] and some other tree or mesh-based tree structures all have in common that they involve a spanning tree for data delivering, which is vulnerable to peer dynamics as well as network congestion, and enforces tremendous delay to those with more depth. In the second category, NICE[4] and ZIGZAG[5], to name a few, perform distributed algorithm and adopt hierarchically clustering structure. Apparently there are specific internal nodes with heavier load and their failure or departure would have grievous impact to large numbers of descendants. In the third category, CoopNet[6] and SplitStream[7], for instance, deploy multiple distribution trees, resulting in either heavy workload on certain internal nodes or dedicated relay nodes for forward traffic. Moreover, all of those previous works suffer the same drawback that the regularity of overlay architecture makes it difficult to accommodate topology change which is fairly common in peer to peer environment. Accordingly, PRM[8] and DONet[9] explore the inherent stability of gossip-based membership management to provide better stability and scalability.

*Manuscript revised March 24, 2006.
This work was supported by National Natural Science Foundation of China under Grant No. 60503063 and 60432030. Some preliminary work has been accepted by ACM Multimedia 2005

However, it is clear that any congestion occurring on the overlay links will influence other downstream links, and in turn lead to visual quality degradation. Most of existing works employ TCP or TCP-Friendly protocol to perform congestion control, but none of them quantitatively investigates the impact of congestion on the performance, which is the key to the streaming applications. Besides, as service availability and system evolution emerge as the crucial components of peer to peer applications, it is more important to examine practical issues rather than to follow a variety of theoretical assumptions in such an area. Although DONet[9] presented a practically feasible approach, the data-driven streaming scheme definitely introduced tremendous latency and it did not provide further information about practical system properties. In this context, the introductory summary motivates the critical research questions, which may be summarized as follows.

1) What is the impact of congestion on the stability of system and how to quantify and evaluate it?

2) How to design and implement a scalable, stable and efficient live video streaming system over peer to peer network?

3) What can we learn from the experiences when such a system is successfully deployed over Internet?

Accordingly, the objective of this paper is to present our research work towards answering these questions, which makes the following contributions. First of all, we consider the typical spanning tree in overlay edges as the representative transmission mode and then develop a statistical link model to capture the impact of congestion on the system performance. Secondly, inspired by the positive feedback of link dependence in terms of stability, we further design and implement a practical system by deploying a gossip-based overlay construction protocol to adapt to peer community evolution and a composite but efficient scheduling streaming mechanism to stream video contents with lower delay than simplex data-driven scheme. The experimental results over PlanetLab[10] indicate that this architecture offers continuous playback, demanding bandwidth and low latency. Thirdly and more interestingly, with the service to totally more than ***500,000***users and maximum ***15,239*** concurrent users provided by only one common streaming server in Feb. 2005, we further exhibit insightful statistical results to reveal system properties, and thus demonstrating that it could scale to reliably support a potentially large and highly dynamic population of end users in the Internet.

The remainder of this paper is organized as follows. Sec.2 concentrates on the statistical link model for the analysis of the impact of congestion. In Sec.3 we mainly present the design and implementation of a practical system with gossip-based overlay and composite streaming mechanism. Some experimental results over PlanetLab are also exhibited. Then in Sec.4 we provide preliminary statistical results from deployment over Internet, followed by the system properties. Sec. 5discusses future work and ends this paper.

2. Statistical Link Model for Congestion Analysis

As mentioned previously, in peer to peer network, applications in general provision data or servicesalong with the overlay edges. When the congestion occurs on one link, the nodes both at the same level and under the congested links will greatly suffer from this. Clearly, for live video streaming services, intrinsic to the notion of congestion is the frequently unfavorable visual quality degradation at end users, which motivates us to investigate the issue of congestion in peer to peer environment. Additionally, since the typical transmission mode is to utilize various spanning trees as data delivery paths, in this section, we mainly target on an unbalanced binary tree as the representative example and accordingly develop a statistical link model to quantify and evaluate the impact of congestion.

2.1 Unbalanced Binary Tree Model

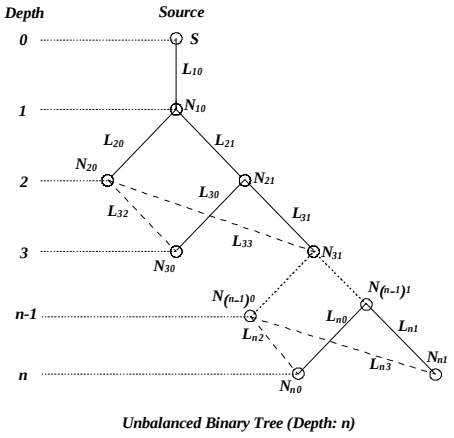

Fig. 1 A Case of Unbalanced Binary Tree

Fig.1 schematically depicts an unbalanced binary tree with the depth n. For simplicity without loss of generality, we first assume that nodes suffering from congestion will leave the tree simultaneously to reduce the congestion and then we could introduce the first metric, i.e. stability factor of the spanning tree to evaluate the impact of congestion

as follows. In a tree with total $T(n)$ $T(n)$ members, if the number of nodes which are affected after one node leaves the tree is denoted by δ, then the average δ could be denoted by $E(\delta)$. Thus the stability factor is defined as $S(n) = E(\delta)/T(n)$. A smaller $S(n)$ means that fewer nodes perceive the change due to congestion, and therefore a more stable spanning tree for date transmission. Aiming to unify the discussion formally, we formulate the tree structure in the following definition.

Definition 1: An unbalanced binary tree typically comprises a set N of nodes and a set L of links which are correspondingly labeled in Fig.1 with the tree depth n. Let N' denote the set of nodes except the source, so the nodes within N' all have the possibility to become the congestion bottleneck. If the congestion probability of link L_{ij} is denoted by p_{ij}, then we have:

$$\begin{aligned} &N' = \{N_{10}, N_{20}, N_{21}, \ldots, N_{n0}, N_{n1}\}, \\ &N = \{S\} \bigcup N', \\ &L = \{L_{10}, L_{20}, L_{21}, \ldots, L_{n0}, L_{n1}\}, \\ &0 < p_{ij} < 1, 0 < i \le n. \end{aligned} \tag{1}$$

It is apparent that the congestion state of link L_{ij} is only dependent on that of its direct upstream link $L_{(i-1)1}$. Note that in order to explore the dependence between adjacent links, here we intentionally introduce the virtual link set $L' = \{L_{32}, L_{33}, \ldots, L_{n2}, L_{n3}\}$, as shown in Fig.1. Therefore, we could further derive the definition of "conceptual link".

Definition 2: In the tree defined as in Definition 1, conceptual link $\overline{L}_i$ represents any transmission links of depth i which packets stream through, where $0 \le i \le n$.

For instance, conceptual link $\overline{L}_3$ consists of links L_{30}, L_{31} and virtual links L_{32}, L_{33}. It should be stressed that the essential purpose of conceptual link $\overline{L}_i$ is to depict the possibility that packets successfully get across the available links of depth i, either existing links or virtual links, to arrive at nodes with more depth. Intuitively, if certain numbers of virtual links are deployed to act as the backup of existing links, it is easy to find out that they will improve the system stability in case of congestion, which would be analyzed in subsection 2.2.

If we utilize a random variable X_i to describe the congestion state of $\overline{L}_i$, we have $X_i \in \{0,1\}$. For ease of exposition, let $p_i = P\{X_i = 1\}$ denote the congestion probability of $\overline{L}_i$, then we could deduce:

Theorem 1: In the tree defined as in Definition 1, the congestion probability p_i of conceptual link $\overline{L}_i$ could be calculated as:

$$P(X_i = x_i) = \begin{cases} p_i = p_{i1}, x_i = 1; \\ 1 - p_i, x_i = 0; \end{cases} \tag{2}$$

Furthermore, the conceptual link is also dependent only on the direct upstream link and thus the congestion states of adjacent virtual links reflect Markov property:

$$\begin{aligned} &P\{X_k = x_k \,|\, X_{k-1} = x_{k-1}, X_{k-2} = x_{k-2}, \ldots, X_1 = x_1\} \\ &= P\{X_k = x_k \,|\, X_{k-1} = x_{k-1}\}; \end{aligned} \tag{3}$$

Towards this end, the set of random variables $\{X_1, X_2, \ldots, X_{n-1}, X_n\}$ have constructed a two-state discrete Markov chain. Accordingly, we could define another metric to quantify the congestion, that is, the probability distribution $\Phi(\overline{L}_i, n)$ that conceptual link $\overline{L}_i$ becomes the congestion bottleneck, which is obtained in the following equation.

$$\Phi(\overline{L}_i, n) = \begin{cases} P\{X_1 = 1\}, i = 1; \\ P\{X_1 = 0\} \cdot \prod_{j=1}^{i-2} P\{X_{j+1} = 0 \,|\, X_j = 0\} \\ \qquad \cdot P\{X_i = 1 \,|\, X_{i-1} = 0\}, 2 \le i \le n; \end{cases} \tag{4}$$

Now it comes to the question that how we obtain the transition probability $P\{X_{i+1} = x_{i+1} \,|\, X_i = x_i\}$. Inspired by [12], we also introduce a dependency-degree factor α, where $\alpha \in [0,1]$, to represent the dependency between random variables. By similar reasoning in [12], we further deduce the one-step transition probability and probability distribution as follows.

Theorem 2: If $\vec{\alpha} = (\alpha_1, \alpha_2, \alpha_3, \ldots, \alpha_{n-1})$ denotes the dependency-degree factor vector between X_i and X_{i+1}, and $\vec{p} = (p_1, p_2, p_3, \ldots p_n)$ denotes conceptual link congestion probability vector, then $P\{X_{i+1} = x_{x+1} \,|\, X_i = x_i\}$ can be calculated by Eq.(5)

$$
\begin{aligned}
&P\{X_{i+1}=0 \mid X_i=0\} \\
&=\begin{cases} 1-(1-\alpha_i)\cdot p_{i+1}, p_i \geq p_{i+1}; \\ (1-\alpha_i)\cdot(1-p_{i+1})+\alpha_i\cdot(\dfrac{1-p_{i+1}}{1-p_i}), p_i < p_{i+1}; \end{cases} \\
&P\{X_{i+1}=1 \mid X_i=0\} \\
&=\begin{cases} (1-\alpha_i)\cdot p_{i+1}, p_i \geq p_{i+1}; \\ (1-\alpha_i)\cdot p_{i+1}+\alpha_i\cdot(\dfrac{p_{i+1}-p_i}{1-p_i}), p_i < p_{i+1}; \end{cases}
\end{aligned} \tag{5}
$$

Combining the Eqs. (4) and (5), we further have

$$
\Phi(\overline{L}_i,\overline{\alpha},\overline{p},n)=\begin{cases} p_1, i=1; \\ (1-p_1)\cdot(1-\alpha_{i-1})p_i \\ \cdot\prod_{j=1}^{i-2}[1-(1-\alpha_j)p_{j+1}], 2\leq i\leq n; \end{cases} \tag{6}
$$

2.2 Statistical Properties

To simply the analysis, in the tree of depth n defined as in Definition 1, we assume $0<p_{ij}=p<1$ and $0<\alpha_i=\alpha<1$. Then the statistical properties of unbalanced binary tree have following theorem.

Theorem 3: Let $\zeta=1-(1-\alpha)p$, probability distribution $\Phi(\overline{L}_i,\alpha,p,n)$ that $\overline{L}_i$ becomes the congestion bottleneck and the stability factor $S(\alpha,p,n)$ is given by Eqs.(7) and (8).

$$
\Phi(\overline{L}_i,\alpha,p,n)=\begin{cases} p, i=1; \\ (1-p)(1-\zeta)\cdot\zeta^{i-1}, 2\leq i\leq n; \end{cases} \tag{7}
$$

$$
S(\alpha,p,n)=\begin{cases} p, \alpha=1; \\ 1-\dfrac{(1-p)(2-\zeta^{n-1}-\zeta^n)}{(1-\zeta)(2n-1)}, 0<\alpha<1; \end{cases} \tag{8}
$$

And, if the virtual links set L' is deployed to take the responsibility of backup links, then it is easy to derive the improved stability factor $S'(\alpha,p,n)$ as given by Eq. (9), where $\eta=1-(1-\alpha)p^2, \theta=1-(1-\alpha)p^4$.

$$
S'(\alpha,p,n)=\begin{cases} p, \alpha=1; \\ 1-\dfrac{1-p}{2n-1}-\dfrac{2\eta(1-p)(1-\theta^{n-1})}{(1-\theta)(2n-1)}, 0<\alpha<1; \end{cases} \tag{9}
$$

The detailed proof can be referred to [11]. Based on above quantitatively statistical results, we could evaluate three particular impacts of congestion in terms of system stability. At first, observe that $\Phi(\overline{L}_i,\alpha,p,n)$ is a decreasing function of α, which implies that larger link dependence leads to smaller $\Phi(\overline{L}_i,\alpha,p,n)$ and in turn results in lower possibility with which the links at depth i become congestion bottleneck. Second and more profound, α also offers appealing influence on the stability factor of the tree. In the view of fact that the larger α, the smaller $S(\alpha,p,n)$, we recognize that improving link dependence positively favors tree stability and thus less nodes will be affected when congestion occurs. At last, it is clear that the improved stability factor $S'(\alpha,p,n)$ always achieve smaller values than those of $S(\alpha,p,n)$. The distinct gap between these two factors substantially comes from the virtual links which play the role of backup links.

So far we take the unbalanced binary tree as the representative example towards understanding the impact of congests in peer to peer network in this section. Since the concept of "peer to peer" is predominantly advocated by academic and commercial communities as a scalable and cost-effective alternative to traditional client-server or infrastructure-based CDN approaches in the area of large-scale Internet services, issues and problems take more on a practical dimension. It is thus important to investigate service availability and system properties in reality, looking beyond a variety of theoretical analysis. To this end, we will focus on the design and implementation of a practical system in next section.

3. Practical System for Live Video Streaming

As argued, it is commonly believed that peer to peer based media streaming systems basically involve two fundamental issues. One is how to design a scalable and stable overlay structure to adapt to peer community evolution while retaining suitable for the scenario of media streaming in terms of packet delivery deadline. The other is how to implement an efficient streaming mechanism, based on existing overlay structure, to facilitate media contents dissemination while ensuring continuous playback at end users. Therefore in subsection 3.1 we divide the practical system design into two parts, that is, overlay structure and streaming mechanism and then provide experimental results over PlanetLab to demonstrate the performance of this prototype in subsection 3.2.

It should be pointed out that here we mainly show its architecture and characteristic features. More detailed descriptions could be referred to [13].

3.1 Overlay Design and Streaming Mechanism

A characteristic of peer to peer network is that individual users join or leave the system independently and continuously, resulting in highly frequent overlay structure maintenance. And for media streaming applications, packet loss due to transient congestion or link switch operations often leads to poor visual quality experiences at end users. In addition, recall that the link dependence conduces to system stability while tree-based structure is vulnerable to congestion as well as high churn rate of peers. Hence all these factors come into play to motivate us to consider the gossip-based overlay as the fundamental philosophy in the design of overlay structure, owing to its inherent stability and scalability [8].

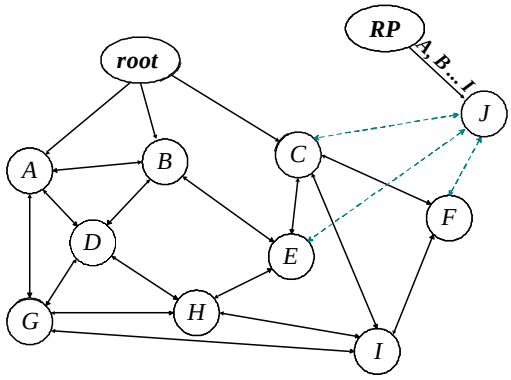

Fig. 2 A Case of Overlay Structure

It is well-known that gossip-based unstructured overlay in general performs distributed membership management, so each peer in our design also takes charge of maintaining part of total members that are currently active in the service community. And intuitively everyone is able to select a proportion of nodes in the fractional member list as the upstream suppliers to fetch video contents, which are so called neighbors. From this viewpoint, the chief task of overlay construction protocol is to help each peer find appropriate neighbors and thereby all those peers self organize into an overlay in a gossip fashion, according to the cooperative relationship between peers. It is worth stressing that in order to provide sufficient bandwidth for video streaming, each peer is usually supplied with more than three neighbors. Fig.2 illustrates a case of the overlay structure. As shown, live video contents are originally streamed from a source server, i.e. the root. And a well-known Rendezvous Point, termed RP, is deployed to guide newly joined nodes to startup.

In such an overlay structure, each peer needs to maintain two lists, namely member list and neighbor list. The former contains a fraction of active members currently in the overlay and is exchanged periodically among peers to refresh the information of members. Note that every one will update its own member list in accordance with the lists obtained from its neighbors. One that quits, fails or experiences long time congestion will be detected by "heartbeat"-like mechanism and removed from the list, resulting in corresponding maintenance in a distributed manner. At the same time, besides the membership evolution, each peer also refines its neighbor list in response to two kinds of events. In one case, as a common circumstance in autonomous overlay network, when one or more neighbors depart or fail, a refinement procedure of neighborship will be awoken to seek substitutes in active members. In the other case, the volume of bidirectional traffic between peer pairs also triggers the refinement mechanism iteratively if the exchanged data in previous round is below a predefined threshold.

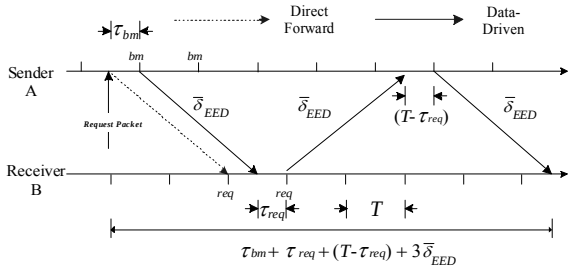

Fig. 3 Latency Comparison between Direct Forwarding and Data-Driven

As mentioned above, while the overlay has been built up and maintained by all participants, it comes to the question how every peer obtains media segments efficiently with the stringent constraint of playback deadline. Although the experimental and practical results both indicate the feasibility of data-driven scheme proposed in DONet[9], as a matter of fact, it introduces tremendous latency during the "three handshakes" process. For sake of presentation, Fig. 3 depicts the schematic comparison between the data-driven and direct forwarding transmission scheme. As shown, in the data-driven scheme, receiver *B* firstly needs to wait for the buffer map generated from sender *A* and if *B* does require an absent packet, it then sends a request message back to sender *A*. Till then, sender *A* starts to transmit the demanded packets. More formally, we denote the average latency in data-driven and direct forwarding scheme as $\overline{T}_{dd}$ and $\overline{T}_{df}$ respectively, which could be approximately calculated as

$$\overline{T}_{dd} = \tau_{bm} + \tau_{req} + (T - \tau_{req}) + 3\overline{\delta}_{EED}, \overline{T}_{df} = \overline{\delta}_{EED}.$$

Apparently the latency of data-driven scheme is far larger than that of direct forwarding transmission. So the simplex

data-driven scheme is not enough for live video streaming services on the Internet, due to the highly delay-sensitive nature of these applications.

In view of fact that direct forwarding transmission achieves rapid distribution by "pushing" mechanism, while date-driven scheme offers assuring transmission for absent packets by "pulling" mechanism, we resort to a composite approach of both them to deliver packets. The fundamental design philosophy is to encourage each one to push packets to its neighbors without any explicit request, while retaining the exchange of buffer maps so as to recall pull mechanism to snatch desirable packets when packet loss occurs. In addition, at the process of startup, peers prefer pull mechanism to ensure the initial visual quality and since then each of them relays the packet as soon as it arrives. Clearly, the acceleration of "push+pull" streaming mechanism will definitely outperform the simplex data-driven scheme in terms of transmission latency. Although the idea of this composite streaming mechanism seems simple and intuitive, it really poses many challenges when it comes to the practical implementation. The first problem that comes to mind is the bandwidth allocation among multiple neighbors. As every peer belongs to different administrative domains and is inherently heterogeneous, we must balance its service capability with the contribution. Accordingly we employ the volume of bidirectional traffic between every peer pair in last time slot as the evaluation metric, which guides the allocation of streaming rates for next interval. Furthermore, in the peer to peer network, the packets received at the peer could not be expected to arrive in order, which incubates another problem, that is, distinguishing that a packet is indeed lost in the transmission and it is on the road. Without the correct decision, it could be imagined that certain packets will be either never received or transferred many times. Therefore we utilize two up-to-date thresholds here. One is at the sender side to avoid redundancy, while the other is located at the receiver for the request of lost packets. With the aid of a tracker, these two thresholds offer an effective solution for this problem [13].

Towards this end, we have presented the architecture overview of overlay design and streaming mechanism. In the practical implementation, we intentionally decompose key components of each peer into three levels from a general point of view. Fig.4 gives an illustration to those key modules. At the top, the application level comprises a player to playback video frames and a predefined buffer to cache packets. At the intermediate resource level, a streaming scheduler is involved to manage packets exchange with neighbors. At the bottom, the structure level includes a neighborship manager and a membership manager. In next subsection, we will exhibit some experimental results over PlanetLab to demonstrate the performance of this architecture.

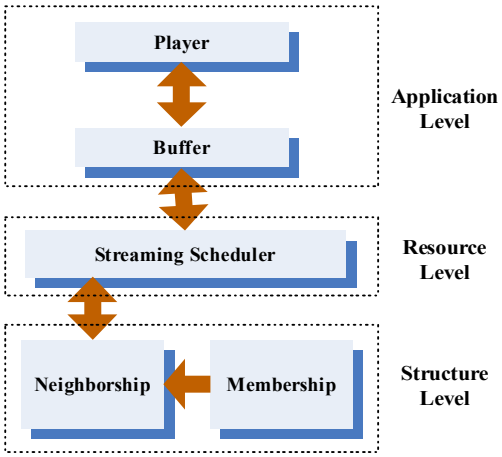

Fig. 4 Key Components in Each Peer

3.2 Experiment Results over PlanetLab

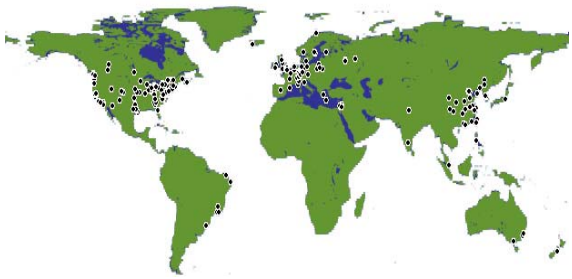

Fig.5 PlanetLab Experiment Platform

We have conducted extensive experiments based on above architecture in PlanetLab testbed, which is an open platform for experimenting, developing, deploying and accessing planetary-scale services. Fig.5 depicts a snapshot of active nodes across continents. Note that now there are more than 500 voluntary nodes altogether, but during our experiments only about 300 nodes were available simultaneously. So in the following discussion the maximum number of nodes will not exceed 350. With the consideration of dynamics in peer to peer environment, we intentionally review system performance in static and dynamic environment separately. In static experiment, each peer will keep alive at all times, typically for 1 hour or more. In contrast, each peer could be turn on or off for a while in dynamic environment. Moreover, we also divide all the experiments into two sets, one is with no limitation on upload capacity of each peer while the other

constrains the upload bandwidth to 500kbps, which imitates the DSL connections. Table 1 summarizes the experiment setup over PlanetLab.

Table 1: Experiment Setup over PlanetLab

Parameter	***Value or Range***
Group Size	about 300~340
Number of neighbors	5
Encoding Rate	310 kbps
Packet Rate	30 packets/sec
Buffer Map Interval	1 second
Request Interval	1 second
Average online time	100 seconds
Average offline time	10 seconds
Upload Bandwidth Limitation 500kbps	Disabled/ Enabled

Since we have claimed that the data-driven scheme induces tremendous latency, we first define the latency between the sampling time of the root and the playback time at peers as **absolute delay**, to explicitly evaluate this sensitive metric in live video steaming applications. Besides, the ratio of packets arriving before playback deadline to total packets is defined as **delivery ratio**, which reflects the visual quality at end users to some extent. The third metric **α-playback-time** denotes the minimum absolute delay at which the delivery ratio is greater than α , where $0 < \alpha < 1$.

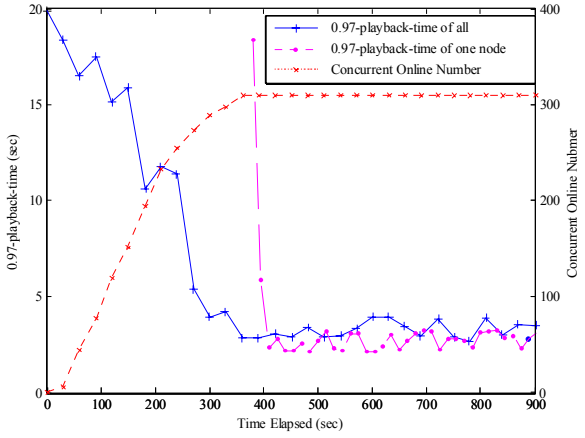

Fig.6 Convergence of 0.97-playback-time in PlanetLab

First of all, we would like to examine the convergence of delivery ratio when the peer joins the service community which is constructed as in the previous subsection. Fig. 6 shows the system evolution when 310 peer machines enter the experiment one by one. Not surprisingly, the average α-playback-time(α =0.97 here) descends to approximately 2.5 seconds quickly after all peers finish the join procedure. And the dashed line in Fig. 6 depicts that the last peer's 0.97-playback-time converges to 2 to 3 seconds within about 20 seconds, which indicates that this architecture works well towards providing fast initial startup with favorable quality of service.

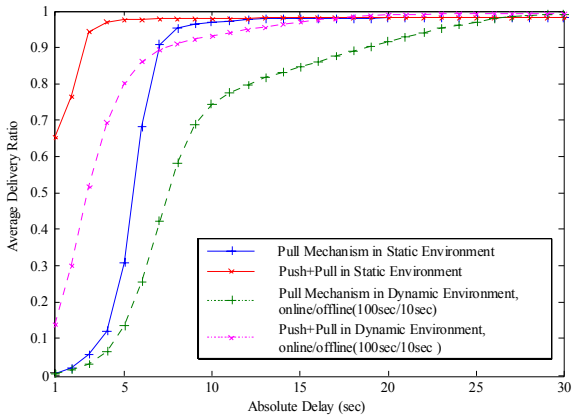

Fig.7 Comparison between Pull and Push+Pull Mechanism without Upload Bandwidth Limitation

Then we compare the performance of the pull mechanism in data-driven scheme with that of proposed "push+pull" mechanism. Note that in dynamic experiments, the average online time and offline time was set to 100 seconds and 10 seconds, respectively, as listed in Tab.1. Fig.7 reveals that the proposed mechanism can achieve an equivalent α-playback-time with much lower absolute delay than the pull mechanism. It is apparent to find out whether in static or dynamic environment, "push+pull" mechanism always exhibits much lower 0.97-playback-times. The solid lines show that the 0.97-playback-times of the former are 4 seconds in static experiments and 16 seconds in dynamic environments, as compared to 11 seconds and 28 seconds of pull mechanism.

Fig.8 further illustrates the comparison between pull and "push+pull" mechanism with 500kbps upload bandwidth constraint. Likewise, we can clearly figure out that proposed streaming mechanism shows better performance with respect to lower delay. The solid lines indicate that the average 0.97-playback-time of "push+pull" are 13 seconds and 20 seconds in static and dynamic environment respectively, while the corresponding values of pull mechanism are about 18 seconds and 24 seconds.

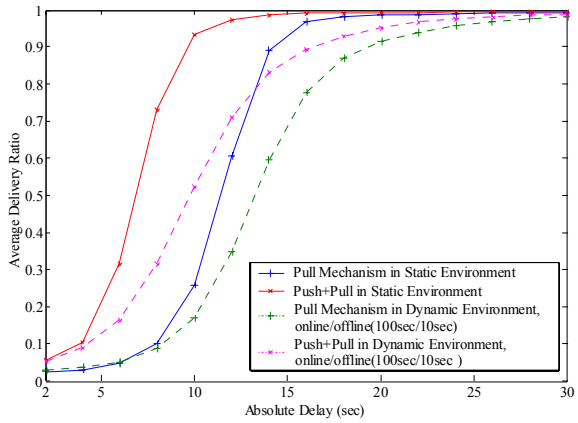

Fig.8 Comparison between Pull and Push+Pull Mechanism with 500kbps Upload Bandwidth

In summary, this architecture provides a scalable, stable as well as efficient peer to peer based live video streaming approach within the scope of testing prototype. Inspired by positive feedback from those experiments, we further extended this prototype and addressed many practical issues to release a public desktop software[14]. Some deployment experiences then will be discussed in next section.

4. Deployment Experiences

Fortunately, this software was deployed by CCTV(China Central Television Station)[15], the largest TV station in China, to live broadcast the Spring Festival Evening over global Internet in Feb. 2005. We encoded the video signals into 300kbps distribution sessions by a common streaming server. As evidenced by service logs, there were more than **500,000** users from about **66** countries over the world enjoying the show via our software. More excitingly, the maximum number of concurrent users reached its peak **15,239** at Feb. 8th, 2005. It is worth noting that only 200 users were configured to directly connect to the streaming server and the rest of users obtained contents through the cooperation between each other. Fig.9 shows the evolutionary system capacity over time. Observe that the average number of peers in this streaming service community was nearly beyond 10,000 all the time. With requesting peers later becoming supplying peers, the systems' total capacity was scalably amplified to hundreds times of traditional client-server approach. Fig. 10 gives a brief description of location diversity of end users according to IP address. We believe that most of 22\% users within other countries are Chinese abroad.

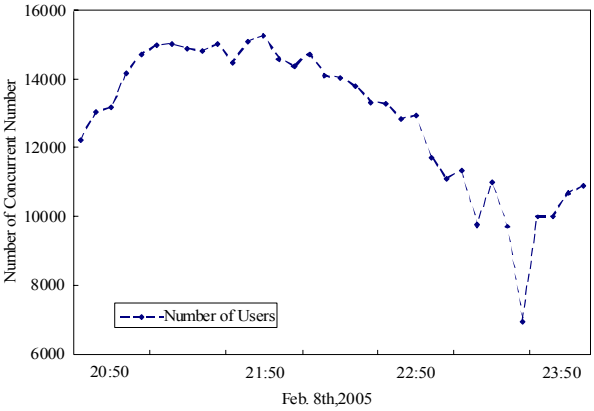

Fig.9 Evolutionary Service Capacity at Feb 8th, 2005

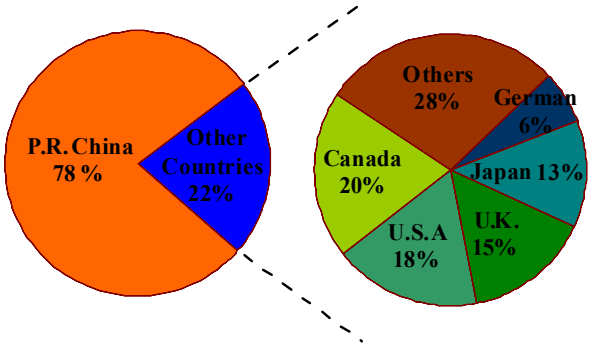

Fig.10 Location Diversity of End Usersh

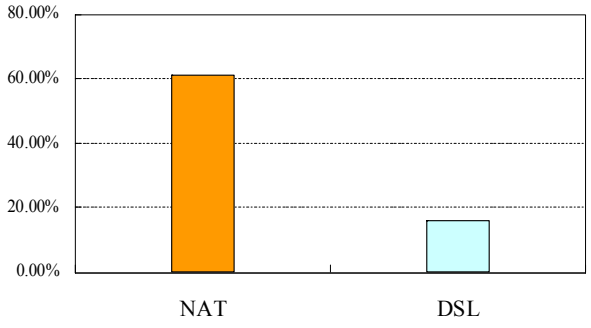

Fig.11 Connection Diversity of End Users

Besides, since each participant in overlay may typically reside in geographically dispersed location, and access the Internet via heterogeneous access technologies, Fig.11 traced that there were nearly 60.8\% users are behind kinds of NAT and at least 16\% users access the Internet via DSL connections.

In the following analysis, another distinguished aspect of peer to peer application, i.e. high churn rate will be presented. Due to page limitation, here we only exhibit the change of request rate over time. Fig.12 and Fig. 13 each depicts the request rate per 30 seconds from 19:55 and 23:00 respectively. It is obvious that even before the start point of this evening show (20:00 at Feb. 8th) or near the end point of it (00:30 at Feb. 9th), the average request rate always kept a record of hundreds of users, except that it occasionally rushed to a peak beyond 3,700. This sporadic flash crowd was also perceived by us during monitoring the system at that night. The successful deployment confirms that this architecture offers outstanding scalability and stability, and thus it is applicable for large-scale live video streaming.

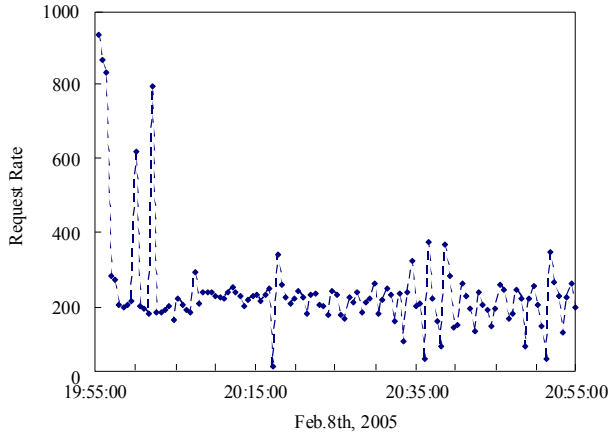

Fig.12 Request Rate from 19:55:00 (per 30 sec)

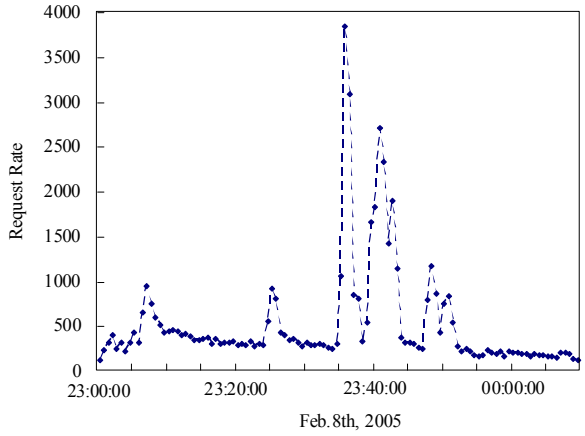

Fig.13 Request Rate from 23:00:00 (per 30 sec)

5. Conclusion and Future Work

In this paper, we target on the design, implementation and experiences in the arena of live video streaming service over peer to peer network. Our contributions come into three flavors. First of all, we develop a statistical link model to quantitatively analyze the impact of congestion, by exploiting an unbalanced binary tree as representative example. Then, we design and implement a practical system by deploying a gossip-based overlay protocol and a composite "push+pull" streaming mechanism with lower delay than simplex data-driven scheme. The experimental results over PlanetLab indicate the efficiency of this architecture. Thirdly, deployment experiences of the service to totally more than 500,000 users and maximum 15,239 concurrent users further demonstrate that it could scale to reliably support a large and highly dynamic population of end users in the Internet. In the near future, we will exploit user characteristics from log information to improve the performance of existing service application. Some incentive mechanisms are also very interesting to further study.

Acknowledgments

The authors would like to thank colleagues Jian-Guang Luo, Yu-Zhe Jin, etc. at Tsinghua University and CCTV. We also gracefully thank anonymous reviewers for insightful comments.

References

[1] Chu Yanghua, Rao S.G., Zhang Hui. A Case for End System Multicast.Proceedings of ACM SIGMETRICS, Santa Clara, CA, USA, ACM Press, pp.1-12, 2000.

[2] Chawathe Y, McCanne S, Brewer EA. RMX: Reliable Multicast for Heterogeneous Networks. Proceedings of IEEE INFOCOM, Tel Aviv, Israel, IEEE Communication Society, pp.795-804, 2000.

[3] Jannotti J, Gifford D, Johnson KL, Kaashoek MF, O'Toole JW. Overcast: Reliable Multicasting with An Overlay Network. Proceedings of the Fourth Symposium on Operating System Design and Implementation (OSDI), San Diego, CA, USA, USENIX Association, pp.197-212, 2000.

[4] Suman Banerjee, Bobby Bhattacharjee, Christopher Kommareddy. Scalable Application Layer Multicast. Proceedings of ACM SIGCOMM 2002, August 2002.

[5] Duc A. Tran, Kien A. Hua, Tai Do. ZIGZAG: An Efficient Peer-to-Peer Scheme for Media Streaming. Proceedings of IEEE INFOCOM 2003, April 2003.

[6] Venkata N. Padmanabhan, Helen. J. Wang, Philip A. Chou. Distributing Streaming Media Content Using Cooperative Networking. Proceedings of NOSSDAV 2002, May 2002.

[7] Miguel Castro, Peter Druschel, Anne-Marie Kermarrec, Animesh Nandi, et al. SplitStream: High-bandwidth Multicast in Cooperative Environments. Proceedings of ACM SOSP 2003, October 2003.

[8] Suman Banerjee, Seungjoon Lee, Bobby Bhattacharjee, Aravind Srinivasan. Resilient Multicast Using Overlays. Proceedings of ACM SIGMETRICS 2004, June 2004.
[9] Xinyan Zhang, Jiangchuan Liu, Bo Li, Tak-Shing Peter Yum. CoolStreaming/DONet: A Data-driven Overlay Network for Efficient Live Media Streaming. Proceedings of IEEE INFOCOM, March 2005.
[10] PlanetLab website, http://www.planet-lab.org/
[11] Baitao Long, Yun Tang, Yuzhuo Zhong. Improving the Stability of Spanning Trees for Application-layer Multicast [R]. Tsinghua University, Department of Computer Science and Technology, 2003.
[12] Zhang Xi, Shin KG. Statistical Analysis of Feedback-Synchronization Signaling Delay for Multicast Flow Control. Proceedings of IEEE INFOCOM, Anchorage, Alaska, USA, IEEE Communications Society, pp.1133-1142 2001.
[13] Meng Zhang, Li Zhao, Yun Tang, Jianguang Luo, Shiqiang Yang. Large-Scale Live Media Streaming over Peer-to-Peer Networks through Global Internet. Proceedings of ACM Workshop on Advances in Peer-to-Peer Multimedia Streamingin, ACM Multimedia 2005, November 2005.
[14] Gridmedia website, http://www.gridmedia.com.cn/
[15] China Central Tevevision Station, http://www.cctv.com/